

19th International Society for Music Information Retrieval Conference

September 23 - 27, 2018
Paris, France

The logo for the 19th International Society for Music Information Retrieval Conference (ISMIR 2018). It features the text "ismir" in a white, lowercase, sans-serif font, with the "i" having a square dot. Below "ismir" is the year "2018" in the same font. To the right of the text is a red, stylized graphic element consisting of a series of curved, overlapping lines that resemble a musical staff or a sound wave. The entire logo is set against a background of white, wavy lines on a dark blue field.

ismir
2018

ISMIR 2018

**Proceedings of the 19th International Society
for Music Information Retrieval Conference**



**September 23 - 27, 2018
Paris, France**

**Edited by
Emilia Gómez, Xiao Hu, Eric Humphrey, Emmanouil Benetos**

ISMIR 2018 is organized by Télécom ParisTech, Institut de Recherche et Coordination Acoustique/Musique (IRCAM) and the International Society for Music Information Retrieval.

Website: <http://ismir2018.ircam.fr/>

ISMIR 2018 Logo & Proceedings Cover Designed by: Geoffroy Peeters

Edited by:

Emilia Gómez (Universitat Pompeu Fabra, Spain)

Xiao Hu (University of Hong Kong, Hong Kong)

Eric Humphrey (Spotify, USA)

Emmanouil Benetos (Queen Mary University of London, UK)

ISBN: 978-2-9540351-2-3

Title: Proceedings of the 19th International Society for Music Information Retrieval Conference

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.

© 2018 International Society for Music Information Retrieval



Organizers



Sponsors

Platinum Partners



Gold Partners



Silver Partners



Bronze Partners



WiMIR Sponsors



Conference Organizing Committee

General Chairs

Slim Essid, Télécom ParisTech, France
Geoffroy Peeters, IRCAM, France
Gaël Richard, Télécom ParisTech, France

Program Chairs

Emilia Gómez, Universitat Pompeu Fabra, Spain
Xiao Hu, University of Hong Kong, Hong Kong
Eric Humphrey, Spotify, USA
Emmanouil Benetos, Queen Mary University of London, UK

Tutorial Chairs

Hélène C. Crayencour, CNRS, France
Meinard Müller, International Audio Laboratories Erlangen, Germany

Sponsorship / Industry Partnership Chair

Blair Kaneshiro, Stanford University, USA

Women in MIR Chairs

Audrey Laplante, University of Montreal, Canada
Eva Zangerle, University of Innsbruck, Austria

Late Breaking News and Demos Chairs

Rachel Bittner, New York University, USA
Juan Pablo Bello, New York University, USA

Unconference Chair

Cynthia C.S. Liem, TU Delft, Netherlands

Music and Technology Chairs

Rebecca Fiebrink, Goldsmiths, University of London, UK
Wendy Mackay, INRIA Saclay, France
Frédéric Bevilacqua, IRCAM, France

"Meetup with industry" Session Chairs

Jimena Royo-Letelier, Deezer, France
Fabien Gouyon, Pandora, USA
Arshia Cont, Antescofo, France

Local Organizing Committee

Alice Cohen-Hadria, IRCAM, France
Hadrien Foroughmand, IRCAM, France
Magdalena Fuentes, Télécom ParisTech, France
Céline Jacques, IRCAM, France
Umut Simsekli, Télécom ParisTech, France
Andrea Vaglio, Deezer, France
Salimatou Yansane, Télécom ParisTech, France

Laurence Zelmar, Télécom ParisTech, France

Advisory Board

Emilia Gómez, Universitat Pompeu Fabra, Spain

Meinard Müller, International Audio Laboratories Erlangen, Germany

Eric J. Humphrey, Spotify, USA

Bob L. Sturm, KTH Royal Institute of Technology, Sweden

Geoffroy Peeters, IRCAM, France

Blair Kaneshiro, Stanford University, USA

Douglas Turnbull, Ithaca College, USA

Iris Yuping Ren, Utrecht University, Netherlands

Program Committee

Andreas Arzt, Johannes Kepler University
 Jean-Julien Aucouturier, CNRS/IRCAM
 Juan Pablo Bello, New York University
 Rachel Bittner, New York University
 Sebastian Böck, Austrian Research Institute for Artificial Intelligence
 Ching-Hua Chuan, University of Miami
 Sally Jo Cunningham, Waikato University
 Roger Dannenberg, Carnegie Mellon University
 Matthew Davies, INESC TEC
 Bas de Haas, Chordify
 Johanna Devaney, The Ohio State University
 Christian Dittmar, International Audio Laboratories Erlangen
 Simon Dixon, Queen Mary University of London
 J. Stephen Downie, University of Illinois
 Zhiyao Duan, University of Rochester
 Andreas Ehmann, Pandora
 Sebastian Ewert, Spotify
 George Fazekas, Queen Mary University of London
 Arthur Flexer, Austrian Research Institute for Artificial Intelligence (OFAI)
 José Fornari, NICS / COCEN / UNICAMP
 Ichiro Fujinaga, McGill University
 Masataka Goto, AIST
 Fabien Gouyon, Pandora
 Dorien Herremans, Singapore University of Technology and Design
 André Holzapfel, KTH
 Ozgur Izmirlı, Connecticut College
 Jyh-Shing Jang, CSIE Dept, National Taiwan University
 Peter Knees, TU Wien
 Audrey Laplante, Université de Montréal
 Jinha Lee, University of Washington
 Kyogu Lee, Seoul National University
 Cynthia Liem, Delft University of Technology
 Michael Mandel, Brooklyn College, CUNY
 Brian McFee, New York University
 Matt McVicar, Jukedeck
 Meinard Müller, International Audio Laboratories Erlangen
 Oriol Nieto, Pandora
 Kevin Page, University of Oxford
 Hélène C. Crayencour, CNRS
 Marcelo Queiroz, University of São Paulo
 Colin Raffel, Google Brain

David Rizo, University of Alicante
Justin Salamon, New York University
Markus Schedl, Johannes Kepler University Linz
Rodrigo Schramm, UFRGS
Jeffrey Scott, Gracenote, Inc.
Xavier Serra, Universitat Pompeu Fabra
Jordan Smith, IRCAM / Université Paris-Saclay
Mohamed Sordo, Pandora
Bob L. Sturm, KTH Royal Institute of Technology
Li Su, Academia Sinica
Douglas Turnbull, Ithaca College
George Tzanetakis, University of Victoria
Julián Urbano, Delft University of Technology
Peter van Kranenburg, Meertens Institute, Amsterdam
Gabriel Vigliensoni, DDMAL/CIRMMT/McGill University
Ye Wang, National University of Singapore
Frans Wiering, Utrecht University
Kazuyoshi Yoshii, Kyoto University
Eva Zangerle, University of Innsbruck

Reviewers

Jakob Abeßer
 Islah Ali-Maclachlan
 Manuel Anglada Tort
 Tom Arjannikov
 Claire Arthur
 David Bainbridge
 Stefan Balke
 Isabel Barbancho
 Michael Barone
 Dogac Basaran
 Christine Bauer
 Alejandro Bellogín
 Oded Ben-Tal
 Gilberto Bernardes
 Louis Bigo
 Juanjo Bosch
 Baris Bozkurt
 Paul Brossier
 Balamurali BT Nair
 Jorge Calvo-Zaragoza
 Carlos Cancino Chacon
 Estefania Cano
 Julio Carabias
 Rafael Caro
 Mark Cartwright
 Pritish Chadna
 Ning Chen
 Ching-Wei Chen
 Tian Cheng
 Srikanth Cherla
 Keunwoo Choi
 Chia-Hao Chung
 Andrea Cogliati
 Nathaniel Condit-Schultz
 Jonathan Cooper
 Emanuele Coviello
 Tim Crawford
 Michael Cuthbert
 Jiajie Dai
 Brecht De Man
 Reinier de Valk
 Andrew Demetriou
 Junqi Deng
 Chris Donahue
 Matthias Dorfer
 Jonathan Driedger
 Anders Elowsson

Jesse Engel
 Philippe Esling
 Jianyu Fan
 Zhe-Cheng Fan
 Mary Farbood
 Tiago Fernandes Tavares
 Andres Ferraro
 Ben Fields
 Flavio Figueireido
 Frederic Font
 Peter Foster
 Dominique Fourer
 Magdalena Fuentes
 Satoru Fukayama
 Martin Gasser
 Roman Gebhardt
 Mathieu Giraud
 Aggelos Gkiokas
 Nicolas Gold
 Rong Gong
 Ryan Groves
 David Grunberg
 Sankalp Gulati
 Chun Guo
 Chitrlekha Gupta
 Yoonchang Han
 Andrew Hankinson
 Yun Hao
 Ben Hayes
 Jason Hockman
 Yu-Hui Huang
 Chris Hubbles
 Karim M. Ibraheem
 Jose M. Inesta
 Charles Inskip
 Tristan Jehan
 Il-Young Jeong
 Maximos
 Kaliakatsos-Papakostas
 Blair Kaneshiro
 Andreas Katsiavalos
 Jaehun Kim
 Jong Wook Kim
 Minje Kim
 Katherine Kinnaird
 Rainer Kleinertz
 Hendrik Vincent Koops

Filip Korzeniowski
 Katerina Kosta
 Frank Kurth
 Pierre Laffitte
 Mathieu Lagrange
 Robin Laney
 Thibault Langlois
 Stefan Lattner
 Florence Leve
 Mark Levy
 David Lewis
 Shengchen Li
 Bochen Li
 Beici Liang
 Elad Liebman
 Yuan-Pin Lin
 Adam Liska
 Meijun Liu
 Patricio López-Serrano
 Vincent Lostanlen
 Simon Lui
 Athanasios Lykartsis
 Alexis McIntyre
 Akira Maezawa
 Elaine Mao
 Leandro Balby Marinho
 Matija Marolt
 Alan Marsden
 David Martins de Matos
 Agustin Martorell
 Matthias Mauch
 Matthew McCallum
 Daniel McEnnis
 Cory McKay
 Andrew McLeod
 Remi Mignot
 Saumitra Mishra
 Nicola Montecchio
 Josh Moore
 Hema Murthy
 Eita Nakamura
 Tomoyasu Nakano
 Juhan Nam
 Maria Navarro
 Eric Nichols
 Ken O'Hanlon
 Mitsunori Ogihara

Sergio Oramas
Nicola Orio
Jeongsoo Park
Antonio Pertusa
Pedro Pestana
Aggelos Pikrakis
Fatemeh Pishdadian
Jordi Pons
Alastair Porter
Matthew Prockup
Laurent Pugin
Elio Quinton
Zafar Rafii
Gang Ren
Iris Ren
Matthias Robine
Francisco
Rodriguez Algarra
Gerard Roma
Oriol Romani
Robert Rowe
Flávio Schiavoni
Alexander Schindler
Jan Schlueter

Hendrik Schreiber
David Sears
Prem Seetharaman
Sertan Şentürk
Francesco Setragno
Daniel Shanahan
Di Sheng
Zhengshan Shi
Joren Six
Olga Slizovskaia
Lloyd Smith
Carl Southall
Ajay Srinivasamurthy
Rebecca Stewart
Fabian-Robert Stoeter
Daniel Stoller
Florian Thalmann
Mi Tian
Marko Tkalcic
George Tourtellot
Chris Tralie
David Trevelyan
Timothy Tsai
Kosetsu Tsukuda

Andreu Vall
Leigh VanHandel
Gissel Velarde
Amruta Vidwans
Richard Vogl
Luke Waldner
Siying Wang
Hsin-Min Wang
Cheng-i Wang
David Weigl
Christof Weiß
Thomas Wilmering
Daniel Wolff
Chih-Wei Wu
Anna Xambó
Karthik Yadati
Yujia Yan
Luwei Yang
Asteris Zacharakis
Frank Zalkow
Yichi Zhang
Kejun Zhang

Contents

Preface	xxi
Keynote Talks	xxix
Drawing sounds: Fourier, Koenig, and Scott <i>Patrick Flandrin</i>	xxxi
Using Data and Machine Learning to Support Human Musical Practices <i>Rebecca Fiebrink</i>	xxxii
Tutorials	xxxv
Open Source and Reproducible MIR Research <i>Brian McFee and Thor Kell</i>	xxxvii
Computational Approaches for Analysis of Non-Western Music Traditions <i>Xavier Serra, Martin Clayton, Barış Bozkurt</i>	xxxix
Statistical Analysis of Results in Music Information Retrieval: Why and How <i>Julián Urbano, Arthur Flexer</i>	xli
Music Separation with DNNs: Making It Work <i>Antoine Liutkus, Fabian-Robert Stöter</i>	xlili
Deep Learning for MIR <i>Alexander Schindler, Thomas Lidy, Sebastian Böck</i>	xlvi
Fundamental Frequency Estimation in Music <i>Rachel Bittner, Alain de Cheveigné, Johanna Devaney</i>	xlvi
Optical Music Recognition for Dummies <i>Jorge Calvo-Zaragoza, Jan Hajič jr., Alexander Pacha, Ichiro Fujinaga</i>	xlvi
Overview and New Challenges of Music Recommendation Research in 2018 <i>Markus Schedl, Peter Knees, Fabien Gouyon</i>	xlix
Session A: Musical objects	1
A Confidence Measure For Key Labelling <i>Roman B. Gebhardt, Michael Stein, Athanasios Lykartsis</i>	3
Improved Chord Recognition by Combining Duration and Harmonic Language Models <i>Filip Korzeniowski, Gerhard Widmer</i>	10
Using Musical Relationships Between Chord Labels in Automatic Chord Extraction Tasks <i>Tristan Carsault, Jerome Nika, Philippe Esling</i>	18
A Predictive Model for Music based on Learned Interval Representations <i>Stefan Lattner, Maarten Grachten, Gerhard Widmer</i>	26
An End-to-end Framework for Audio-to-Score Music Transcription on Monophonic Excerpts <i>Miguel A. Román, Antonio Pertusa, Jorge Calvo-Zaragoza</i>	34
Evaluating Automatic Polyphonic Music Transcription <i>Andrew McLeod, Mark Steedman</i>	42
Onsets and Frames: Dual-Objective Piano Transcription <i>Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, Douglas Eck</i>	50
Player Vs Transcriber: A Game Approach To Data Manipulation For Automatic Drum Transcription <i>Carl Southall, Ryan Stables, Jason Hockman</i>	58

Towards Full-Pipeline Handwritten OMR with Musical Symbol Detection by U-Nets	
<i>Jan Hajič jr., Matthias Dorfer, Gerhard Widmer, Pavel Pecina</i>	225
Searching Page-Images of Early Music Scanned with OMR: A Scalable Solution Using Minimal Absent Words	
<i>Tim Crawford, Golnaz Badkobeh, David Lewis</i>	233
Optical Music Recognition in Mensural Notation with Region-based Convolutional Neural Networks	
<i>Alexander Pacha, Jorge Calvo-Zaragoza</i>	240
Camera-PrIMuS: Neural End-to-End Optical Music Recognition on Realistic Monophonic Scores	
<i>Jorge Calvo-Zaragoza, David Rizo</i>	248
Document Analysis of Music Score Images with Selectional Auto-Encoders	
<i>Francisco Castellanos, Jorge Calvo-Zaragoza, Gabriel Vigliensoni, Ichiro Fujinaga</i>	256
Genre-Agnostic Key Classification With Convolutional Neural Networks	
<i>Filip Korzeniowski, Gerhard Widmer</i>	264
Deep Watershed Detector for Music Object Recognition	
<i>Lukas Tuggener, Ismail Elezi, Jürgen Schmidhuber, Thilo Stadelmann</i>	271

Session C: Source separation, symbolic, emotion 279

Deep Neural Networks with Voice Entry Estimation Heuristics for Voice Separation in Symbolic Music Representations	
<i>Reinier de Valk, Tillman Weyde</i>	281
Music Source Separation Using Stacked Hourglass Networks	
<i>Sungheon Park, Taehoon Kim, Kyogu Lee, Nojun Kwak</i>	289
The Northwestern University Source Separation Library	
<i>Ethan Manilow, Prem Seetharaman, Bryan Pardo</i>	297
Improving Bass Saliency Estimation using Transfer Learning and Label Propagation	
<i>Jakob Abeßer, Stefan Balke, Meinard Müller</i>	306
Improving Peak-picking Using Multiple Time-step Loss Functions	
<i>Carl Southall, Ryan Stables, Jason Hockman</i>	313
Zero-Mean Convolutions for Level-Invariant Singing Voice Detection	
<i>Jan Schlüter, Bernhard Lehner</i>	321
Music Generation and Transformation with Moment Matching-Scattering Inverse Networks	
<i>Mathieu Andreux, Stéphane Mallat</i>	327
Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation	
<i>Daniel Stoller, Sebastian Ewert, Simon Dixon</i>	334
SE and SNL diagrams: Flexible data structures for MIR	
<i>Melissa R. McGuirl, Katherine M. Kinnaird, Claire Savard, Erin H. Bugbee</i>	341
JSYMBOLIC 2.2: Extracting Features from Symbolic Music for use in Musicological and MIR Research	
<i>Cory McKay, Julie Cumming, Ichiro Fujinaga</i>	348
Relevance of Musical Features for Cadence Detection	
<i>Louis Bigo, Laurent Feisthauer, Mathieu Giraud, Florence Levé</i>	355
On the Relationships between Music-induced Emotion and Physiological Signals	
<i>Xiao Hu, Fanjie Li, Jeremy T. D. Ng</i>	362
Music Mood Detection Based on Audio and Lyrics with Deep Neural Net	
<i>Rémi Delbouys, Romain Hennequin, Francesco Piccoli, Jimena Royo-Letelier, Manuel Moussallam</i>	370
Identifying Emotions in Opera Singing: Implications of Adverse Acoustic Conditions	
<i>Emilia Parada-Cabaleiro, Maximilian Schmitt, Anton Batliner, Simone Hantke, Giovanni Costantini, Klaus Scherer, Bjoern Schuller</i>	376

Musical Texture and Expressivity Features for Music Emotion Recognition

Renato Panda, Ricardo Malheiro, Rui Pedro Paiva 383

Shared Generative Representation of Auditory Concepts and EEG to Reconstruct Perceived and Imagined Music

André Ofner, Sebastian Stober 392

Exploring Musical Relations Using Association Rule Networks

Renan de Padua, Verônica Oliveira de Carvalho, Solange Rezende, Diego Furtado Silva 400

Session D: Corpora and voice**407**

A Crowdsourced Experiment for Tempo Estimation of Electronic Dance Music

Hendrik Schreiber, Meinard Müller 409

Computational Corpus Analysis: A Case Study on Jazz Solos

Christof Weiss, Stefan Balke, Jakob Abeßer, Meinard Müller 416

Controlled Vocabularies for Music Metadata

Pasquale Lisena, Konstantin Todorov, Cécile Cecconi, Françoise Leresche, Isabelle Canno, Frédéric Puyrenier, Martine Voisin, Thierry Le Meur, Raphaël Troncy 424

DALI: A Large Dataset of Synchronized Audio, Lyrics and notes, Automatically Created using Teacher-student Machine Learning Paradigm.

Gabriel Meseguer-Brocal, Alice Cohen-Hadria, Geoffroy Peeters 431

OpenMIC-2018: An Open Data-set for Multiple Instrument Recognition

Eric Humphrey, Simon Durand, Brian McFee 438

From Labeled to Unlabeled Data – On the Data Challenge in Automatic Drum Transcription

Chih-Wei Wu, Alexander Lerch 445

GuitarSet: A Dataset for Guitar Transcription

Qingyang Xi, Rachel Bittner, Johan Pauwels, Xuzhou Ye, Juan Pablo Bello 453

Musical-Linguistic Annotations of Il Lauro Secco

Emilia Parada-Cabaleiro, Maximilian Schmitt, Anton Batliner, Bjoern Schuller 461

VocalSet: A Singing Voice Dataset

Julia Wilkins, Prem Seetharaman, Alison Wahl, Bryan Pardo 468

The NES Music Database: A multi-instrumental dataset with expressive performance attributes

Chris Donahue, Huanru Henry Mao, Julian McAuley 475

Audio-Aligned Jazz Harmony Dataset for Automatic Chord Transcription and Corpus-based Research

Vsevolod Eremenko, Emir Demirel, Baris Bozkurt, Xavier Serra 483

Methodologies for Creating Symbolic Corpora of Western Music Before 1600

Julie Cumming, Cory McKay, Jonathan Stuchbery, Ichiro Fujinaga 491

Precision of Sung Notes in Carnatic Music

Venkata Viraraghavan, Rangarajan Aravind, Hema Murthy 499

Revisiting Singing Voice Detection: A quantitative review and the future outlook

Kyungyun Lee, Keunwoo Choi, Juhan Nam 506

Vocals in Music Matter: the Relevance of Vocals in the Minds of Listeners

Andrew Demetriou, Andreas Jansson, Aparna Kumar, Rachel Bittner 514

Vocal Melody Extraction with Semantic Segmentation and Audio-symbolic Domain Transfer Learning

Wei Tsung Lu, Li Su 521

Empirically Weighting the Importance of Decision Factors for Singing Preference

Michael Barone, Karim Ibrahim, Chitrlekha Gupta, Ye Wang 529

Session E: Timbre, tagging, similarity, patterns and alignment

537

Analysis by Classification: A Comparative Study of Annotated and Algorithmically Extracted Patterns in Symbolic Music Data

Iris Yuping Ren, Anja Volk, Wouter Swierstra, Remco Veltkamp 539

Generalized Skipgrams for Pattern Discovery in Polyphonic Streams

Christoph Finkensiep, Markus Neuwirth, Martin Rohrmeier 547

Comparison of Audio Features for Recognition of Western and Ethnic Instruments in Polyphonic Mixtures

Igor Vatolkin, Günter Rudolph 554

Instrudiver: A Music Visualization System Based on Automatically Recognized Instrumentation

Takumi Takahashi, Satoru Fukayama, Masataka Goto 561

Instrument Activity Detection in Polyphonic Music using Deep Neural Networks

Siddharth Gururani, Cameron Summers, Alexander Lerch 569

Jazz Solo Instrument Classification with Convolutional Neural Networks, Source Separation, and Transfer Learning

Juan S. Gómez, Jakob Abeßer, Estefanía Cano 577

Aligned Sub-Hierarchies: A Structure-based Approach to the Cover Song Task

Katherine M. Kinnaird 585

Audio-to-Score Alignment using Transposition-invariant Features

Andreas Arzt, Stefan Lattner 592

Semi-supervised Lyrics and Solo-singing Alignment

Chitraksha Gupta, Rong Tong, Haizhou Li, Ye Wang 600

Concert Stitch: Organization and Synchronization of Crowd Sourced Recordings

Vinod Subramanian, Alexander Lerch 608

A Data-driven Approach to Mid-level Perceptual Musical Feature Modeling

Anna Aljanaki, Mohammad Soleymani 615

Disambiguating Music Artists at Scale with Audio Metric Learning

Jimena Royo-Letelier, Romain Hennequin, Viet-Anh Tran, Manuel Moussallam 622

Driftin' Down the Scale: Dynamic Time Warping in the Presence of Pitch Drift and Transpositions

Simon Waloschek, Aristotelis Hadjakos 630

End-to-end Learning for Music Audio Tagging at Scale

Jordi Pons, Oriol Nieto, Matthew Prockup, Erik M. Schmidt, Andreas F. Ehmann, Xavier Serra 637

Audio Based Disambiguation of Music Genre Tags

Romain Hennequin, Jimena Royo-Letelier, Manuel Moussallam 645

Learning Domain-Adaptive Latent Representations of Music Signals Using Variational Autoencoders

Yin-Jyun Luo, Li Su 653

Learning Interval Representations from Polyphonic Music Sequences

Stefan Lattner, Maarten Grachten, Gerhard Widmer 661

Session F: Machine and human learning of music

669

Influences on the Social Practices Surrounding Commercial Music Services: A Model for Rich Interactions

Louis Spinelli, Josephine Lau, Liz Pritchard, Jin Ha Lee 671

Investigating Cross-Country Relationship between Users' Social Ties and Music Mainstreamness

Christine Bauer, Markus Schedl 678

Listener Anonymizer: Camouflaging Play Logs to Preserve User's Demographic Anonymity

Kosetsu Tsukuda, Satoru Fukayama, Masataka Goto 687

On the Impact of Music on Decision Making in Cooperative Tasks	
<i>Elad Liebman, Corey N. White, Peter Stone</i>	695
VenueRank: Identifying Venues that Contribute to Artist Popularity	
<i>Emmanouil Krasanakis, Emmanouil Schinas, Symeon Papadopoulos, Yiannis Kompatsiaris, Pericles Mitkas</i>	702
The Many Faces of Users: Modeling Musical Preference	
<i>Eva Zangerle, Martin Pichl</i>	709
Representation Learning of Music Using Artist Labels	
<i>Jiyoung Park, Jongpil Lee, Jangyeon Park, Jung-Woo Ha, Juhan Nam</i>	717
StructureNet: Inducing Structure in Generated Melodies	
<i>Gabriele Medeot, Srikanth Cherla, Katerina Kosta, Matt McVicar, Samer Abdallah, Marco Selvi, Ed Newton-Rex, Kevin Webster</i>	725
Summarizing and Comparing Music Data and Its Application on Cover Song Identification	
<i>Diego Furtado Silva, Felipe Falcão, Nazareno Andrade</i>	732
Transferring the Style of Homophonic Music Using Recurrent Neural Networks and Autoregressive Model	
<i>Wei Tsung Lu, Li Su</i>	740
MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer	
<i>Gino Brunner, Andres Konrad, Yuyi Wang, Roger Wattenhofer</i>	747
Understanding a Deep Machine Listening Model Through Feature Inversion	
<i>Saumitra Mishra, Bob L. Sturm, Simon Dixon</i>	755
Comparing RNN Parameters for Melodic Similarity	
<i>Tian Cheng, Satoru Fukayama, Masataka Goto</i>	763
Visualization of Audio Data Using Stacked Graphs	
<i>Mathieu Lagrange, Mathias Rossignol, Grégoire Lafay</i>	771
Two Web Applications for Exploring Melodic Patterns in Jazz Solos	
<i>Klaus Frieler, Frank Höger, Martin Pfeiderer, Simon Dixon</i>	777
Learning to Listen, Read, and Follow: Score Following as a Reinforcement Learning Game	
<i>Matthias Dorfer, Florian Henkel, Gerhard Widmer</i>	784
Matrix Co-Factorization for Cold-Start Recommendation	
<i>Olivier Gouvert, Thomas Oberlin, Cédric Févotte</i>	792

Preface

Welcome to ISMIR 2018!

Dear ISMIR 2018 Attendees,

Sixteen years after its first venue in Paris in 2002, it is our great pleasure to welcome you back in the City of Lights for the 19th International Society for Music Information Retrieval Conference (ISMIR). This year's conference is organized by Télécom ParisTech and IRCAM.

ISMIR is the world's leading research forum on processing, searching, organizing and accessing music-related data. The success of ISMIR is continuously growing, as evidenced by the regular increase of its audience and the variety of its main program and satellite events. We have tried our best, with the invaluable help of our colleagues of the organizing committees and ISMIR board to maintain the tradition and spirit of previous editions while experimenting some new ideas for the scientific program, presentations and surrounding events.

We are excited to present this year's program. A total of 235 complete and well-formatted papers entered the review process. Special care was taken to assemble an experienced and interdisciplinary review panel comprising people from many different academic and industrial institutions worldwide. As in previous years, reviews were double-blind (i.e., both the authors and the reviewers were anonymous) with a two-tier review model involving a pool of 269 reviewers, including a program committee (PC) of 60 members. Each paper was assigned to a PC member and three reviewers. The reviewers' assignments were based on topic preferences, bidding on papers, and PC member assignments. Following the review phase, PC members and reviewers entered a discussion phase aiming to homogenize acceptance versus rejection decisions. Handling four submissions on average, each PC member was asked to adopt an active role in the review process by partaking in an intensive discussion phase with the other reviewers and providing a detailed meta-review. Final acceptance decisions were based on 940 reviews and meta-reviews. Of the 235 reviewed papers, 104 were accepted, resulting in an acceptance rate of 44.2%. The table shown on the next page summarizes the ISMIR publication statistics over the history of the conference. A very special thanks goes to our technical program chairs for their extensive efforts to ensure a fair and efficient paper selection.

For the first time at ISMIR, all papers will be presented both orally and as posters.

A number of other changes in the main program have also been implemented, in particular to have a specific and novel "Meetup with industry" event, which is organized at *Station F*, a large and emblematic business incubator for startups; and to host a dedicated exhibition of selected installations of Interactive Machine-Learning for Music.

Along with the main ISMIR program, we are happy to host a tutorial day and a number of satellite events including hackathons, workshops and web seminars.

We would like to thank the whole organizing committee for their dedication and invaluable help in setting up all the details of the conference. We are particularly grateful to all our

sponsors who provided a considerable support to the conference - many thanks to them! And a very warm thanks to Blair Kaneshiro for her sustained efforts and actions towards our sponsors.

In short, welcome to Paris and have a fantastic ISMIR2018!

Geoffroy Peeters, Slim Essid and Gaël Richard, General Chairs

Year	Location	Oral	Poster	Total Papers	Total Pages	Total Authors	Unique Authors	Pages/ Paper	Authors/ Paper	Unique Authors/ Paper
2000	Plymouth	19	16	35	155	68	63	4.4	1.9	1.8
2001	Indiana	25	16	41	222	100	86	5.4	2.4	2.1
2002	Paris	35	22	57	300	129	117	5.3	2.3	2.1
2003	Baltimore	26	24	50	209	132	111	4.2	2.6	2.2
2004	Barcelona	61	44	105	582	252	214	5.5	2.4	2
2005	London	57	57	114	697	316	233	6.1	2.8	2
2006	Victoria	59	36	95	397	246	198	4.2	2.6	2.1
2007	Vienna	62	65	127	486	361	267	3.8	2.8	2.1
2008	Philadelphia	24	105	105	630	296	253	6	2.8	2.4
2009	Kobe	38	85	123	729	375	292	5.9	3	2.4
2010	Utrecht	24	86	110	656	314	263	6	2	2.4
2011	Miami	36	97	133	792	395	322	6	3	2.4
2012	Porto	36	65	101	606	324	264	6	3.2	2.6
2013	Curitiba	31	67	98	587	395	236	5.9	3	2.4
2014	Taipei	33	73	106	635	343	271	6	3.2	2.6
2015	Málaga	24	90	114	792	370	296	7	3.2	2.6
2016	New York	25	88	113	781	341	270	6.9	3.0	2.4
2017	Suzhou	24	73	97	716	324	248	7.4	3.3	2.6
2018	Paris	104		104	786	337	265	7.5	3.2	2.5

Tutorials

Eight tutorials will take place on Sunday, September 23rd 2018. There will be four parallel tutorials in the morning, and another four parallel tutorials in the afternoon. All tutorials will take place at Télécom ParisTech.

Morning Sessions:

Tutorial 1: Open Source and Reproducible MIR Research

Brian McFee, New York University

Thor Kell, Spotify

Tutorial 2: Computational Approaches for Analysis of Non-Western Music Traditions

Xavier Serra, Universitat Pompeu Fabra

Martin Clayton, Durham University

Barış Bozkurt, Universitat Pompeu Fabra

Tutorial 3: Statistical Analysis of Results in Music Information Retrieval: Why and How

Julián Urbano, Delft University of Technology

Arthur Flexer, Austrian Research Institute for Artificial Intelligence

Tutorial 4: Music Separation with DNNs: Making It Work

Antoine Liutkus, Inria

Fabian-Robert Stöter, Inria

Afternoon Sessions:

Tutorial 5: Deep Learning for MIR

Alexander Schindler, Austrian Institute of Technology

Thomas Lidy, Musimap

Sebastian Böck, Austrian Research Institute for Artificial Intelligence

Tutorial 6: Fundamental Frequency Estimation in Music

Rachel Bittner, Spotify

Alain de Chevigne, CNRS

Johanna Devaney, City University of New York

Tutorial 7: Optical Music Recognition for Dummies

Jorge Calvo-Zaragoza, McGill University

Jan Hajič jr., Charles University

Alexander Pacha, TU Wien

Ichiro Fujinaga, McGill University

Tutorial 8: Overview and New Challenges of Music Recommendation Research in 2018

Markus Schedl, Johannes Kepler University Linz

Peter Knees, TU Wien

Fabien Gouyon, Pandora

Keynote Speakers

We are honored to have two distinguished keynote speakers:

Drawing sounds: Fourier, K  nig, and Scott

Patrick Flandrin

Physics Department

  cole Normale Sup  rieure de Lyon

Using Data and Machine Learning to Support Human Musical Practices

Rebecca Fiebrink

Department of Computing

Goldsmiths, University of London

WiMIR Annual Meeting

Women in MIR (WiMIR) is a group of people in the MIR community dedicated to promoting the role of, and increasing opportunities for, women in the field. We meet to network, share information, and discuss in an informal setting the goal of building a community that supports women – and more broadly, diversity – in the field of MIR.

WiMIR has held annual meetings at the ISMIR conference since 2012, garnering a high turnout of both female and male attendees. Since 2016, WiMIR has also organized a mentoring program connecting female students, postdocs, and early-stage researchers to more senior females and male allies in the field.

At this year's ISMIR, we will have a special WiMIR cocktail during Tuesday, September 25th lunch followed by a special WiMIR session.

Unconference Evening

As in previous years, we will have a special “Unconference” in which participants break up into smaller groups to discuss Music-IR issues of particular interest. This is an informal and informative opportunity to get to know peers and colleagues from around the world. This year, the “Unconference” will last a whole evening, on Tuesday, September 25th at T  l  com ParisTech.

Late-Breaking/Demo session & Unconference Debriefing

Thursday morning is dedicated to late-breaking papers and Music-IR system demonstrations. Abstracts for these presentations are available online. After this session, the results of the discussions held during the unconference will be debriefed in a plenary session.

Meetup with Industry afternoon

The annual ISMIR (International Society for Music Information Retrieval) Conference is the world's leading R&D forum for music tech since 2000. For the first time, the 2018 edition in Paris will include a Meetup with Industry, organized by Antescofo with support from Deezer and Pandora. This will be a unique networking platform, and a perfect opportunity to:

- Discover and shape the future of music technology
- Connect with and recruit top talent from the MIR community
- Meet world-class experts and explore the current state-of-the-art in music tech. applications
- Connect with and learn from leading music tech. companies
- Discover music startups leveraging latest technologies in a variety of music applications
- Exchange with other players and startups in the music industry
- Showcase and market latest products and services

This will be a half-day event with networking activities and Industry talks, taking place on Thursday September 27, 2018, from 2PM to 6PM, at Station F, the biggest startup campus in the world, in the heart of Paris.

Satellite Events

This year, six satellite workshops will take place around ISMIR in various places (IRCAM, CNAM and Télécom ParisTech).

- **Hacking Audio and Music Research (HAMR)** will take place at the Deezer headquarters in Paris immediately before ISMIR on 21-22/09/2018
- **1st Workshop on Music Reading Systems (WoRMS)** will be held at the CNAM Thursday 2018/09/20
- **Workshop on “The New Shape of Audio Branding”** will be held at IRCAM Thursday 2018/09/20
- **5th International Conference on Digital Libraries for Musicology** will be held at IRCAM Friday 2018/09/28
- **WiMIR 1st Annual Workshop** will be held at Télécom ParisTech Friday 2018/09/28
- **Radio 2.0 Webinar Series: From music data to value creation** will be held at Vivendi headquarters Friday 2018/09/28

Social Events

In addition to the academic focus of ISMIR, we are offering a number of unique social events. The social program provides participants with an opportunity to relax after meetings, to experience Paris, and to network with other ISMIR participants. The social program includes:

Welcome Reception

The welcome reception will happen at the Balajo on Sunday September 23rd. The Balajo is a historical dancing-hall in the Bastille neighborhood opened in 1936. It has welcomed many famous artists such as Mistinguett, Marlene Dietrich and Django Reinhardt. There, we will have a private concert of Gipsy Jazz music with *hors d'oeuvres* and drinks.

ISMIR Banquet and Jam Session

The banquet will happen Wednesday September 26th as a Seine river cruise buffet(with views over all the most famous Paris buildings). The cruise dock will be at the foot of the Eiffel tower. The cruise will be concluded with the first ever ISMIR Jam Session on a boat.

Host City

Paris is the capital and most populous city of France. Situated on the river Seine in the north of the country, it is in the center of the Île-de-France region, also known as the *région parisienne*, "Paris Region". It is home for the most visited art museum in the world, the *Louvre*, as well as the *Musée d'Orsay*, noted for its collection of French Impressionist art, and the *Musée National d'Art Moderne*, a museum of modern and contemporary art. The notable architectural landmarks of Paris include *Notre Dame Cathedral* (12th century); the *Sainte-Chapelle* (13th century); the Eiffel Tower (1889); and the Basilica of *Sacré-Cœur* on Montmartre (1914). In 2014 Paris received 22.4 million visitors, making it one of the world's top tourist destinations. Paris is also known for its fashion, particularly the twice-yearly Paris Fashion Week, and for its haute cuisine, and three-star restaurants.

Acknowledgments

We are very proud to present to you the proceedings of ISMIR 2018. The conference program was made possible thanks to the hard work of many people including the members of the organizing committee, the many reviewers and meta-reviewers from the program committee.

Special thanks go to this year's sponsors:

Platinum Partners:

- CNRS
- Deezer
- Gracenote
- Native Instruments
- Pandora

- Shazam
- Smule
- Spotify
- Yousician

Gold Partners:

- iZotope
- Jukedek
- Steinberg

Silver Partners:

- ACRCLOUD
- Adobe
- FeedForward AI

Bronze Partners:

- Google

We also gratefully acknowledge the sponsors who contributed specifically to Women in Music Information Retrieval (WiMIR) initiatives and Student Travel awards:

- Smule
- Spotify
- CCRMA
- Gracenote
- iZotope
- Native Instruments
- Shazam
- Steinberg

Last but not least, the ISMIR program is possible only thanks to the excellent contributions of our community in response to our Call for Participation. The biggest acknowledgment goes to you, the authors, reviewers, researchers, and participants of this conference. We wish you a productive and memorable stay in Paris.

Emilia Gómez, Universitat Pompeu Fabra, Spain

Xiao Hu, University of Hong Kong, Hong Kong

Eric Humphrey, Spotify, USA

Emmanouil Benetos, Queen Mary University of London, UK

Program Chairs

Slim ESSID, Télécom ParisTech, France

Geoffroy Peeters, IRCAM, France

Gaël Richard, Télécom ParisTech, France

General Chairs

Keynote Talks

Keynote Talk 1

Drawing sounds: Fourier, Kœnig, and Scott

Patrick Flandrin

Physics Department

École Normale Supérieure de Lyon

Abstract

In his seminal work, first published in 1811, Joseph Fourier was primarily concerned with the building of an analytic theory of heat, but it was realized soon after that the expansion methods he developed for this purpose had potential applications far beyond, in physics as well as in mathematics. This was in particular the case for sounds with, in the middle of the XIXth century, a quest for graphical representations in time and/or in frequency, thanks to dedicated devices. One apparatus designer, Rudolph Kœnig, was particularly instrumental in such studies, in the two domains. On the one hand, he built an actual Fourier analyzer based on resonators, manometric flames and mirrors. On the other hand, he collaborated with Édouard-Léon Scott de Martinville on his project of a « phonautograph » that, in 1857, permitted the first ever recording of a human voice in the form of a graph on a paper sheet. Whereas Scott's objective was transcription and not restitution, some of his sound graphs have been recently scanned and digitized, allowing us to actually hear him singing.

Beyond celebrating this year the 250th birthday of Fourier and reviving the forgotten memory of Scott, intertwining those complementary approaches via Kœnig is believed to offer a way of revisiting issues such as the physical significance of Fourier modes, or the questionable necessity of their use as features in modern recognition systems.

Biography

Patrick Flandrin received the engineer degree from ICPI Lyon, France, in 1978, and the Doct.-Ing. and Docteur d'État degrees from INP Grenoble, France, in 1982 and 1987, respectively. He joined CNRS in 1982, where he is currently Research Director. Since 1991, he has been with the Signals, Systems and Physics Group, within the Physics Department at ENS de Lyon, France. He is currently President of GRETSI, the French Association for Signal and Image Processing. His research interests include mainly nonstationary signal processing (with emphasis on time-frequency and time-scale methods), scaling stochastic processes and complex systems. He authored two monographs in those areas, the most recent one being *Explorations in Time-Frequency Analysis* (Cambridge University Press, 2018). Dr. Flandrin was awarded the Philip Morris Scientific Prize in Mathematics (1991), the SPIE Wavelet Pioneer Award (2001), the Prix Michel Monpetit from the French Academy of Sciences (2001), the Silver Medal from CNRS (2010), and the Technical Achievement Award from the IEEE Signal Processing Society (2017). Past Distinguished Lecturer of the IEEE Signal Processing Society (2010-2011), he is a Fellow of the IEEE (2002) and of EURASIP (2009), and he has been elected member of the French Academy of Sciences in 2010.

Keynote Talk 2

Using Data and Machine Learning to Support Human Musical Practices

Rebecca Fiebrink

Department of Computing
Goldsmiths, University of London

Abstract

It's 2018, and machine learning seems to suddenly be everywhere: playing Go, driving cars, serving us targeted advertising. Machine learning can compose new folk tunes and synthesise new sounds. What does this mean for those of us who compose or perform new music, or who create new interactions with sound? What does our future hold, besides sitting at home all day listening to algorithmically generated music after robots take our jobs?

In this talk, I'll invite you to consider what I believe to be a more important and interesting question: How can we instead use machine learning to better support human creative activities? I'll describe some highlights from my own research and others', including using machine learning and related techniques to support new approaches to musical instrument design, to enable latency-free networked musical performance and personalised audience experiences, and to enable a much broader range of people—from software developers to children to music therapists—to build new musical and sonic interactions. I'll discuss how machine learning can support human creative practices, for instance by enabling faster prototyping and exploration of new technologies (including by non-programmers), by supporting greater embodied engagement in design, and by changing the ways that creators are able to think about the design process and about themselves. I'll discuss how these findings inform new ways of thinking about what machine learning is good for, how to make more useful and usable creative machine learning tools, how to teach creative practitioners about machine learning, and what the future of human creative practice might look like.

Biography

Dr. Rebecca Fiebrink is a Senior Lecturer at Goldsmiths, University of London. Her research focuses on designing new ways for humans to interact with computers in creative practice. As both a computer scientist and a musician, much of her work focuses on applications of machine learning to music: for example, how can machine learning algorithms help people to create new musical instruments and interactions? How does machine learning change the type of musical systems that can be created, the creative relationships between people and technology, and the set of people who can create new technologies? Much of Fiebrink's work is also driven by a belief in the importance of inclusion, participation, and accessibility. She works frequently with human-centred and participatory design processes, and she is currently working on projects related to creating new accessible technologies with people with disabilities, designing inclusive machine learning curricula and tools, and applying participatory design methodologies in the digital humanities.

Fiebrink is the developer of the Wekinator, open-source software for real-time interactive machine learning whose current version has been downloaded over 10,000 times. She is the creator of a MOOC titled "Machine Learning for Artists and Musicians," which launched in

2016 on the Kadenze platform. She was previously an Assistant Professor at Princeton University, where she co-directed the Princeton Laptop Orchestra. She has worked with companies including Microsoft Research, Sun Microsystems Research Labs, Imagine Research, and Smule. She has performed with a variety of musical ensembles, including as a laptopist in Sideband and Squirrel in the Mirror, the principal flutist in the Timmins Symphony Orchestra, and the keyboardist in the University of Washington computer science rock band "The Parody Bits." She holds a PhD in Computer Science from Princeton University.

Tutorials

Tutorial 1

Open Source and Reproducible MIR Research

Brian McFee and Thor Kell

Abstract

The goal of this tutorial is to provide hands-on, practical training for MIR researchers to learn modern tools for improving the quality of their research software.

We will provide a project template repository, which we will use as scaffolding to demonstrate practices and techniques including version control, continuous integration, automatic documentation, environments & software dependency tracking, and packaging & distribution. While many of the techniques we will cover are independent of language or programming environment, the practices are best demonstrated by example, and we expect Python to be the most accessible and generally useful language for the expected attendees.

The tutorial will begin with a brief introduction to version control with git and GitHub. We will only cover the basics: creating an account, cloning, pushing, and pulling. We will walk attendees through the process of making their own copy of the repository, verifying that tests and documentation work, implementing some simple functionality, and working with automatic testing. The goal of this exercise is to expose attendees to modern development practices, so that their software is continuously tested during development. Attendees will then learn how to create documentation with Sphinx and automate the process with the ReadTheDocs service. We will teach standard documentation style, and provide an exercise in which attendees write documentation for a previously undocumented function. We will also discuss how to write README files and how to make a package easy for others to use. We note that many of the tools we will demonstrate are provided by web services (Travis, GitHub, etc.), which have freely available counterparts that can run on local servers (Jenkins, 8 GitLab, 9 etc.). We will provide pointers to these alternative implementations, but for ease of exposition and simplicity, we will stick to the web-based services for the tutorial.

Finally, we will close with a tour of the Python/MIR ecosystem. This part of the tutorial will give an overview of the available packages commonly used in MIR research, such as the scipy stack, pandas, librosa, mir eval, and jupyter. We will spend some extra time on Jupyter, including plotting and sonification, as well as discussing how to save and iterate on notebooks.

Brian McFee is a Moore-Sloan Fellow at New York University's Center for Data Science and Music and Audio Research Lab (MARL). He received the B.S. degree (2003) in Computer Science from the University of California, Santa Cruz, and M.S. (2008) and Ph.D. (2012) degrees in Computer Science and Engineering from the University of California, San Diego. His work touches on various topics at the intersection of machine learning, information retrieval, and audio analysis. He is an active open source software developer, and the principal maintainer of the librosa package for music and audio signal processing. His favorite genre is "chip-tune", and he likes dogs.

Thor Kell earned his B.Sc in Computer Science & Music from the University of Victoria in 2011, and an M.A. in Music Technology from McGill in 2015. He has worked for SoundCloud, The Echo Nest, and is currently an engineer at Spotify. His interests include algorithmic music creation as well as track ordering and automatic mixing for DJ sets. His favorite genre is “post”, and he likes cats.

Tutorial 2

Computational Approaches for Analysis of Non-Western Music Traditions

Xavier Serra, Martin Clayton, and Barış Bozkurt

Abstract

The main goal of this tutorial is to present an overview for computational analysis applied on non-Western music traditions and within this context, also discuss the role of musicology perspective in Music Information Retrieval. The target audience is researchers and students interested in MIR and computational musicology. The tutorial comprises of three parts.

The first two parts are dedicated to critical overview of recent studies with engineering and musicology perspective. We will start by presenting some of the relevant problems and challenges for analysis of non-Western music traditions that have been studied from an MIR perspective. In the second part we discuss some current and potential challenges posed by musicological research in diverse musical genres, including in rhythmic analysis.

The last part will present resources created during the CompMusic project, that are openly available to the community. It will include demonstrations for accessing non-Western music data and processing these data (focusing on intonation and rhythm analysis) with publicly available tools. We will consider research corpora from various music traditions: Hindustani (North India), Carnatic (South India), Turkish-makam (Turkey), Arab-Andalusian (Maghreb), and Beijing Opera (China). While we will focus on a few culture-specific MIR tasks for our demonstrations, we will also discuss open research problems that can be studied using these datasets. This session would serve as a quick start for students and researchers without prior experience in analysis of non-Western music and will provide them a good entry point for further investigation.

Xavier Serra is a Professor of the Department of Information and Communication Technologies and Director of the Music Technology Group at the Universitat Pompeu Fabra in Barcelona. After a multidisciplinary academic education he obtained a PhD in Computer Music from Stanford University in 1989 with a dissertation on the spectral processing of musical sounds that is considered a key reference in the field. His research interests cover the analysis, description and synthesis of sound and music signals, with a balance between basic and applied research and approaches from both scientific/technological and humanistic/artistic disciplines. Dr. Serra is very active in promoting initiatives in the field of Sound and Music Computing at the local and international levels, being involved in the editorial board of a number of journals and conferences and giving lectures on current and future challenges of the field. He has been awarded an Advanced Grant of the European Research Council to carry out the project CompMusic aimed at promoting multicultural approaches in music computing research which produced the resources this tutorial will present.

Martin Clayton is Professor in Ethnomusicology in Durham University. He studied at the School of Oriental and African Studies (SOAS) in London, where he obtained degrees in Music and Hindi (BA, 1988) and Ethnomusicology (PhD, 1993). His research interests

include Hindustani (North Indian) classical music, rhythmic analysis, musical entrainment and embodiment, comparative musicology and early field recordings, British-Asian music and Western music in India. He currently directs the research project Interpersonal Entrainment in Music Performance, a collaborative, multidisciplinary effort to understand ensemble synchronisation and coordination cross-culturally.

Bariş Bozkurt is a Post-doctoral researcher in the Music Technology Group at the Universitat Pompeu Fabra in Barcelona. In the first phase of his research career, he dedicated to development of signal processing algorithms for speech analysis and synthesis. He has obtained his PhD degree in 2005 from Faculte Polytechnique De Mons, Belgium and also worked in speech industry after his PhD. Since 2007, he has been teaching in Electrical Engineering and Computer Science departments, and carrying research in the fields of audio signal processing and computational musicology.

Tutorial 3

Statistical Analysis of Results in Music Information Retrieval: Why and How

Julián Urbano and Arthur Flexer

Abstract

Nearly since the beginning, the ISMIR and MIREX communities have promoted rigor in experimentation through the creation of datasets and the practice of statistical hypothesis testing to determine the reliability of the improvements observed with those datasets. In fact, MIR researchers have adopted a certain way of going about statistical testing, namely non-parametric approaches like the Friedman test and multiple comparisons corrections like Tukey's. In a way, they have become a standard of reporting and judging results for researchers, reviewers, committees, journal editors, etc. It is nowadays more frequent to require statistically significant improvements over a baseline with a well-established dataset.

But hypothesis testing can be very misleading if not well understood. To many researchers, especially newcomers, even the simpler analyses and tests are seen as a black box where one puts performance scores and gets a p-value which, as they are told, must be smaller than 0.05. Therefore, significance tests are in part responsible of determining what gets published, what research lines to follow, and what project to fund, so it is very important to understand what they really mean and how they should be carried out and interpreted. We will also focus on experimental validity, and will show how a lack of internal or external validity, even if experiments are reliable and repeatable and hypothesis testing is done correctly, can render even your best results invalid. Problems discussed include adversarial examples or the lack of inter-rater agreement when annotating ground truth data.

The goal of this tutorial is to help MIR researchers and developers get a better understanding of how these statistical methods work and how they should be interpreted. Starting from the very beginning of the evaluation process, it will show that statistical analysis is always required, but that too much focus on it, or the incorrect approach, is just harmful. The tutorial will attempt to provide better insight into statistical analysis of results, present better solutions and guidelines, and point the attendees to the larger but ignored problems of evaluation and reproducibility in MIR.

The work presented in this tutorial was supported by the Vienna Science and Technology Fund (WWTF, project MA14-018).

The work presented in this tutorial was supported by the European Commission H2020 project TROMPA (770376-2).

Julián Urbano is an Assistant Professor at Delft University of Technology, The Netherlands. His research is primarily concerned with evaluation in IR, working in both the music and text domains. Current topics of interest are the application of statistical methods for the construction of datasets, the reliability of evaluation experiments, statistical significance testing for IR, low-cost evaluation and stochastic simulation for evaluation. He has published over 50 research papers in related venues like Foundations and Trends in IR, the IR Journal,

the Journal of Multimedia IR, ISMIR, CMMR, ACM SIGIR, ACM CIKM and ECIR, winning two best paper awards and a best reviewer award. He has been active in the ISMIR community since 2010, both as author and PC member, and is also co-organizer of the MediaEval AcousticBrainz task. He is reviewer for other conferences and journals, such as ACM CIKM, HCOMP, IEEE TASLP, IEEE MM, ACM TWEB, IEEE TKDE or the Information Sciences journal.

Arthur Flexer is a senior researcher, project manager and vice-head at the ‘Intelligent Music Processing and Machine Learning Group’ of the Austrian Research Institute for Artificial Intelligence (OFAI). He has twenty-five years of experience in basic research on machine learning with an emphasis on applications to music in the last twelve years. He holds a PhD in psychology which provides him with the necessary background concerning design and evaluation of experiments. He has published comprehensively on the role of experiments and on problems of ground truth and inter-rater agreement, all in the field of MIR. He is author and co-author of more than 80 peer-reviewed articles. He has been active in the ISMIR community since 2005 and has also published in related venues like DAFx, SMC, ECIR, Journal of Machine Learning Research and Journal of New Music Research. He is a member of the editorial board of the Transactions of the International Society for Music Information Retrieval (TISMIR).

Tutorial 4

Music Separation with DNNs: Making It Work

Antoine Liutkus and Fabian-Robert Stöter

Abstract

This tutorial concerns music source separation, that we also call music demixing, with a resolute focus on methods using DNN.

- In an introductory part, we will motivate the tutorial by explaining how music separation with DNN emerged with data-driven methods coming from machine-learning or image processing communities. This comes with machine-learning tricks to make methods work in practice. Meanwhile, many audio processing good practices are often forgotten or not correctly applied, although they are mandatory for good performance.
- In a second part, we present and discuss the few concepts that are mandatory to design a source separation method. Each point will firstly be the focus of screencasting from an interactive notebook session that all the audience will be invited to, and then will also be explained with a theoretical presentation when appropriate. The whole tutorial will be thus split into practical hands-on sessions using online interactive Python sessions and more classical theoretical insights.
- The third part of the tutorial provides some feedback on what seems to be important to get good performance in practice, with a focus on the training stage. On the one hand, many of the tricks discussed there are not often discussed in papers because a lot of them are negative results that are hard to publish: some interesting ideas that turn out ineffective yet. On the other hand, we also show how some very simple things make a huge difference in practice.
- In the following part, we pick one single system, resulting from the previous discussion, and show how its performance can be dramatically improved by using just a few simple tricks at test time, including resynthesis methods, filtering tricks, and how to go stereo.

This tutorial is first targeted at PhD students and at engineers, that want to implement audio demixing methods in practice and to achieve state of the art performance while keeping highly readable code. Second, by showing how pytorch enables easy design and debugging, including new cost functions, architectures, etc., it will hopefully be of interest to researchers wondering how to do actual investigations on audio with DNNs, without being just users of high-level black-box systems.

Antoine Liutkus received the State Engineering degree from Télécom ParisTech, France, in 2005, and the M.Sc. degree in acoustics, computer science and signal processing applied to music (ATIAM) from the Université Pierre et Marie Curie (Paris VI), Paris, in 2005. He worked as a research engineer on source separation at Audionamix from 2007 to 2010 and obtained his PhD in electrical engineering at Télécom ParisTech in 2012. He is currently researcher at Inria, France. His research interests include audio source separation and machine learning.

Fabian-Robert Stöter received the diploma degree in electrical engineering in 2012 from the Leibniz Universität Hannover and worked towards his Ph.D. degree in audio signal processing in the research group of B. Edler at the International Audio Laboratories Erlangen,

Germany. He is currently researcher at Inria, France. His research interests include supervised and unsupervised methods for audio source separation and signal analysis of highly overlapped sources.

Tutorial 5

Deep Learning for MIR

Alexander Schindler, Thomas Lidy, and Sebastian Böck

Abstract

Deep Learning has become state of the art in visual computing and continuously emerges into the Music Information Retrieval (MIR) and audio retrieval domain. To bring attention to this topic we provide an introductory tutorial on deep learning for MIR. Besides a general introduction to neural networks, the tutorial covers a wide range of MIR relevant deep learning approaches. Convolutional Neural Networks are currently a de-facto standard for deep learning based audio retrieval. Recurrent Neural Networks have proven to be effective in onset detection tasks such as beat or audio-event detection. Siamese Networks have shown to be effective in learning audio representations and distance functions specific for music similarity retrieval. We introduce these different neural network layer types and architectures on the basis of standard MIR tasks such as music classification, similarity estimation and onset detection. We will incorporate both academic and industrial points of view into the tutorial. The tutorial will be accompanied by a Github repository for the presented content as well as references to state of the art work and literature for further reading. This repository will remain public after the conference.

Alexander Schindler is member of the Music Information Retrieval group at the Technical University since 2010 where he actively participates in research, various international projects and currently finishes his Ph.D on audio-visual analysis of music videos. He participates in teaching MIR, machine learning and DataScience. Alexander is currently employed as scientist at the AIT Austrian Institute of Technology where he is responsible for establishing a deep learning group. In various projects he focusses on deep-learning based audio-classification, audio event-detection and audio-similarity retrieval tasks.

Thomas Lidy has been a researcher in music information retrieval in combination with machine learning at TU Wien since 2004. Since 2015, he has been focusing on how Deep Learning can further improve music & audio analysis, winning 3 international benchmarking contests. He is currently the Head of Machine Learning at Musimap, a company that uses Deep Learning to analyze styles, moods and emotions in the global music catalog, in order to create emotion-aware search & recommender engines that empower music supervisors to find the music for their needs and music streaming platforms to deliver the perfect playlists according to people's mood.

Sebastian Böck received his diploma degree in electrical engineering from the Technical University in Munich in 2010 and his PhD in computer science from the Johannes Kepler University Linz. He continued his research at the Austrian Research Institute for Artificial Intelligence (OFAI) and recently also joined the MIR team at the Technical University of Vienna. His main research topic is the analysis of time event series in music signals, with a strong focus on artificial neural networks.

Tutorial 6

Fundamental Frequency Estimation in Music

Rachel Bittner, Alain de Cheveigné, and Johanna Devaney

Abstract

This tutorial will cover the core concepts in fundamental frequency estimation, starting with the monophonic case and building up to the polyphonic case. Topics will include pitch perception, common algorithms, data and annotation, and evaluation metrics. This tutorial aims to provide the audience with an understanding of the challenges, the common approaches, and the open problems in f0 estimation, as well as the potential applications of f0 estimation to transcription, music performance analysis, and source separation problems.

Rachel Bittner is a Research Scientist at Spotify in New York City, and recently completed her Ph.D. at the Music and Audio Research Lab at New York University under Dr. Juan P. Bello. Previously, she was a research assistant at NASA Ames Research Center working with Durand Begault in the Advanced Controls and Displays Laboratory. She did her master's degree in math at NYU's Courant Institute, and her bachelor's degree in music performance and math at UC 2 Irvine. Her research interests are at the intersection of audio signal processing and machine learning, applied to musical audio. Her dissertation work applied machine learning to various types of fundamental frequency estimation.

Alain de Cheveigné trained in physics, maths and neuroscience at Université Pierre et Marie Curie, and is currently Senior Scientist with the Centre National de la Recherche Scientifique (CNRS) in France, affiliated with Ecole normale supérieure (Paris) and UCL (London). He is active in psychophysics and modeling of auditory perception, and data processing for audio and electrophysiological signals. He is author of the widely used YIN method for f0 estimation, and of several widely cited chapters on pitch perception. He currently heads a H2020 project on the cognitive control of hearing aids.

Johanna Devaney is an Assistant Professor of Music Technology at Brooklyn College, City University of New York and the speciality chief editor for the Digital Musicology section of *Frontiers in Digital Humanities*. Previously she taught in the Music Technology program at NYU Steinhardt and the Music Theory and Cognition program at Ohio State University. Johanna completed her post-doc at the Center for New Music and Audio Technologies (CNMAT) at the University of California at Berkeley and her PhD in music technology at the Schulich School of Music of McGill University. She also holds an MPhil degree in music theory from Columbia University, as well as an MA in composition from York University in Toronto. Johanna's research seeks to understand how humans engage with music, primarily through performance, with a particular focus on intonation in the singing voice, and how computers can be used to model and augment our understanding of this engagement.

Tutorial 7

Optical Music Recognition for Dummies

Jorge Calvo-Zaragoza, Jan Hajič jr, Alexander Pacha, and Ichiro Fujinaga

Abstract

Optical Music Recognition (OMR) is a field of research that investigates how to computationally decode music notation in documents. As most musical compositions in the Western tradition have been written rather than recorded, bringing this music into the digital domain can significantly diversify the sources for MIR, digital musicology, and more broadly lower the costs of introducing previously unheard works to audiences worldwide. While OMR has been regarded as a largely unsolved problem, this situation has recently shifted: new large-scale datasets and tools have been released, methods based on deep learning are successfully dealing with musical symbol detection and partial end-to-end recognition, and applications of OMR such as retrieval have started migrating from article introductions to the Results sections.

Our tutorial will present this new and rather exciting state of the art in OMR. We will demonstrate recent methods and results, introduce the audience to the tools and datasets used to achieve them, and showcase the opportunities for using OMR. Finally, we will introduce the current challenges in OMR.

After the tutorial, the participants should be familiar with state-of-the-art OMR research, and should be able to start using existing tools to integrate OMR into their own work, whether in MIR or (digital) musicology. For those interested in working on OMR themselves, the tutorial should provide a head start. The tutorial will be hands-on: if you wish to get the most out of it, be ready to follow jupyter notebooks.

Jorge Calvo-Zaragoza received his PhD degree in computer science from the University of Alicante (Spain) in 2016. He joined the Single Interface for Music Score Searching and Analysis (SIMSSA) project as Postdoctoral Fellow in 2017. He is currently the recipient of a Juan de la Cierva postdoctoral grant from the Spanish government. His scientific contribution has focused so far on ancient manuscripts, in notations like neumatic or mensural. He has also made important contributions to the pre-processing of music score images such as the development of algorithms to separate the elementary graphic layers of the document (i.e., staff, notes, text, or background). He has authored more than 20 papers about Optical Music Recognition in peer-reviewed journals and international conferences.

Jan Hajič jr. is a Ph.D. student at Charles University (Czech Republic), where he is working on Optical Music Recognition and Deep Learning since 2016 at the Center of Excellence for Multimodal Data Interpretation project. He has especially contributed to infrastructure for OMR, creating e.g. the first full-pipeline OMR dataset (MUSCIMA++). He also has a strong musical background, having studied composition at the Janáček Academy of Music and Performing Arts.

Alexander Pacha received his M.Sc. with honors in computer science from the TUM, University of Augsburg and LMU Munich (Germany) in 2013. He is a professional software engineer, trainer for clean code programming and a passionate musician and composer. Since 2017 he is a PhD student at the TU Wien (Austria) working on Optical Music Recognition and Deep Learning.

Ichiro Fujinaga is an Associate Professor and the Chair of the Music Technology Area at the Schulich School of Music at McGill University. He has Bachelor's degrees in Music/Percussion and Mathematics from University of Alberta and a Master's degree in Music Theory and a Ph.D. in Music Technology from McGill. He is currently directing a large optical music recognition and analysis project called Single Interface for Music Score Searching and Analysis (SIMSSA).

Tutorial 8

Overview and New Challenges of Music Recommendation Research in 2018

Markus Schedl, Peter Knees, and Fabien Gouyon

Abstract

The current revolution in the music industry represents great opportunities and challenges for music recommendation systems. Recommendation systems are now central to music streaming platforms, which are rapidly increasing in listenership and becoming the top source of revenue for the music industry. It is increasingly more common for a music listener to simply access music than to purchase and own it in a personal collection. In this scenario, recommendation calls no longer for a one-shot recommendation for the purpose of a track or album purchase, but for a recommendation of a listening experience, comprising a very wide range of challenges, such as sequential recommendation, or conversational and contextual recommendations. Recommendation technologies now impact all actors in the rich and complex music industry ecosystem (listeners, labels, music makers and producers, concert halls, advertisers, etc.).

To acknowledge these developments, we give an introductory tutorial providing an overview of music recommendation research, as well as the challenges it faces today. We focus on three use cases: automatic playlist generation, context-aware music recommendation, and recommendation in the creative process of music making.

Markus Schedl is an Associate Professor at the Johannes Kepler University Linz / Department of Computational Perception. He graduated in Computer Science from the Vienna University of Technology and earned his Ph.D. from the Johannes Kepler University Linz. Markus further studied International Business Administration at the Vienna University of Economics and Business Administration as well as at the Handelshögskolan of the University of Gothenburg, which led to a Master's degree. His main research interests include web and social media mining, information retrieval, multimedia, and music information research. He has been an active member of the MIR community since 14 years and since then co-authored almost 200 peer-reviewed research articles.

Peter Knees is Assistant Professor of the Faculty of Informatics, TU Wien, Austria. For over a decade he has been an active member of the ISMIR community, reaching out to the related areas of multimedia, text IR, and recommender systems. Apart from serving on the program committees of major conferences in the field, he has organized several workshops on topics of media retrieval. He is an experienced teacher of graduate-level courses on recommender systems and information retrieval and has given tutorials on music information retrieval at RecSys, SIGIR, ECIR, RuSSIR, and the Indonesian Summer School on MIR.

The presenter Peter Knees was supported by the Austrian Research Promotion Agency (FFG) under Bridge 1 grant number 858514 (SmarterJam).

Fabien Gouyon is Principal Scientist at the music streaming service Pandora, where he does applied research on personalized music recommendation, and works with the Music Genome Project. Before joining Pandora, he received a PhD in Computer Science while working in the Music Technology Group in the University Pompeu Fabra in Barcelona, and was a co-founder of Barcelona Music and Audio Technologies (BMAT), worked in the Austrian Research Institute for Artificial Intelligence in Vienna, and started and led the Sound and Music Computing Group while teaching at the University of Porto. He was President of ISMIR in 2016-2017.

Session A

Musical objects

A CONFIDENCE MEASURE FOR KEY LABELLING

Roman B. Gebhardt

Audio Communication Group,
TU Berlin

r.gebhardt@campus.
tu-berlin.de

Athanasios Lykartsis

Audio Communication Group,
TU Berlin

athanasios.lykartsis@
tu-berlin.de

Michael Stein

Native Instruments GmbH

michael.stein@
native-instruments.de

ABSTRACT

We present a new measure for automatically estimating the confidence of musical key classification. Our approach leverages the degree of harmonic information held within a musical audio signal (its “keyness”) as well as the steadiness of local key detections across its duration (its “stability”). Using this confidence measure, musical tracks which are likely to be misclassified, i.e. those with low confidence, can then be handled differently from those analysed by standard, fully automatic key detection methods. By means of a listening test, we demonstrate that our developed features significantly correlate with listeners’ ratings of harmonic complexity, steadiness and the uniqueness of key. Furthermore, we demonstrate that tracks which are incorrectly labelled using an existing key detection system obtain low confidence values. Finally, we introduce a new method called “root note heuristics” for the special treatment of tracks with low confidence. We show that by applying these root note heuristics, key detection results can be improved for minimalistic music.

1. INTRODUCTION

A major commercial use case of musical key detection is its application in DJ software programs including Native Instruments’ Traktor¹ and Pioneer’s rekordbox². It represents the basis for harmonic music mixing [9], a DJing technique which is mostly bounded to electronic dance music (EDM). However, the concept of musical key is not universally applicable to all styles of music, especially those of a minimalistic nature, which is often the case in (EDM) [7, 10, 21]. A particular challenge of key detection in EDM is that the music often does not follow classic Western music standards in terms of its harmonic composition and progression. This applies to a broad range of contemporary EDM music which can be composed in

¹<https://www.native-instruments.com/de/products/traktor/>

²<https://rekordbox.com/de/>

a chromatic space or, if following classic characteristics, uses more “exotic” modes such as e.g. Phrygian [19], which is actually predominant for certain genres such as Acid House, Electronic Body Music (EBM) and New Beat, which, since the 1980s represent a prominent source of inspiration for contemporary EDM. A further difficulty is the tendency of certain electronic music to be strongly percussive and very minimalistic in terms of its harmonic content [5]. In fact, following pioneering groups like Kraftwerk, melodic minimalism is a main characteristic of techno music [13]. Today, a wide range of EDM productions are exclusively percussion-based. The lack of harmonic information clearly leads to problems in assigning an unambiguous key label, which is still the most widely used way to describe a track in its harmonic composition [21].

In the recent years, confidence measures have gained interest in the field of MIR, namely related to tempo estimation [8, 17]. The described scenario motivates to establish such measure for key detection tasks. Crucial factors to consider are the degree to which a musical audio signal conforms to the concept of musical key, and furthermore to explore where a single key persists throughout a recording. Being able to capture this information automatically could therefore serve as an indicator to predict potential misclassifications. It may also be used to define a threshold to decide whether to label a track with a key or alternatively simply with a root note [10], within a genre-specific framework [21] or in spatial coordinates [2, 3, 12]. Alternatively, multiple key labels could be assigned for tracks containing key changes [16]. We collate this information to derive a key detection confidence measure and present an alternative means for handling music where a traditional key assignment is not possible. The remainder of this paper is structured as follows: in Section 2, we present the development of the confidence features as well as a special key detection method for tracks of a minimalistic nature. Section 3 outlines our evaluation of the developed features and the special treatment of low confidence scoring tracks. Finally, we conclude our work and provide an outlook for future work in Section 4.

2. METHOD

To establish the confidence measure, we follow two hypotheses and for each we develop a feature: First, there must be sufficient harmonic information within the signal



to reliably determine a key, i.e., it would be inappropriate to label a track consisting exclusively of percussive content with a meaningful key. Consequently, we denote our first confidence feature as *keyness* to indicate the amount of harmonic content within a musical piece. Second, we state that any local key changes throughout the duration of a track will inevitably lead to a discrepancy between a given global label and at least some regions. Our second confidence feature, which measures the steadiness of key information, will be referred to as *stability*. The development of both features is discussed in the following subsections.

2.1 Keyness

Various approaches have been taken to the problem of assigning a musical key designation based on the information retrieved from an audio signal. A straightforward method would be to follow the well-known key template approach introduced by Krumhansl et al. [15], where the correlation of an input signal's chroma distribution with the chosen key's template could be used as a keyness measure. Often, these templates are not needed, for instance when the key detection is handled within a tonal space model like Chew's Spiral Array [3] or Harte et al.'s Tonal Centroid Space [12]. To avoid the necessity of computing the correlations and to keep our approach most simple, we bypass this option and retrieve keyness information directly from the chromagram. For this, we use a chromagram representation which emphasizes tonal content, based on a perceptually inspired filtering process in [10]. This procedure removes energy in the chromagram evoked by noisy and/or percussive sounds, which are especially present in EDM. We then apply Chuan et al.'s fuzzy analysis technique [4] to further "clean" the chromagram. Figure 1 shows the resulting chromagram of an EDM track³ with a temporal resolution of 250 ms and below it, the curve resulting from the sum of the frame-wise individual chroma energies $E(c, t)$ ranging from 0 to 1 for each chroma c at time-frame t :

$$E_c(t) = \sum_{c=1}^{12} E(c, t). \quad (1)$$

We denote $E_c(t)$, the *chroma energy*. By inspection of the resulting curve, a raw subdivision of the track into three partly recurring harmonic structures can be observed: The first with a chroma energy equal (or close to) zero is present in the purely percussive regions which accord to our represent regions of *low* keyness. The second structure describes the G# power chord (where G# is the root and D# the fifth), which reaches chroma energy values of 1 to approximately 1.75 for $E_c(t)$. The power chord is widely used in EDM productions and is ambiguous in terms of the mode of its tonic's key due to the third missing. Finally, the third structure in the middle of the track holds a

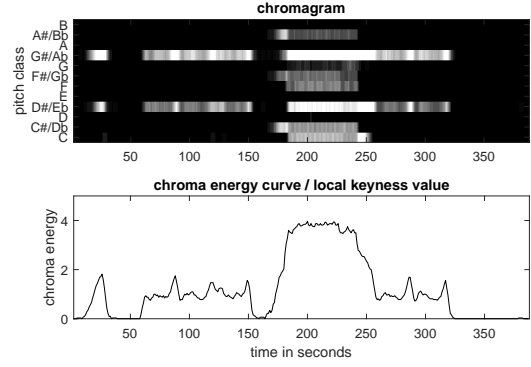


Figure 1. Chromagram (upper plot) and local keyness curve (lower plot) of an EDM track derived from the framewise energies of the chromagram.

chroma energy level of approximately 4 which far exceeds the other regions. In fact, it is the only region that contains a sufficient number of notes present to use as the basis for detecting the key. As this representative example demonstrates, the straightforward calculation of chroma energy can be informative about how much harmonic information is contained in a musical audio signal.

To obtain a global keyness measure, we average the chroma energy vector $E_c(t)$ over the full duration T of the track and obtain the keyness value, K :

$$K = \frac{1}{T} \cdot \sum_{t=0}^T E_c(t). \quad (2)$$

2.2 Stability

The second confidence feature, stability, is derived from the steadiness of key classifications throughout the full duration of the track. For this purpose, we take into account the vector of local key detections using a template-based approach on temporal frames with 250 ms length and 125 ms hop-size. In DJ software which was the framework of our research, the 24 key classes are usually displayed in the 12-dimensional subspace of so-called "Camelot numbers" [6] each of which corresponds to a certain "hour" on the circle of fifths. This implies that a major key and its relative minor are considered equivalent. The middle plot of Figure 2 shows the progression of Camelot classifications over time. It is important to note that both the vertical axis of the middle plot and the horizontal axis of the lower histogram plot are circular i.e. the chroma has been "wrapped". In our example, the most frequently detected Camelot number is 1 (B/G# m) which is followed by its direct neighbour one fifth above, 2 (F# / Ebm). The right tail of the distribution fades out with small counts for numbers 3 (Db/Bbm) and 4 (Ab/Fm), whereas the left tail's only present value is 11 (A/F# m). For a high degree of stability, we would expect a low angular spread of camelot detections throughout, which we compute in terms of the circular variance $V(cam)$ of the distribution according to [1]. In terms of a numeric measure for the stability

³ Praise You 2009 (Fatboy Slim vs. Fedde Le Grand Dub): <https://www.discogs.com/de/Fatboy-Slim-vs-Fedde-Le-Grand-Praise-You-2009/release/1967533>

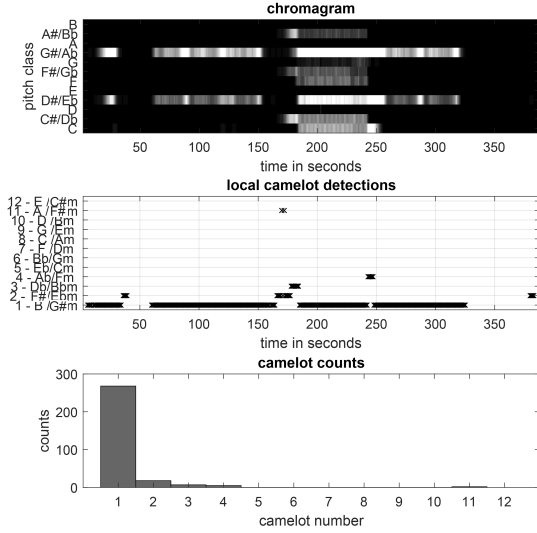


Figure 2. Local camelot decisions (middle plot) and histogram of absolute camelot counts (lower plot). The number describes the “hour” on the circle of fifths.

of the whole track, we define the confidence feature of stability, S , as:

$$S = 1 - V(cam), \quad (3)$$

with $V(cam)$ depicting the circular variance of the camelot vector. Thus, the stability of a track will be 0 for a uniform histogram and 1 for maximum stability (where only one camelot number is detected throughout). In more complex compositions in classical music, we can expect key changes throughout musical pieces. However, these key changes are usually small moves on the circle of fifths and consequently small steps on the Camelot wheel (e.g. just one “hour” for a fifth). When using the circular histogram, these key changes would not have a strong impact on the variance of the distribution and would therefore exert only a small influence on the stability feature. In the special case of pop or EDM, key modulation is mostly absent [7].

2.3 An Overall Confidence Feature

In the two previous subsections, we discussed the development of two features to measure the *keyness* and *stability* of musical audio, both representing independent approaches to find a quantitative measure for the overall confidence of a key detection. As discussed, the two features focus on different characteristics of the music signal. While the keyness measure describes the amount of harmonic information held by a track, the stability feature focusses on the steadiness of key detections throughout a whole track. Collectively these features will penalise the presence of key changes within a track as well as “random” labels from a key classification system caused by harmonic structures which don’t conform to the classic major / minor distribution. We state that, for a “trustworthy” key detection

which is informative for harmonic mixing, a given track should score high for both of these features. Thus, we define an overall confidence feature as the linear combination, C , of the subfeatures K and S with variable weighting parameters κ and σ . We quantise K and S and discretise them individually to evenly distributed percentiles, resulting in C_k for K and C_s for S . As a result, the lowest percentile of 1 comprises tracks scoring lower in K (or S respectively) than 99% of the database which is discussed in Section 3.2. This is done to ensure an even distribution of the subfeature values over all tracks as well as to map both to a range from 1 to 100:

$$C = \frac{\kappa \cdot C_k + \sigma \cdot C_s}{\kappa + \sigma} \quad (4)$$

We consider the choice of κ and σ to be genre-dependent. For minimalistic music such as EDM, where we do not expect highly complex harmonic structure or key changes that would eventually lead to a low score for C_s , we believe greater emphasis should be given to C_k to filter e.g. purely percussive tracks. However, for the analysis of classical music, more importance should be attributed to the stability feature C_s . Here, we should not expect a lack of harmonic information, but frequent and “far” key changes would lead to less clarity about the key the piece is composed in. In this paper, we set the values of $\kappa = 5$ and $\sigma = 2$ for the evaluation of a database mainly containing EDM tracks, however we intend to explore the effect of modifying these values and genre-specific parameterisations in future work.

2.4 Root Note Heuristics

With the proposed confidence feature, C , it is possible to determine a threshold below which a key detection should not be considered reliable. This raises an important question of how to treat problematic (i.e. low confidence) tracks in terms of assigning a key label. One option could be the use of multiple key labels for tracks with low stability [16] or to use root note labelling for tracks with low keyness [10]. Alternatively, for EDM, minimalistic tracks could be labelled as the root note’s minor key due to the strong bias towards minor mode in this genre [7, 14]. We call this procedure “*root note heuristics*” and apply it to tracks whose keyness falls below a certain threshold. For the case of root note detection, we first accumulate the chroma energies $E(c, t)$ over time to obtain a global chroma energy vector $\underline{E}(c)$:

$$\underline{E}(c) = \sum_{t=0}^T E(c, t). \quad (5)$$

To detect the most predominant chroma, and hence *root note*, we apply a simple binary template $T(c)$ in which the referenced chroma and its dominant are given an equal weight of 1, with all pitch classes set to 0. Consideration of the fifth interval is made to explicitly take power chords into account and allow them to point towards their root. We shift this template circularly by one step for each chroma value accordingly and calculate the inner product per shift.

This results in the likelihood $R(c)$ of the chroma c to be the root of the track:

$$R(c) = \langle T(c), \underline{E}(c) \rangle \quad (6)$$

Finally, the minor mode of the chroma with the highest value of $R(c)$ is assigned to the track as a whole.

3. EVALUATION

For an extensive analysis of our developed confidence features, we undertook two separate evaluation procedures. First, to examine the validity of our subfeatures *keyness* and *stability*, we conducted a listening test where we asked participants to rate a set of musical audio examples according to three questions concerning their harmonic content. Second, we evaluated the degree to which the calculated confidence score for each single track would be associated with a given genre label and whether it was detected correctly by a key detection system and - if not - whether the error was close to the ground truth key label or not. Hence, we would then be able to use the confidence score as a prediction measure for the potential rejection of a key decision and eventually the special treatment of the corresponding tracks. Both approaches are discussed in the following subsections.

3.1 Listening Test for Subfeature Evaluation

The listening experiment was performed as an online survey, in which we presented 12 different representative excerpts⁴ of length 120 s which we considered sufficient to allow the perception of any potential key changes. These 12 excerpts could be characterised by the following four properties A - D:

- A: Clear and unique key throughout (Track IDs 1, 8, 12)
- B: Change in key structure (Track IDs 2, 7, 10)
- C: Non-Western melodic content (Track IDs 3, 4, 6)
- D: No or little melodic content (Track IDs 5, 9, 11)

After listening to the audio samples, participants were asked to rate them on a 10-point Likert scale in terms of their harmonic complexity, i.e. whether the tracks followed the major/minor scheme and how clearly they adhered to one unique key throughout. In order to prevent any bias in the participant ratings, no information about the developed features was provided. However, a short training phase was set up before the test to ensure participants understood the questions they were going to be asked. In total, we recruited 29 participants (22 male, 7 female) who self-reported as musically trained. The participants' ages ranged from 23 to 66 with an average of 10 years of musical training. In the following sections, the relatedness of the ratings with the computed subfeatures C_k , C_s as well as the overall confidence C will be discussed.

⁴ A link to the examples will be provided in the camera ready copy.

3.1.1 Keyness

To assess the subfeature of keyness, we asked participants to rate the audio excerpts according to two questions. With the first, we aimed to test if the concept of the keyness feature as a general measure for tonal density or complexity (not necessarily relating to a key) would prove appropriate:

Q1: “To which degree do you find the presented audio harmonically complex?”

We hypothesised a positive correlation between the ratings and the computed values of C_k , however we made no assumption about the coherence of the ratings with C_s as harmonically complex excerpts could still be unstable in harmony or key. The mean ratings as well as the corresponding feature values C , C_k and C_s are displayed in the leftmost column of Figure 3. For a measure of relatedness, we calculated Spearman's rho correlation measure for the ratings' means across participants and the feature values. With a choice of $\alpha = 0.05$ as the level of significance, the observed strong positive correlation ($r_s = 0.63, p < 0.05$) between the ratings and computed values for the keyness feature C_k supports our initial hypothesis. However, some outliers can be identified, for which the formulation of the question might have been misleading: Excerpt 7 (second rated from category B) exhibits strong break beat percussion and a rather chaotic melodic progression with a short minor mode piano passage, which would contribute to a low score for C_k . Feedback from some participants revealed the excerpt was considered as rather challenging, which caused it to be rated high in terms of complexity. Excerpt 9 (the highest rated excerpt from category D) is also mostly percussive with pitched voice samples and sounds. Its relatively unusual composition might also have caused some participants to rate it “complex”. The excerpts from category A consist of quite common, repetitive chord structures which therefore may not have been perceived as particularly complex in a musical sense. However, they all feature a high amount of harmonic content, and therefore represent “complex” musical excerpts in line with our keyness definition. As discussed in 2.1, the keyness feature is derived from the average amount of tonal information throughout the analysed signal. We argued that in the case of Western music, a high amount of tonal information usually indicates the presence of a major or minor scheme as harmonic layerings of notes deviating from Western scales rarely appear [20] and thus a higher density of tonal information should point towards the clear presence of a musical key. To examine the validity of this assumption, the second question of the listening test focussed on whether the keyness feature could in fact be used as an indicator for the presence of a major/minor scheme within the audio:

Q2: “To which degree does the presented audio fit the major/minor scheme?”

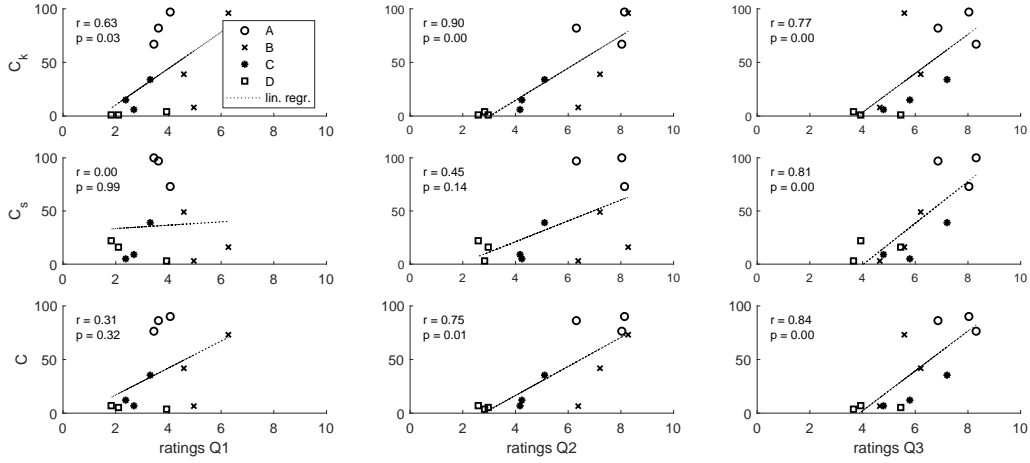


Figure 3. Mean ratings on the questions **Q1**, **Q2** and **Q3** for the 12 stimuli and their corresponding feature values C , C_k and C_s with the respective Spearman correlation coefficients r .

Again, we hypothesised a positive correlation between the ratings and C_k , but again, not C_s . The results are presented in the subplots in the middle column of Figure 3. Our hypothesis regarding C_k was supported with a very strong positive correlation of $r_s = 0.90$, $p < 0.01$. Remarkably, it even exceeds the correlation of the stronger hypothesis we explored in **Q1** regarding its relatedness to the complexity ratings, as discussed in 3.1.1: Of the four outliers discussed above, namely one excerpt from category D and all the excerpts from category A, all agree much more strongly with the C_k value. As with **Q1**, no significant correlation between the ratings and the values of C_s was observed.

3.1.2 Stability

To evaluate the stability subfeature C_s , participants were asked to rate the stimuli according to the question:

Q3: “To which certainty does the audio correspond to one unique and distinct key?”

We expected the ratings for **Q3** to be correlated with the computed values of C_s , as key changes should result in lower the ratings and stability. In addition, we also hypothesised a positive correlation to C_k as a lack of harmonic information could complicate a clear assignment to one unique key. The subplots in the rightmost column of Figure 3 show the outcomes of the third question. As can be seen, both subfeatures exhibit a significant correlation with the mean ratings. While C_k shows a strong positive correlation with a Spearman coefficient of $r_s = 0.77$, $p < 0.01$, the correlation of C_s is even stronger ($r_s = 0.81$, $p < 0.01$). The combination of both in the overall confidence feature C results in an even higher correlation $r_s = 0.84$, $p < 0.01$, which fortifies our choice to combine both features in order to explain the certainty of a unique key decision and therefore the confidence of a key assignment.

3.2 Evaluation on an annotated dataset

In the second part of our evaluation progress, we tested how the computed confidence scores relate to genre labels and whether a track’s key classification was correct or not. We based our analysis on a private commercial database comprised of 834 tracks consisting mainly of EDM (697 total) as well as 137 tracks from Harte’s [11] Beatles dataset with key labels forming the ground truth. A subset of 101 of the EDM tracks were labelled “Inharmonic” and represented tracks that were considered ambiguous or unclassifiable by musical experts.

3.2.1 Genre Specific Differences

For a first observation, we compare the means of C_k and C_s for the three different subsets, namely the Beatles, the “Inharmonic” labelled EDM subset, and the remainder of the EDM tracks. According to our model, C_k should be high for the Beatles dataset, since it contains mostly melodic music. However, we should expect lower values for the EDM set, following the hypothesis that EDM is often of a more minimalistic melodic nature. For the subset of EDM tracks labelled “Inharmonic” we shouldn’t expect much harmonic information, and hence low score for C_k . Alternatively, a lack of clarity about the label might occur due to the use of a non-Western scale, and would therefore result in a low value for C_s . We hypothesised C_s to reach higher scores for the remaining EDM tracks as we expected a more stable melodic structure for these than the Beatles tracks which inherit a number of key changes and sometimes unconventional harmonic content. The results in table 1 show that our expectations are confirmed. The “Inharmonic” subset scores substantially lower in all (sub)-features, while the Beatles dataset scores high in keyness whereas the remainder of the EDM dataset achieves high values in stability.

Subset	C_k	C_s	C
EDM	49.4	55.8	51.2
EDM Inharmonic	14.3	30.6	19.0
Beatles	82.0	42.1	70.6

Table 1. Confidence score means for the different subsets, in the range 1 - 100.

3.2.2 Prediction of Misclassification

We aimed to assess whether the confidence feature would be an appropriate indicator of the degree to which an automatic key detection could be considered trustworthy, primarily for the application of harmonic mixing. To provide automatic estimates of musical key, we used a key-template based system built into a state-of-the-art DJ software, which was modified by incorporating the pre-processing stage as proposed in [10]. Given our equalisation of relative keys to equal Camelot numbers as discussed in 3.1.2, we defined three different labelling categories: *Match* for key detections matching the ground truth label, *Fifth* for fifth related errors and thus, one Camelot number away from the ground truth and *Other* for detections greater than one Camelot number apart. Across the 834 tracks, we counted 627 *Matches*, 117 *Fifths* and 90 *Others*. Three hypotheses were put forward: We expected tracks for which our key detection result matched the ground truth to score higher in confidence than those from both other categories. We were less sure about the tracks from the *Fifth* category, but intuitively expected them to score higher than those from *Other*. Figure 4 shows the distributions of the confidence scores C within the three groups. We performed a Welch-ANOVA which supported this hypothesis with high significance, $F(2, 170.41) = 64.16, p < .001$. To test the mean differences between the three groups, we conducted a Games-Howell post-hoc analysis which showed significant differences between all three pairs for $\alpha = 0.01$.

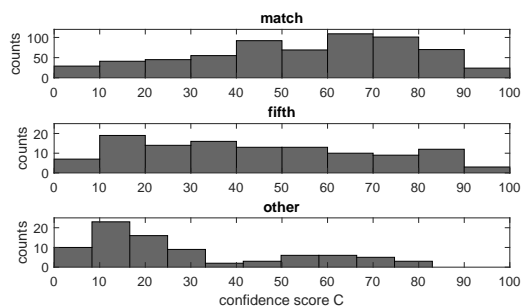


Figure 4. Distributions of the confidence scores C within the three different labelling categories.

3.2.3 Root Note Heuristics

Finally, we evaluated the special treatment of the “root note heuristics” introduced in 2.4. For this, we took into consideration the counts of the three labelling categories between

Subset	Match	Fifth	Other
EDM	469 / 468	86 / 85	41 / 43
EDM Inharmonic	50 / 59	17 / 17	34 / 25
Beatles	108 / 108	14 / 14	15 / 15

Table 2. Counts of labelling categories for the three subsets without / **with** the application of the root note heuristics method.

the different subsets. As a preliminary investigation, we applied the heuristics to the lowest sixth quantile scoring tracks. The resulting absolute counts for the labelling categories are shown in Table 2. While the Beatles and normal EDM subsets are barely affected, a clear improvement is achieved within the “Inharmonic” subset. Using the root note heuristics, the number of correctly detected tracks could be increased by 18%. Furthermore, we were able to reduce the number of *Other* classified errors by 26%.

4. CONCLUSIONS

In this paper, we described the development of a confidence feature for key labelling, as a means to measure the likelihood of an automatic key classification being correct. For this, we developed two subfeatures, keyness and stability, to estimate the amount of tonal content of musical audio as well as the steadiness of key detections throughout the full duration of the track respectively. Both subfeatures were evaluated by means of a listening test. Our analysis demonstrated high correlations for harmonic complexity, accordance to the major/minor scheme and the uniqueness of one key between the participants’ ratings and the developed features. Furthermore, we showed that our confidence feature can be helpful indicator of cases where an automatic estimated key label can be trusted. Our confidence measure may also be used as a threshold to switch between different key detection approaches. To this end, we introduced a root note heuristics method that can be used as a special key detection approach for tracks of harmonically minimalistic nature, and we showed that the application of this procedure could positively affect key detection performance. However, the presented root note heuristics approach is still at an early stage of development, therefore these promising results motivate continued research towards adjusting the threshold and further development of alternative key detection methods. This work has mostly been focussed on EDM. A major area of future work would therefore be to generalise the key confidence concept for other genres, where it would be necessary to also take into account relative errors instead of considering only in the Camelot subspace. Also, other possible ways to use the developed features can be considered: Since the keyness feature is sequentially analysed over time, this allows inference about individual segments of a track. In the context of harmonic mixing, this information could be extremely useful by allowing a DJ to locate appropriate regions for executing the transition between two tracks, thus avoiding harmonic clashes [9, 18].

5. REFERENCES

- [1] P. Berens. Circstat: A MATLAB toolbox for circular statistics. *Journal of Statistical Software*, 31(10):1–21, 2009.
- [2] G. Bernandes, D. Cocharro, M. Caetano, and M.E.P. Davies. A multi-level tonal interval space for modelling pitch relatedness and musical consonance. *Journal of New Music Research*, 45(4):281–294, 2016.
- [3] E. Chew. *Towards a mathematical model of Tonality*. Ph.D. thesis, MIT, Cambridge, MA, 2000.
- [4] C.-H. Chuan and E. Chew. Fuzzy analysis in pitch class determination for polyphonic audio key finding. In *Proc. of the 6th International Society for Music Information Retrieval (ISMIR 2005) Conference*, pages 296–303, 2005.
- [5] G. Dayal and E. Ferrigno. *Electronic Dance Music*. Grove Music Online, Oxford University Press, 2012.
- [6] Á Faraldo. *Tonality Estimation in Electronic Dance Music*. Ph.D. thesis, UPF, Barcelona, 2017.
- [7] Á Faraldo, E. Gómez, S. Jordà, and P. Herrera. Key estimation in electronic dance music. In *Proc. of the 38th European Conference on Information Retrieval*, pages 335–347, 2016.
- [8] F. Font and X. Serra. Tempo estimation for music loops and a simple confidence measure. In *Proc. of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, pages 269–275, 2016.
- [9] R.B. Gebhardt, M.E.P. Davies, and B.U. Seeber. Psychoacoustic approaches for harmonic music mixing. *Applied Sciences*, 6(5):123, 2016.
- [10] R.B. Gebhardt and J. Margraf. Applying psychoacoustics to key detection and root note extraction in EDM. In *Proc. of the 13th International Symp. on CMMR*, pages 482–492, 2017.
- [11] C. Harte. *Towards automatic extraction of harmony from music signals*. Ph.D. thesis, University of London, London, 2010.
- [12] C. Harte, M. Sandler, and M. Gasser. Detecting harmonic change in musical audio. In *Proc. of the 1st ACM workshop on Audio and music computing multimedia*, pages 21–26, 2006.
- [13] J. Hemming. *Methoden der Erforschung populärer Musik*. Springer VS, Wiesbaden, 2016. In German.
- [14] P. Knees, Á. Faraldo, P. Herrera, R. Vogl, S. Böck, F. Hörschläger, and M. Le Goff. Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections. In *Proc. of the 16th International Society for Music Information Retrieval (ISMIR 2015) Conference*, pages 364–370, 2015.
- [15] C. Krumhansl, E. Kessler, and J. Edwards. Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. *Psychological Review*, 89(4):334–368, 1982.
- [16] K. Noland and M.B. Sandler. Key estimation using a hidden markov model. In *Proc. of the 7th International Society for Music Information Retrieval (ISMIR 2006) Conference*, pages 121–126, 2006.
- [17] J. Pauwels, K. O’Hanlon, G. Fazekas, and M.B. Sandler. Confidence measures and their applications in music labelling systems based on hidden markov models. In *Proc. of the 18th Conference of the International Society for Music Information Retrieval (ISMIR 2017)*, pages 279–279, 2017.
- [18] M. Spicer. (ac)cumulative form in pop-rock music. *Twentieth Century Music*, 1(1):29 – 64, 2004.
- [19] P. Tagg. From refrain to rave: The decline of figure and the rise of ground. *Popular Music*, 13(2):209 – 222, 1994.
- [20] D. Temperley. *Music and Probability*. MIT Press, Cambridge, MA, 2007.
- [21] R. Wooller and A. Brown. A framework for discussing tonality in electronic dance music. In *Proc. Sound: Space - The Australasian Computer Music Conference*, pages 91 – 95, 2008.

IMPROVED CHORD RECOGNITION BY COMBINING DURATION AND HARMONIC LANGUAGE MODELS

Filip Korzeniowski and Gerhard Widmer

Institute of Computational Perception,
Johannes Kepler University, Linz, Austria
filip.korzeniowski@jku.at

ABSTRACT

Chord recognition systems typically comprise an acoustic model that predicts chords for each audio frame, and a temporal model that casts these predictions into labelled chord segments. However, temporal models have been shown to only smooth predictions, without being able to incorporate musical information about chord progressions. Recent research discovered that it might be the low hierarchical level such models have been applied to (directly on audio frames) which prevents learning musical relationships, even for expressive models such as recurrent neural networks (RNNs). However, if applied on the level of chord sequences, RNNs indeed can become powerful chord predictors. In this paper, we disentangle temporal models into a harmonic language model—to be applied on chord sequences—and a chord duration model that connects the chord-level predictions of the language model to the frame-level predictions of the acoustic model. In our experiments, we explore the impact of each model on the chord recognition score, and show that using harmonic language and duration models improves the results.

1. INTRODUCTION

Chord recognition methods recognise and transcribe musical chords from audio recordings. Chords are highly descriptive harmonic features that form the basis of many kinds of applications: theoretical, such as computational harmonic analysis of music; practical, such as automatic lead-sheet creation for musicians¹ or music tutoring systems²; and finally, as basis for higher-level tasks such as cover song identification or key classification. Chord recognition systems face the two key problems of extracting meaningful information from noisy audio, and casting this information into sensible output. These translate to *acoustic modelling* (how to predict a chord label for each position or frame in the audio), and *temporal modelling* (how to create

meaningful segments of chords from these possibly volatile frame-wise predictions).

Acoustic models extract frame-wise chord predictions, typically in the form of a distribution over chord labels. Originally, these models were hand-crafted and split into feature extraction and pattern matching, where the former computed some form of pitch-class profiles (e.g. [26, 29, 33]), and the latter used template matching or Gaussian mixtures [6, 14] to model these features. Recently, however, neural networks became predominant for acoustic modelling [18, 22, 23, 27]. These models usually compute a distribution over chord labels directly from spectral representations and thus fuse both feature extraction and pattern matching. Due to the discriminative power of deep neural networks, these models achieve superior results.

Temporal models process the predictions of an acoustic model and cast them into coherent chord segments. Such models are either task-specific, such as hand-designed Bayesian networks [26], or general models learned from data. Here, it is common to use hidden Markov models [8] (HMMs), conditional random fields [23] (CRFs), or recurrent neural networks (RNNs) [2, 32]. However, existing models have shown only limited capabilities to improve chord recognition results. First-order models are not capable of learning meaningful musical relations, and only smooth the predictions [4, 7]. More powerful models, such as RNNs, do not perform better than their first-order counterparts [24]. In addition to the fundamental flaw of first-order models (chord patterns comprise more than two chords) both approaches are limited by the low hierarchical level they are applied on: the temporal model is required to predict the next symbol for each audio frame. This makes the model focus on short-term smoothing, and neglect longer-term musical relations between chords, because, most of the time, the chord in the next audio frame is the same as in the current one. However, exploiting these longer-term relations is crucial to improve the prediction of chords. RNNs, if applied on *chord sequences*, are capable of learning these relations, and become powerful chord predictors [21].

Our contributions in this paper are as follows: i) we describe a probabilistic model that allows for the integration of chord-level language models with frame-level acoustic models, by connecting the two using chord duration models; ii) we develop and apply chord language models and chord duration models based on RNNs within this framework;

¹ <https://chordify.net/>

² <https://yousician.com>



© Filip Korzeniowski and Gerhard Widmer. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: Filip Korzeniowski and Gerhard Widmer. “Improved Chord Recognition by Combining Duration and Harmonic Language Models”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

and iii) we explore how these models affect chord recognition results, and show that the proposed integrated model outperforms existing temporal models.

2. CHORD SEQUENCE MODELLING

Chord recognition is a sequence labelling task, i.e. we need to assign a categorical label $y_t \in \mathcal{Y}$ (a chord from a chord alphabet) to each member of the observed sequence \mathbf{x}_t (an audio frame), such that y_t is the harmonic interpretation of the music represented by \mathbf{x}_t . Formally,

$$\hat{y}_{1:T} = \operatorname{argmax}_{y_{1:T}} P(y_{1:T} | \mathbf{x}_{1:T}). \quad (1)$$

Assuming a generative structure as shown in Fig. 1, the probability distribution factorises as

$$P(y_{1:T} | \mathbf{x}_{1:T}) \propto \prod_t \frac{1}{P(y_t)} P_A(y_t | \mathbf{x}_t) P_T(y_t | y_{1:t-1}),$$

where P_A is the acoustic model, P_T the temporal model, and $P(y_t)$ the label prior which we assume to be uniform as in [31].

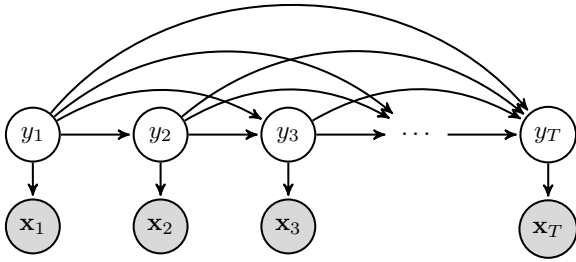


Figure 1. Generative chord sequence model. Each chord label y_t depends on all previous labels $y_{1:t-1}$.

The temporal model P_T predicts the chord symbol of each audio frame. As discussed earlier, this prevents both finite-context models (such as HMMs or CRFs) and unrestricted models (such as RNNs) to learn meaningful harmonic relations. To enable this, we disentangle P_T into a *harmonic language model* P_L and a *duration model* P_D , where the former models the harmonic progression of a piece, and the latter models the duration of chords.

The language model P_L is defined as $P_L(\bar{y}_k | \bar{y}_{1:k-1})$, where $\bar{y}_{1:k} = \mathcal{C}(y_{1:k})$, and $\mathcal{C}(\cdot)$ is a sequence compression mapping that removes all consecutive duplicates of a chord (e.g. $\mathcal{C}((C, C, F, F, G)) = (C, F, G)$). The frame-wise labels $y_{1:t}$ are thus reduced to chord changes, and P_L can focus on modelling these.

The duration model P_D is defined as $P_D(s_t | y_{1:t-1})$, where $s_t \in \{c, s\}$ indicates whether the chord changes (c) or stays the same (s) at time t . P_D thus only predicts whether the chord will change or not, but not which chord will follow—this is left to the language model P_L . This definition allows P_D to consider the preceding *chord labels* $y_{1:t-1}$; in practice, we restrict the model to only depend on

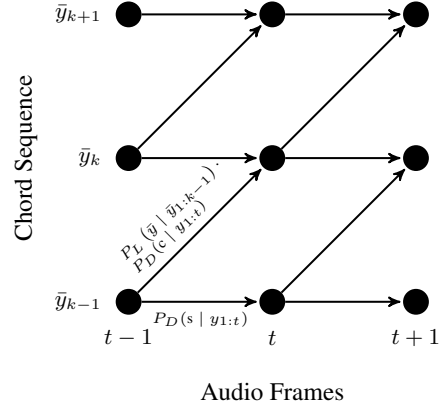


Figure 2. Chord-time lattice representing the temporal model P_T , split into a language model P_L and duration model P_D . Here, $\bar{y}_{1:K}$ represents a concrete chord sequence. For each audio frame, we move along the time-axis to the right. If the chord changes, we move diagonally to the upper right. This corresponds to the first case in Eq. 2. If the chord stays the same, we move only to the right. This corresponds to the second case of the equation.

the preceding chord *changes*, i.e. $P_D(s_t | s_{1:t-1})$. Exploring more complex models of harmonic rhythm is left for future work.

Using these definitions, the temporal model P_T factorises as

$$P_T(y_t | y_{1:t-1}) = \begin{cases} P_L(\bar{y}_k | \bar{y}_{1:k-1}) P_D(c | y_{1:t-1}) & \text{if } y_t \neq y_{t-1} \\ P_D(s | y_{1:t-1}) & \text{else} \end{cases} \quad (2)$$

The chord progression can then be interpreted as a path through a chord-time lattice as shown in Fig. 2.

This model cannot be decoded efficiently at test-time because each y_t depends on all predecessors. We will thus use either models that restrict these connections to a finite past (such as higher-order Markov models) or use approximate inference methods for other models (such as RNNs).

3. MODELS

The general model described above requires three sub-models: an acoustic model P_A that predicts a chord distribution from each audio frame, a duration model P_D that predicts when chords change, and a language model P_L that predicts the progression of chords in the piece.

3.1 Acoustic Model

The acoustic model we use is a VGG-style convolutional neural network, similar to the one presented in [23]. It uses three convolutional blocks: the first consists of 4 layers of $32 \times 3 \times 3$ filters (with zero-padding in each layer), followed by 2×1 max-pooling in frequency; the second comprises 2 layers of 64 such filters followed by the same pooling scheme; the third is a single layer of 128 12×9 filters. Each of the blocks is followed by feature-map-wise dropout with

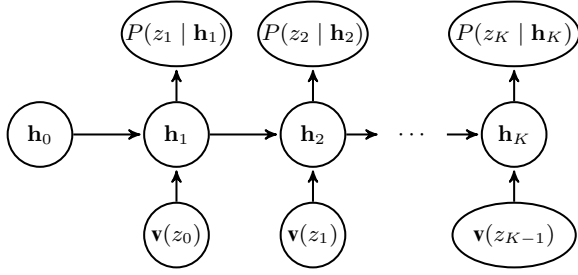


Figure 3. Sketch of a RNN used for next step prediction, where z_k refers to an arbitrary categorical input, $\mathbf{v}(\cdot)$ is a (learnable) input embedding vector, and \mathbf{h}_k the hidden state at step k . Arrows denote matrix multiplications followed by a non-linear activation function. The input is padded with a dummy input z_0 in the beginning. The network then computes the probability distribution for the next symbol.

probability 0.2, and each layer is followed by batch normalisation [19] and an ELU activation function [10]. Finally, a linear convolution with 25 1×1 filters followed by global average pooling and a softmax produces the chord class probabilities $P_A(y_t | \mathbf{x}_t)$.

The input to the network is a 1.5 s patch of a quarter-tone spectrogram computed using a logarithmically spaced triangular filter bank. Concretely, we process the audio at a sample rate of 44 100 Hz using the STFT with a frame size of 8192 and a hop size of 4410. Then, we apply to the magnitude of the STFT a triangular filter bank with 24 filters per octave between 65 Hz and 2 100 Hz. Finally, we take the logarithm of the resulting magnitudes to compress the input range.

Neural networks tend to produce over-confident predictions, which in further consequence could over-rule the predictions of a temporal model [9]. To mitigate this, we use two techniques: first, we train the model using uniform smoothing (i.e. we assign a proportion of $1 - \beta$ to other classes during training); second, during inference, we apply the *temperature softmax* function $\sigma_\tau(\mathbf{z})_j = e^{z_j/\tau} / \sum_{k=1}^K e^{z_k/\tau}$ instead of the standard softmax in the final layer. Higher values of τ produce smoother probability distributions. In this paper, we use $\beta = 0.9$ and $\tau = 1.3$, as determined in preliminary experiments.

3.2 Language Model

The language model P_L predicts the next chord, regardless of its duration, given the chord sequence it has previously seen. As shown in [21], RNN-based models perform better than n-gram models at this task. We thus adopt this approach, and refer the reader to [21] for details.

To give an overview, we follow the set-up introduced by [28] and use a recurrent neural network for next-chord prediction. The network’s task is to compute a probability distribution over all possible next chord symbols, given the chord symbols it has observed before. Figure 3 shows an RNN in a general next-step prediction task. In our case, the inputs z_k are the chord symbols given by $\mathcal{C}(y_{1:T})$.

We will describe in detail the network’s hyper-parameters in Section 4, where we will also evaluate the

effect the language models have on chord recognition.

3.3 Duration Model

The duration model P_D predicts whether the chord will change in the next time step. This corresponds to modelling the duration of chords. Existing temporal models induce implicit duration models: for example, an HMM implies an exponential chord duration distribution (if one state is used to model a chord), or a negative binomial distribution (if multiple left-to-right states are used per chord). However, such duration models are simplistic, static, and do not adapt to the processed piece.

An explicit duration model has been explored in [4], where beat-synchronised chord durations were stored as discrete distributions. Their approach is useful for beat-synchronised models, but impractical for frame-wise models—the probability tables would become too large, and data too sparse to estimate them. Since our approach avoids the potentially error-prone beat synchronisation, the approach of [4] does not work in our case.

Instead, we opt to use recurrent neural networks to model chord durations. These models are able to adapt to characteristics of the processed data [21], and have shown great potential in processing periodic signals [1] (and chords do change periodically within a piece). To train an RNN-based duration model, we set up a next-step-prediction task, identical in principle to the set-up for harmonic language modelling: the network has to compute the probability of a chord change in the next time step, given the chord changes it has seen in the past. We thus simplify $P_D(s_t | y_{1:t-1}) \hat{=} P_D(s_t | s_{1:t-1})$, as mentioned earlier. Again, see Fig. 3 for an overview (for duration modelling, replace z_k with s_t).

In Section 4, we will describe in detail the hyper-parameters of the networks we employed, and compare the properties of various settings to baseline duration models. We will also assess the impact on the duration modelling quality on the final chord recognition result.

3.4 Model Integration

Dynamic models such as RNNs have one main advantage over their static counter-parts (e.g. n-gram models for language modelling or HMMs for duration modelling): they consider all previous observations when predicting the next one. As a consequence, they are able to adapt to the piece that is currently processed—they assign higher probabilities to sub-sequences of chords they have seen earlier [21], or predict chord changes according to the harmonic rhythm of a song (see Sec. 4.3). The flip side of the coin is, however, that this property prohibits the use of dynamic programming approaches for efficient decoding. We cannot exactly and efficiently decode the best chord sequence given the input audio.

Hence we have to resort to approximate inference. In particular, we employ *hashed beam search* [32] to decode the chord sequence. General beam search restricts the search space by keeping only the N_b best solutions up to the current time step. (In our case, the N_b best paths through

all possible chord-time lattices, see Fig. 2.) However, as pointed out in [32], the beam might saturate with almost identical solutions, e.g. the same chord sequence differing only marginally in the times the chords change. Such pathological cases may impair the final estimate. To mitigate this problem, hashed beam search forces the tracked solutions to be diverse by pruning similar solutions with lower probability.

The similarity of solutions is determined by a task-specific *hash function*. For our purpose, we define the hash function of a solution to be the last N_h chord symbols in the sequence, regardless of their duration; formally, the hash function $f_h(y_{1:t}) = \bar{y}_{(k-N_h):k}$. (Recall that $\bar{y}_{1:k} = \mathcal{C}(y_{1:t})$.) In contrast to the hash function originally proposed in [32], which directly uses $y_{(t-N_h):t}$, our formulation ensures that sequences that differ only in timing, but not in chord sequence, are considered similar.

To summarise, we approximately decode the optimal chord transcription as defined in Eq. 1 using hashed beam search, which at each time step keeps the best N_b solutions, and at most N_s similar solutions.

4. EXPERIMENTS

In our experiments, we will first evaluate harmonic language and duration models individually. Here, we will compare the proposed models to common baselines. Then, we will integrate these models into the chord recognition framework we outlined in Section 2, and evaluate how the individual parts interact in terms of chord recognition score.

4.1 Data

We use the following datasets in 4-fold cross-validation. **Isophonics**³: 180 songs by the Beatles, 19 songs by Queen, and 18 songs by Zweieck, 10:21 hours of audio; **RWC Pop-ular** [15]: 100 songs in the style of American and Japanese pop music, 6:46 hours of audio; **Robbie Williams** [13]: 65 songs by Robbie Williams, 4:30 of audio; and **McGill Billboard** [3]: 742 songs sampled from the American billboard charts between 1958 and 1991, 44:42 hours of audio. The compound dataset thus comprises 1125 unique songs, and a total of 66:21 hours of audio.

Furthermore, we used the following data sets (with duplicate songs removed) as additional data for training the language and duration models: 173 songs from the **Rock** [11] corpus; a subset of 160 songs from **UsPop2002**⁴ for which chord annotations are available⁵; 291 songs from **Weimar Jazz**⁶, with chord annotations taken from lead sheets of Jazz standards; and **Jay Chou** [12], a small collection of 29 Chinese pop songs.

We focus on the major/minor chord vocabulary, and following [7], map all chords containing a minor third to minor, and all others to major. This leaves us with 25 classes: 12 root notes \times {major, minor} and the ‘no-chord’ class.

³ <http://isophonics.net/datasets>

⁴ <https://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html>

⁵ <https://github.com/tmc323/Chord-Annotations>

⁶ <http://jazzomat.hfm-weimar.de/dbformat/dboverview.html>

	<i>GRU-512</i>	<i>GRU-32</i>	<i>4-gram</i>	<i>2-gram</i>
log-P	-1.293	-1.576	-1.887	-2.393

Table 1. Language model results: average log-probability of the correct next chord computed by each model.

4.2 Language Models

The performance of neural networks depends on a good choice of hyper-parameters, such as number of layers, number of units per layer, or unit type (e.g. vanilla RNN, gated recurrent unit (GRU) [5] or long short-term memory unit (LSTM) [17]). The findings in [21] provide a good starting point for choosing hyper-parameter settings that work well. However, we strive to find a simpler model to reduce the computational burden at test time. To this end, we perform a grid search in a restricted search space, using the validation score of the first fold. We search over the following settings: number of layers $\in \{1, 2, 3\}$, number of units $\in \{256, 512\}$, unit type $\in \{\text{GRU}, \text{LSTM}\}$, input embedding $\in \{\text{one-hot}, \mathbb{R}^8, \mathbb{R}^{16}, \mathbb{R}^{24}\}$, learning rate $\in \{0.001, 0.005\}$, and skip connections $\in \{\text{on}, \text{off}\}$. Other hyper-parameters were fixed for all trials: we train the networks for 100 epochs using stochastic gradient descent with mini-batches of size 4, employ the Adam update rule [20], and starting from epoch 50, linearly anneal the learning rate to 0.

To increase the diversity in the training data, we use two data augmentation techniques, applied each time we show a piece to the network. First, we randomly shift the key of the piece; the network can thus learn that harmonic relations are independent of the key, as in roman numeral analysis. Second, we select a sub-sequence of random length instead of the complete chord sequence; the network thus has to learn to cope with varying context sizes.

The best model turned out to be a single-layer network of 512 GRUs, with a learnable 16-dimensional input embedding and without skip connections, trained using a learning rate of 0.005⁷. We compare this model and a smaller, but otherwise identical RNN with 32 units, to two baselines: a 2-gram model, and a 4-gram model. Both can be used for chord recognition in a higher-order HMM [25]. We train the n -gram models using maximum likelihood estimation with Lidstone smoothing as described in [21], using the key-shift data augmentation technique (sub-sequence cropping is futile for finite context models). As evaluation measure, we use the average log-probability of predicting the correct next chord. Table 1 presents the test results. The GRU models predict chord sequences with much higher probability than the baselines.

When we look into the input embedding $\mathbf{v}(\cdot)$, which was learned by the RNN during training from a random initialisation, we observe an interesting positioning of the chord symbols (see Figure 4). We found that similar patterns develop for all 1-layer GRUs we tried, and these patterns are consistent for all folds we trained on. We observe i) that

⁷ Due to space constraints, we cannot present the complete grid search results.

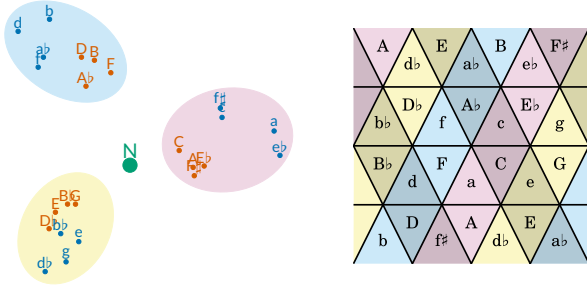


Figure 4. Chord embedding projected into 2D using PCA (left); Tonnetz of triads (right). The “no-chord” class resides in the center of the embedding. Major chords are upper-case and orange, minor chords lower-case and blue. Clusters in the projected embedding and the corresponding positions in the Tonnetz are marked in color. If projected into 3D (not shown here), the chord clusters split into a lower and upper half of four chords each. The chords in the lower halves are shaded in the Tonnetz representation.

chords form three clusters around the center, in which the minor chords are farther from the center than major chords; ii) that the clusters group major and minor chords with the same root, and the distance between the roots are minor thirds (e.g. C, E \flat , F \sharp , A); iii) that clockwise movement in the circle of fifths corresponds to clockwise movement in the projected embedding; and iv) that the way chords are grouped in the embedding corresponds to how they are connected in the Tonnetz.

At this time, we cannot provide an explanation for these automatically emerging patterns. However, they warrant a further investigation to uncover why this specific arrangement seems to benefit the predictions of the model.

4.3 Duration Models

As for the language model, we performed a grid search on the first fold to find good choices for the recurrent unit type $\in \{\text{vanilla RNN, GRU, LSTM}\}$, and number of recurrent units $\in \{16, 32, 64, 128, 256\}$ for the LSTM and GRU, and $\{128, 256, 512\}$ for the vanilla RNN. We use only one recurrent layer for simplicity. We found networks of 256 GRU units to perform best; although this indicates that even bigger models might give better results, for the purposes of this study, we think that this configuration is a good balance between prediction quality and model complexity.

The models were trained for 100 epochs using the Adam update rule [20] with a learning rate linearly decreasing from 0.001 to 0. The data was processed in mini-batches of 10, where the sequences were cut in excerpts of 200 time steps (20 s). We also applied gradient clipping at a value of 0.001 to ensure a smooth learning progress.

We compare the best RNN-based duration model with two baselines. The baselines are selected because both are implicit consequences of using HMMs as temporal model, as it is common in chord recognition. We assume a single parametrisation for each chord; this ostensible simplification is justified, because simple temporal models such as HMMs do not profit from chord information, as shown

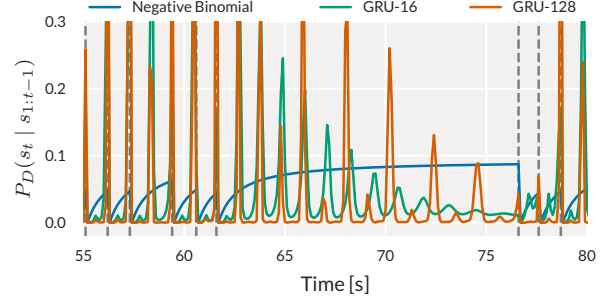


Figure 5. Probability of chord change computed by different models. Gray vertical dashed lines indicate true chord changes.

	GRU-256	GRU-16	Neg. Binom.	Exp.
log-P	-2.014	-2.868	-3.946	-4.003

Table 2. Duration model results: average log-probability of chord durations computed by each model.

by [4, 7]. The first baseline we consider is a *negative binomial distribution*. It can be modelled by a HMM using n states per chord, connected in a left-to-right manner, with transitions of probability p between the states (self-transitions thus have probability $1 - p$). The second, a special case of the first with $n = 1$, is an *exponential distribution*; this is the implicit duration distribution used by all chord recognition models that employ a simple 1-state-per-chord HMM as temporal model. Both baselines are trained using maximum likelihood estimation.

To measure the quality of a duration model, we consider the average log-probability it assigns to a chord duration. The results are shown in Table 2. We further added results for the simplest GRU model we tried—using only 16 recurrent units—to indicate the performance of small models of this type. We will also use this simple model when judging the effect of duration modelling on the final result in Sec. 4.4. As seen in the table, both GRU models clearly out-perform the baselines.

Figure 5 shows the reason why the GRU performs so much better than the baselines: as a dynamic model, it can adapt to the harmonic rhythm of a piece, while static models are not capable of doing so. We see that a GRU with 128 units predicts chord changes with high probability at periods of the harmonic rhythm. It also reliably remembers the period over large gaps in which the chord did not change (between seconds 61 and 76). During this time, the peaks decay differently for different multiples of the period, which indicates that the network simultaneously tracks multiple periods of varying importance. In contrast, the negative binomial distribution statically yields a higher chord change probability that rises with the number of audio frames since the last chord change. Finally, the smaller GRU model with only 16 units also manages to adapt to the harmonic rhythm; however, its predictions between the peaks are noisier, and it fails to remember the period correctly in the time without chord changes.

<i>Model</i>	<i>Root</i>	<i>Maj/Min</i>	<i>Seg.</i>
2-gram / neg. binom.	0.812	0.795	0.804
GRU-512 / GRU-256	0.821	0.805	0.814

Table 3. Results of the standard model (2-gram language model with negative binomial durations) compared to the best one (GRU language and duration models).

4.4 Integrated Models

The individual results for the language and duration models are encouraging, but only meaningful if they translate to better chord recognition scores. This section will thus evaluate if and how the duration and language models affect the performance of a chord recognition system.

The acoustic model used in these experiments was trained for 300 epochs (with 200 parameter updates per epoch) using a mini-batch size of 512 and the Adam update rule with standard parameters. We linearly decay the learning rate to 0 in the last 100 epochs.

We compare all combinations of language and duration models presented in the previous sections. For language modelling, these are the GRU-512, GRU-32, 4-gram, and 2-gram models; for duration modelling, these are the GRU-256, GRU-16, and negative binomial models. (We leave out the exponential model, because its results differ negligibly from the negative binomial one). The models are decoded using the Hashed Beam Search algorithm, as described in Sec. 3.4: we use a beam width of $N_b = 25$, where we track at most $N_s = 4$ similar solutions as defined by the hash function f_h , where the number of chords considered is set to $N_h = 5$. These values were determined by a small number of preliminary experiments.

Additionally, we evaluate exact decoding results for the n-gram language models in combination with the negative binomial duration distribution. This will indicate how much the results suffer due to the approximate beam search.

As main evaluation metric, we use the weighted chord symbol recall (WCSR) over the major/minor chord alphabet, as defined in [30]. We thus compute $WCSR = t_c/t_a$, where t_c is the total duration of chord segments that have been recognised correctly, and t_a is the total duration of chord segments annotated with chords from the target alphabet. We also report chord root accuracy and a measure of segmentation (see [16], Sec. 8.3). Table 3 compares the results of the standard model (the combination that implicitly emerges in simple HMM-based temporal models) to the best model found in this study. Although the improvements are modest, they are consistent, as shown by a paired t-test ($p < 2.487 \times 10^{-23}$ for all differences).

Figure 6 presents the effects of duration and language models on the WCSR. Better language and duration models directly improve chord recognition results, as the WCSR increases linearly with higher log-probability of each model. As this relationship does not seem to flatten out, further improvement of each model type can still increase the score. We also observe that the approximate beam search does not impair the result by much compared to exact decoding (compare the dotted blue line with the solid one).

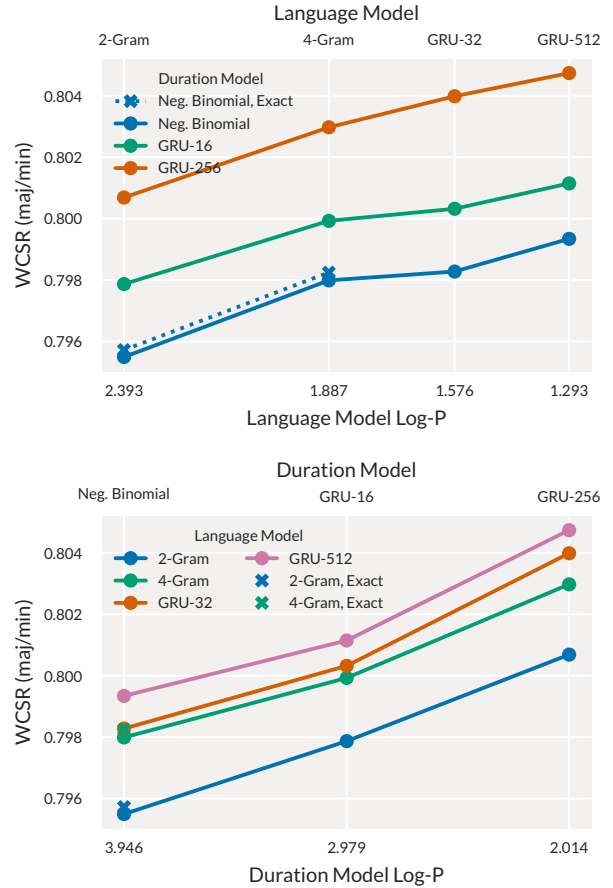


Figure 6. Effect of language and duration models on the final result. Both plots show the same results from different perspectives.

5. CONCLUSION AND DISCUSSION

We described a probabilistic model that disentangles three components of a chord recognition system: the acoustic model, the duration model, and the language model. We then developed better duration and language models than have been used for chord recognition, and illustrated why the RNN-based duration models perform better and are more meaningful than their static counterparts implicitly employed in HMMs. (For a similar investigation for chord language models, see [21].) Finally, we showed that improvements in each of these models directly influence chord recognition results.

We hope that our contribution facilitates further research in harmonic language and duration models for chord recognition. These aspects have been neglected because they did not show great potential for improving the final result [4, 7]. However, we believe (see [24] for some evidence) that this was due to the improper assumption that temporal models applied on the time-frame level can appropriately model musical knowledge. The results in this paper indicate that chord transitions modelled on the chord level, and connected to audio frames via strong duration models, indeed have the capability to improve chord recognition results.

6. ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC) under the EU's Horizon 2020 Framework Programme (ERC Grant Agreement number 670035, project "Con Espressione").

7. REFERENCES

- [1] Sebastian Böck and Markus Schedl. Enhanced Beat Tracking With Context-Aware Neural Networks. In *14th International Conference on Digital Audio Effects (DAFx-11)*, Paris, France, September 2011.
- [2] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio Chord Recognition With Recurrent Neural Networks. In *14th International Society for Music Information Retrieval Conference (ISMIR)*, Curitiba, Brazil, November 2013.
- [3] John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis. In *12th International Society for Music Information Retrieval Conference (ISMIR)*, Miami, USA, October 2011.
- [4] Ruofeng Chen, Weibin Shen, Ajay Srinivasamurthy, and Parag Chordia. Chord Recognition Using Duration-Explicit Hidden Markov Models. In *13th International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, October 2012.
- [5] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, *arXiv:1409.1259*, Doha, Qatar, October 2014.
- [6] Taemin Cho. *Improved Techniques for Automatic Chord Recognition from Music Audio Signals*. Dissertation, New York University, New York, 2014.
- [7] Taemin Cho and Juan P. Bello. On the Relative Importance of Individual Components of Chord Recognition Systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(2):477–492, February 2014.
- [8] Taemin Cho, Ron J Weiss, and Juan Pablo Bello. Exploring Common Variations in State Of The Art Chord Recognition Systems. In *Proceedings of the Sound and Music Computing Conference (SMC)*, Barcelona, Spain, July 2010.
- [9] Jan Chorowski and Navdeep Jaitly. Towards better decoding and language model integration in sequence to sequence models. *arXiv:1612.02695*, December 2016.
- [10] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *International Conference on Learning Representations (ICLR)*, *arXiv:1511.07289*, San Juan, Puerto Rico, February 2016.
- [11] Trevor de Clercq and David Temperley. A corpus analysis of rock harmony. *Popular Music*, 30(01):47–70, January 2011.
- [12] Junqi Deng and Yu-Kwong Kwok. Automatic Chord estimation on seventhsbass Chord vocabulary using deep neural network. In *2016 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, Shanghai, China, March 2016.
- [13] Bruno Di Giorgi, Massimiliano Zanoni, Augusto Sarti, and Stefano Tubaro. Automatic chord recognition based on the probabilistic modeling of diatonic modal harmony. In *Proceedings of the 8th International Workshop on Multidimensional Systems*, Erlangen, Germany, September 2013.
- [14] Takuya Fujishima. Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music. In *Proceedings of the International Computer Music Conference (ICMC)*, Beijing, China, October 1999.
- [15] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC Music Database: Popular, Classical and Jazz Music Databases. In *3rd International Conference on Music Information Retrieval (ISMIR)*, Paris, France, 2002.
- [16] Christopher Harte. *Towards Automatic Extraction of Harmony Information from Music Signals*. Dissertation, Department of Electronic Engineering, Queen Mary, University of London, London, United Kingdom, 2010.
- [17] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.
- [18] Eric J. Humphrey and Juan P. Bello. Rethinking Automatic Chord Recognition with Convolutional Neural Networks. In *11th International Conference on Machine Learning and Applications (ICMLA)*, Boca Raton, USA, December 2012.
- [19] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167*, March 2015.
- [20] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, *arXiv:1412.6980*, San Diego, USA, May 2015.
- [21] Filip Korzeniowski, David R. W. Sears, and Gerhard Widmer. A Large-Scale Study of Language Models for Chord Prediction. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, April 2018.

- [22] Filip Korzeniowski and Gerhard Widmer. Feature Learning for Chord Recognition: The Deep Chroma Extractor. In *17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, USA, August 2016.
- [23] Filip Korzeniowski and Gerhard Widmer. A Fully Convolutional Deep Auditory Model for Musical Chord Recognition. In *26th IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Salerno, Italy, September 2016.
- [24] Filip Korzeniowski and Gerhard Widmer. On the Futility of Learning Complex Frame-Level Language Models for Chord Recognition. In *Proceedings of the AES International Conference on Semantic Audio*, Erlangen, Germany, June 2017.
- [25] Filip Korzeniowski and Gerhard Widmer. Automatic Chord Recognition with Higher-Order Harmonic Language Modelling. In *26th European Signal Processing Conference (EUSIPCO)*, Rome, Italy, September 2018.
- [26] M. Mauch and S. Dixon. Simultaneous Estimation of Chords and Musical Context From Audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1280–1289, August 2010.
- [27] Brian McFee and Juan Pablo Bello. Structured Training for Large-Vocabulary Chord Recognition. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, October 2017.
- [28] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010*, pages 1045–1048, Chiba, Japan, September 2010.
- [29] Meinard Müller, Sebastian Ewert, and Sebastian Kreuzer. Making chroma features more robust to timbre changes. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Taipei, Taiwan, April 2009.
- [30] Johan Pauwels and Geoffroy Peeters. Evaluating automatically estimated chord sequences. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013.
- [31] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco. Connectionist Probability Estimators in HMM Speech Recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):161–174, January 1994.
- [32] Siddharth Sigtia, Nicolas Boulanger-Lewandowski, and Simon Dixon. Audio Chord Recognition With A Hybrid Recurrent Neural Network. In *16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, October 2015.
- [33] Yushi Ueda, Yuki Uchiyama, Takuya Nishimoto, Nobutaka Ono, and Shigeki Sagayama. HMM-based Approach for Automatic Chord Detection Using Refined Acoustic Features. In *2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, Dallas, USA, March 2010.

USING MUSICAL RELATIONSHIPS BETWEEN CHORD LABELS IN AUTOMATIC CHORD EXTRACTION TASKS

Tristan Carsault¹

Jérôme Nika^{2,1}

Philippe Esling¹

¹Ircam, CNRS, Sorbonne Université, UMR 9912 STMS, ²L3i Lab, University of La Rochelle
{carsault, jnika, esling}@ircam.fr

ABSTRACT

Recent research on Automatic Chord Extraction (ACE) has focused on the improvement of models based on machine learning. However, most models still fail to take into account the prior knowledge underlying the labeling alphabets (chord labels). Furthermore, recent works have shown that ACE performances have reached a glass ceiling. Therefore, this prompts the need to focus on other aspects of the task, such as the introduction of musical knowledge in the representation, the improvement of the models towards more complex chord alphabets and the development of more adapted evaluation methods.

In this paper, we propose to exploit specific properties and relationships between chord labels in order to improve the learning of statistical ACE models. Hence, we analyze the interdependence of the representations of chords and their associated distances, the precision of the chord alphabets, and the impact of performing alphabet reduction before or after training the model. Furthermore, we propose new training losses based on musical theory. We show that these improve the results of ACE systems based on Convolutional Neural Networks. By analyzing our results, we uncover a set of related insights on ACE tasks based on statistical models, and also formalize the musical meaning of some classification errors.

1. INTRODUCTION

Automatic Chord Extraction (ACE) is a topic that has been widely studied by the Music Information Retrieval (MIR) community over the past years. However, recent results seem to indicate that the rate of improvement of ACE performances has diminished over the past years [20].

Recently, a part of the MIR community pointed out the need to rethink the experimental methodologies. Indeed, current evaluation methods do not account for the intrinsic relationships between different chords [10]. Our work is built on these questions and is aimed to give some insights on the impact of introducing musical relationships between chord labels in the development of ACE methods.

Most ACE systems are built on the idea of extracting features from the raw audio signal and then using these features to construct a chord classifier [4]. The two major families of approaches that can be found in previous research are *rule-based* and *statistical* models. On one hand, the rule-based models rely on music-theoretic rules to extract information from the precomputed features. Although this approach is theoretically sound, it usually remains brittle to perturbations in the spectral distributions from which the features were extracted. On the other hand, statistical models rely on the optimization of a loss function over an annotated dataset. However, the generalization capabilities of these models are highly correlated to the size and completeness of their training set. Furthermore, most training methods see musical chords as independent labels and do not take into account the inherent relations between chords.

In this paper, we aim to target this gap by introducing musical information directly in the training process of statistical models. To do so, we propose to use prior knowledge underlying the labeling alphabets in order to account for the inherent relationships between chords directly inside the loss function of learning methods. Due to the complexity of the ACE task and the wealth of models available, we choose to rely on a single Convolutional Neural Network (CNN) architecture, which provides the current best results in ACE [19]. First, we study the impact of chord alphabets and their relationships by introducing a specific *hierarchy* of alphabets. We show that some of the reductions proposed by previous researches might be inadequate for learning algorithms. We also show that relying on more finely defined and extensive alphabets allows to grasp more interesting insights on the errors made by ACE systems, even though their accuracy is only marginally better or worse. Then, we introduce two novel chord distances based on musical relationships found in the *Tonnetz-space* or directly between chord components through their categorical differences. These distances can be used to define novel loss functions for learning algorithms. We show that these new loss functions improve ACE results with CNNs. Finally, we perform an extensive analysis of our approach and extract insights on the methodology required for ACE. To do so, we develop a specifically-tailored analyzer that focuses on the functional relations between chords to distinguish *strong* and *weak* errors. This analyzer is intended to be used for future ACE research to develop a finer understanding on the reasons behind the success or failure of ACE systems.



© Tristan Carsault, Jérôme Nika, Philippe Esling. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Tristan Carsault, Jérôme Nika, Philippe Esling. “Using musical relationships between chord labels in Automatic Chord Extraction tasks”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

2. RELATED WORKS

Automatic Chord Extraction (ACE) is defined as the task of labeling each segment of an audio signal using an alphabet of musical chords. In this task, chords are seen as the concomitant or successive combination of different notes played by one or many instruments.

2.1 Considerations on the ACE task

Whereas most MIR tasks have benefited continuously from the recent advances in deep learning, the ACE field seems to have reached a glass ceiling. In 2015, Humphrey and Bello [10] highlighted the need to rethink the whole ACE methodology by giving four insights on the task.

First, several songs from the reference annotated chord datasets (Isophonics, RWC-Pop, McGill Billboard) are not always tuned to 440Hz and may vary up to a quarter-tone. This leads to multiple misclassifications on the concomitant semi-tones. Moreover, chord labels are not always well suited to describe every song in these datasets.

Second, the chord labels are related and some subsets of those have hierarchical organizations. Therefore, the one-to-K assessment where all errors are equivalently weighted appears widely incorrect. For instance, the misclassification of a $C:Maj$ as a $A:min$ or $C\#:Maj$, will be considered equivalently wrong. However, $C:Maj$ and $A:min$ share two pitches in common whereas $C:Maj$ and $C\#:Maj$ have totally different pitch vectors.

Third, the very definition of the ACE task is also not entirely clear. Indeed, there is a frequent confusion between two different tasks. First, the literal *recognition* of a local audio segment using a chord label and its precise extensions, and, second, the *transcription of an underlying harmony*, taking into account the functional aspect of the chords and the long-term structure of the song. Finally, the labeling process involves the subjectivity of the annotators. For instance, even for expert annotators, it is hard to agree on possible chord inversions.

Therefore, this prompts the need to focus on other aspects such as the introduction of musical knowledge in the representation of chords, the improvement of the models towards more complex chord alphabets and the development of more adapted evaluation methods.

2.2 Workflow of ACE systems

Due to the complexity of the task, ACE systems are usually divided into four main modules performing *feature extraction*, *pre-filtering*, *pattern matching* and *post-filtering* [4].

First, the *pre-filtering* usually applies low-pass filters or harmonic-percussive source separation methods on the raw signal [12, 26]. This optional step allows to remove noise or other percussive information that are irrelevant for the chord extraction task. Then, the audio signal is transformed into a time-frequency representation such as the Short-Time Fourier Transform (STFT) or the Constant-Q Transform (CQT) that provides a logarithmically-scaled frequencies. These representations are sometimes summarized in a pitch class vector called *chromagram*. Then, suc-

cessive time frames of the spectral transform are averaged in context windows. This allows to smooth the extracted features and account for the fact that chords are longer-scale events. It has been shown that this could be done efficiently by feeding STFT context windows to a CNN in order to obtain a clean chromagram [13].

Then, these extracted features are classified by relying on either a rule-based chord template system or a statistical model. Rule-based methods give fast results and a decent level of accuracy [21]. With these methods, the extracted features are classified using a fixed dictionary of chord profiles [2] or a collection of decision trees [12]. However, these methods are usually brittle to perturbations in the input spectral distribution and do not generalize well.

Statistical models aim to extract the relations between precomputed features and chord labels based on a training dataset in which each temporal frame is associated to a label. The optimization of this model is then performed by using gradient descent algorithms to find an adequate configuration of its parameters. Several probabilistic models have obtained good performances in ACE, such as multivariate Gaussian Mixture Model [3] and convolutional [9, 14] or recurrent [1, 25] Neural Networks.

Finally, *post-filtering* is applied to smooth out the classified time frames. This is usually based on a study of the transition probabilities between chords by a Hidden Markov Model (HMM) optimized with the Viterbi algorithm [17] or with Conditional Random Fields [15].

2.3 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a statistical model composed of layers of artificial neurons that transform the input by repeatedly applying convolution and pooling operations. A convolutional layer is characterized by a set of convolution kernels that are applied in parallel to the inputs to produce a set of output *feature maps*. The convolution kernels are defined as three-dimensional tensors $h \in \mathbb{R}^{M \times U \times V}$ where M is the number of kernels, U is the height and V the width of each kernel. If we note the input as matrix X , then the output feature maps are defined by $Y = X * h_m$ for every kernels, where $*$ is a 2D discrete convolution operation

$$(A * B)_{i,j} = \sum_{r=0}^{(T-1)} \sum_{s=0}^{(F-1)} A_{r,s} B_{i-r,j-s} \quad (1)$$

for $A \in \mathbb{R}^{T \times F}$ and $B \in \mathbb{R}^{U \times V}$ with $0 \leq i \leq T+U-1$ and $0 \leq j \leq F+V-1$.

As this convolutional layer significantly increases the dimensionality of the input data, a pooling layer is used to reduce the size of the feature maps. The pooling operation reduces the maps by computing local mean, maximum or average of sliding context windows across the maps. Therefore, the overall structure of a CNN usually consists in alternating convolution, activation and pooling layers. Finally, in order to perform classification, this architecture

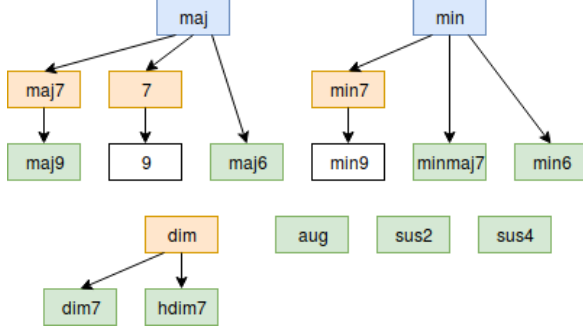


Figure 1. Hierarchy of the chord alphabets (blue: A_0 , orange: A_1 , green: A_2)

is typically followed by one or many fully-connected layers. Thus, the last layer produces a probability vector of the same size as the chord alphabet. As we will rely on the architecture defined by [9], we redirect interested readers to this paper for more information.

3. OUR PROPOSAL

3.1 Definition of alphabets

Chord annotations from reference datasets are very precise and include extra notes (in parenthesis) and basses (after the slash) [7]. With this notation, we would obtain over a thousand chord classes with very sparse distributions. However, we do not use these extra notes and bass in our classification. Therefore, we can remove this information

$$F : \text{maj7}(11)/3 \rightarrow F : \text{maj7} \quad (2)$$

Even with this reduction, the number of chord qualities (eg. *maj7*, *min*, *dim*) is extensive and we usually do not aim for such a degree of precision. Thus, we propose three alphabets named A_0 , A_1 and A_2 with a controlled number of chord qualities. The level of precision of the three alphabets increases gradually (see Figure 1). In order to reduce the number of chord qualities, each one is mapped to a parent class when it exists, otherwise to the *no-chord* class N .

The first alphabet A_0 contains all the major and minor chords, which defines a total of 25 classes

$$A_0 = \{N\} \cup \{P \times \text{maj}, \text{min}\} \quad (3)$$

where P represents the 12 pitch classes.

Here, we consider the interest of working with chord alphabets larger than A_0 . Therefore, we propose an alphabet containing all chords present in the harmonization of the major scale (usual notation of harmony in jazz music). This corresponds to the orange chord qualities and their parents in Figure 1. The chord qualities without heritage are included in the *no-chord* class N , leading to 73 classes

$$A_1 = \{N\} \cup \{P \times \text{maj}, \text{min}, \text{dim}, \text{maj7}, \text{min7}, 7\} \quad (4)$$

Finally, the alphabet A_2 is inspired from the large vocabulary alphabet proposed by [19]. This most complete chord alphabet contains 14 chord qualities and 169 classes

$$A_2 = \{N\} \cup \{P \times \text{maj}, \text{min}, \text{dim}, \text{aug}, \text{maj6}, \text{min6}, \text{maj7}, \text{minmaj7}, \text{min7}, 7, \text{dim7}, \text{hdim7}, \text{sus2}, \text{sus4}\} \quad (5)$$

3.2 Definition of chord distances

In most CNN approaches, the model does not take into account the nature of each class when computing their differences. Therefore, this distance which we called categorical distance D_0 is the binary indicator

$$D_0(\text{chord}_1, \text{chord}_2) = \begin{cases} 0 & \text{if } \text{chord}_1 = \text{chord}_2 \\ 1 & \text{if } \text{chord}_1 \neq \text{chord}_2 \end{cases} \quad (6)$$

However, we want here to include the relationships between chords directly in our model. For instance, a $C:\text{maj7}$ is closer to an $A:\text{min7}$ than a $C\#:\text{maj7}$. Therefore, we introduce more refined distances that can be used to define the loss function for learning.

Here, we introduce two novel distances that rely on the representation of chords in an harmonic space or in a pitch space to provide a finer description of the chord labels. However, any other distance that measure similarities between chords could be studied [8, 18].

3.2.1 Tonnetz distance

A *Tonnetz-space* is a geometric representation of the tonal space based on harmonic relationships between chords. We chose a Tonnetz-space generated by three transformations of the major and minor triads [5] changing only one of the three notes of the chords: the *relative transformation* (transforms a chord into his relative major / minor), the *parallel transformation* (same root but major instead of minor or conversely), the *leading-tone exchange* (in a major chord the root moves down by a semitone, in a minor chord the fifth moves up by a semitone). Representing chords in this space has already shown promising results for classification on the A_0 alphabet [11].

We define the cost of a path between two chords as the sum of the successive transformations. Each transformation is associated to the same cost. Furthermore, an extra cost is added if the chords have been reduced beforehand in order to fit the alphabet A_0 . Then, our distance D_1 is:

$$D_1(\text{chord}_1, \text{chord}_2) = \min(C) \quad (7)$$

with C the set of all possible path costs from chord_1 to chord_2 using a combination of the three transformations.

3.2.2 Euclidean distance on pitch class vectors

In some works, pitch class vectors are used as an intermediate representation for ACE tasks [16]. Here, we use these pitch class profiles to calculate the distances between chords according to their harmonic content.

Each chord from the dictionary is associated to a 12-dimensional binary pitch vector with 1 if the pitch is present in the chord and 0 otherwise (for instance $C:\text{maj7}$

becomes $(1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1)$. The distance between two chords is defined as the Euclidean distance between the two binary pitch vectors.

$$D_2(chord_1, chord_2) = \sqrt{\sum_{i=0}^{11} (chord_1^i - chord_2^i)^2} \quad (8)$$

Hence, this distance allows to account for the number of pitches that are shared by two chords.

The D_0 , D_1 or D_2 distance is used to define the loss function for training the CNN classification model.

3.3 Introducing the relations between chords

To train the model with our distances, we first reduce the original labels from the Isophonics dataset¹ so that they fit one of our three alphabets A_0 , A_1 , A_2 . Then, we denote y_{true} as the one-hot vector where each bin corresponds to a chord label in the chosen alphabet A_i . The output of the model, noted y_{pred} , is a vector of probabilities over all the chords in a given alphabet A_i . In the case of D_0 , we train the model with a loss function that simply compares y_{pred} to the original label y_{true} . However, for our proposed distances (D_1 and D_2), we introduce a similarity matrix M that associates each couple of chords to a similarity ratio.

$$M_{i,j} = \frac{1}{D_k(chord_i, chord_j) + K} \quad (9)$$

K is an arbitrary constant to avoid division by zero. The matrix M is symmetric and we normalize it with its maximum value to obtain \bar{M} . Afterwards, we define a new y_{true} which is the matrix multiplication of the old y_{true} and the normalized matrix \bar{M} .

$$y_{true}^- = y_{true} \bar{M} \quad (10)$$

Finally, the loss function for D_1 and D_2 is defined by a comparison between this new ground truth y_{true}^- and the output y_{pred} . Hence, this loss function can be seen as a weighted multi-label classification.

4. EXPERIMENTS

4.1 Dataset

We perform our experiments on the *Beatles* dataset as it provides the highest confidence regarding the ground truth annotations [6]. This dataset is composed by 180 songs annotated by hand. For each song, we compute the CQT by using a window size of 4096 samples and a hop size of 2048. The transform is mapped to a scale of 3 bins per semi-tone over 6 octaves ranging from C1 to C7. We augment the available data by performing all transpositions from -6 to +6 semi-tones and modifying the labels accordingly. Finally, to evaluate our models, we split the data into a training (60%), validation (20%) and test (20%) sets.

¹ <http://isophonics.net/content/reference-annotations-beatles>

4.2 Models

We use the same CNN model for all test configurations, but change the size of the last layer to fit the size of the selected chord alphabet. We apply a batch normalization and a Gaussian noise addition on the inputs layer. The architecture of the CNN consists of three convolutional layers followed by two fully-connected layers. The architecture is very similar to the first CNN that has been proposed for the ACE task [9]. However, we add dropout between each convolution layer to prevent over-fitting.

For training, we use the ADAM optimizer with a learning rate of 2.10^{-5} for a total of 1000 epochs. We reduce the learning rate if the validation loss has not improved during 50 iterations. Early stopping is applied if the validation loss has not improved during 200 iterations and we keep the model with the best validation accuracy. For each configuration, we perform a 5-cross validation by repeating a random split of the dataset.

5. RESULTS AND DISCUSSION

The aim of this paper is not to obtain the best classification scores (which would involve pre- or post-filtering methods) but to study the impact on the classification results of different musical relationships (as detailed in the previous section). Therefore, we ran 9 instances of the CNN model corresponding to all combinations of the 3 alphabets A_0 , A_1 , A_2 and 3 distances D_0 , D_1 , D_2 to compare their results from both a *quantitative* and *qualitative* point of view. We analyzed the results using the *mireval* library [22] to compute classification scores, and a Python *ACE Analyzer* that we developed to reveal the musical meaning of classification errors and, therefore, understand their qualities.

5.1 Quantitative analysis: MIREX evaluation

Regarding the MIREX evaluation, the efficiency of ACE models is assessed through classification scores over different alphabets [22]. The MIREX alphabets for evaluation have a gradation of complexity from Major/Minor to Tetrads. In our case, for the evaluation on a specific alphabet, we apply a reduction from our training alphabet A_i to the MIREX evaluation alphabet. Here, we evaluate on three alphabet : Major/Minor, Sevenths, and Tetrads. These alphabets correspond roughly to our three alphabets (Major/Minor $\sim A_0$, Sevenths $\sim A_1$, Tetrads $\sim A_2$).

5.1.1 MIREX Major/minor

Figure 2 depicts the average classification scores over all frames of our test dataset for different distances and alphabets. We can see that the introduction of the D_1 or D_2 distance improves the classification compared to D_0 . With these distances, and even without pre- or post-filtering, we obtain classification scores that are superior to that of similar works (75.9% for CNN with post-filtering but an extended dataset in [10] versus 76.3% for $A_2 - D_1$). Second, the impact of working first on large alphabets (A_1 and A_2), and then reducing on A_0 for the test is negligible on Maj/Min (only from a quantitative point of view, see 5.2).

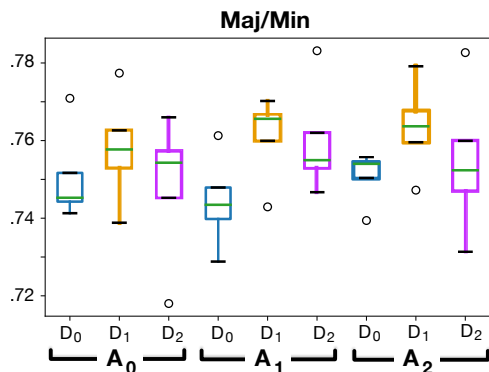


Figure 2. Results of the 5-folds: evaluation on MIREX Maj/Min (\sim reduction on A_0).

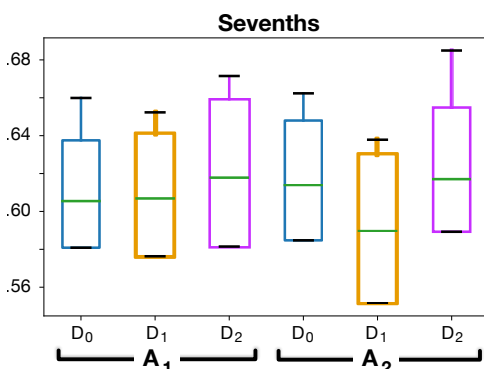


Figure 3. Results of the 5-folds: evaluation on MIREX Sevenths (\sim reduction on A_1).

5.1.2 MIREX Sevenths

With more complex alphabets, the classification score is lower than for MIREX Maj/Min. This result is not surprising since we observe this behavior on all ACE systems. Moreover, the models give similar results and we can not observe a particular trend between the alphabet reductions or the different distances. The same result is observed for the evaluation with MIREX tetrads (\sim reduction on A_2). Nonetheless, the MIREX evaluation uses a binary score to compare chords. Because of this approach, the qualities of the classification errors cannot be evaluated.

5.2 Qualitative analysis: understanding the errors

In this section, we propose to analyze ACE results from a qualitative point of view. The aim here is not to introduce new alphabets or distances in the models, but to introduce a new type of evaluation of the results. Our goal is twofold: to understand what causes the errors in the first place, and to distinguish “weak” from “strong” errors with a *functional* approach.

In tonal music, the *harmonic functions* qualify the roles and the tonal significances of chords, and the possible equivalences between them within a sequence [23, 24]. Therefore, we developed an *ACE Analyzer* including two modules discovering some formal musical relationships

Model	Tot.	\subset Maj	\subset min
A_0-D_0	34.93	-	-
A_0-D_1	36.12	-	-
A_0-D_2	35.37	-	-
A_1-D_0	52.40	23.82	4.37
A_1-D_1	57.67	28.31	5.37
A_1-D_2	55.17	25.70	4.21
A_2-D_0	55.28	26.51	4.29
A_2-D_1	60.47	31.61	6.16
A_2-D_2	55.45	25.74	4.78

Table 1. Left: total percentage of errors corresponding to inclusions or chords substitutions rules, right: percentage of errors with inclusion in the correct triad (% of the total number of errors).

Model	rel. M	rel. m	T subs. 2	$m \rightarrow M$	$M \rightarrow m$
A_0-D_0	4.19	5.15	2.37	7.26	12.9
A_0-D_1	4.40	5.20	2.47	7.66	13.4
A_0-D_2	5.13	4.87	2.26	8.89	10.89
A_1-D_0	2.63	3.93	1.53	4.46	8.83
A_1-D_1	3.05	3.36	1.58	5.53	7.52
A_1-D_2	3.02	4.00	1.62	5.84	8.07
A_2-D_0	2.54	4.15	1.51	4.96	8.54
A_2-D_1	2.79	2.97	1.54	5.29	7.46
A_2-D_2	3.11	4.26	1.63	5.34	7.59

Table 2. Left: percentage of errors corresponding to usual chords substitutions rules, right: percentage of errors “major instead of minor” or inversely (% of the total number of errors).

between the target chords and the chords predicted by ACE models. Both modules are generic and independent of the classification model, and are available online.²

5.2.1 Substitution rules

The first module detects the errors corresponding to hierarchical relationships or usual *chord substitutions* rules: using a chord in place of another in a chord progression (usually substituted chords have two pitches in common with the triad that they are replacing).

Table 1 presents: *Tot.*, the total fraction of errors that can be explained by the whole set of substitution rules we implemented, and \subset *Maj* and \subset *min*, the errors included in the correct triad (e.g. *C:maj* instead of *C:maj7*, *C:min7* instead of *C:min*). Table 2 presents the percentages of errors corresponding to widely used substitution rules: *rel. m* and *rel. M*, relative minor and major; *T subs. 2*, tonic substitution different from *rel. m* or *rel. M* (e.g. *E:min7* instead or *C:maj7*), and the percentages of errors $m \rightarrow M$ and $M \rightarrow m$ (same root but major instead of minor or conversely). The tables only show the categories representing more than 1% of the total number of errors, but other substitutions (that are not discussed here) were analyzed: tritone substitution, substitute dominant, and equivalence of *dim7* chords modulo inversions.

First, *Tot.* in Table 1 shows that a huge fraction of errors can be explained by usual substitution rules. This percent-

²http://repmus.ircam.fr/dyci2/ace_analyzer

Model	Non-diat. targ.	Non-diat. pred.
A_0-D_0	37.96	28.41
A_0-D_1	44.39	15.82
A_0-D_2	45.87	17.60
A_1-D_0	38.05	21.26
A_1-D_1	37.94	20.63
A_1-D_2	38.77	20.23
A_2-D_0	37.13	30.01
A_2-D_1	36.99	28.41
A_2-D_2	37.96	28.24

Table 3. Errors occurring when the target is non-diatonic (% of the total number of errors), non-diatonic prediction errors (% of the subset of errors on diatonic targets).

age can reach 60.47%, which means that numerous classification errors nevertheless give useful indications since they mistake a chord for another chord with an equivalent function. For instance, Table 2 shows that a significant amount of errors (up to 10%) are relative major / minor substitutions. Besides, for the three distances, the percentage in *Tot.* (Table 1) increases with the size of the alphabet: larger alphabets seem to imply weaker errors (higher amount of equivalent harmonic functions).

We can also note that numerous errors (between 28.19% and 37.77%) correspond to inclusions in major or minor chords ($\subset Maj$ and $\subset min$, Table 1) for A_1 and A_2 . In the framework of the discussion about *recognition* and *transcription* mentioned in introduction, this result questions the relevance of considering exhaustive extensions when the goal is to extract and formalize an underlying harmony.

Finally, for A_0 , A_1 , and A_2 , using D_1 instead of D_0 increases the fraction of errors attributed to categories in the left part of Table 2 (and in almost all the configurations when using D_2). This shows a qualitative improvement since all these operations are considered as valid chord substitutions. On the other hand, the impact on the (quite high) percentages in the right part of Table 2 is not clear. We can assume that temporal smoothing can be one of the keys to handle the errors $m \rightarrow M$ and $M \rightarrow m$.

5.2.2 Harmonic degrees

The second module of our *ACE Analyzer* focuses on *harmonic degrees*. First, by using the annotations of key in the dataset in addition to that of chords, this module determines the roman numerals characterizing the harmonic degrees of the predicted chord and of the target chord (e.g. in C, if a chord is an extension of C, I; if it is an extension of D: *min*, ii; etc.) when it is possible (e.g. in C, if a chord is an extension of C# it does not correspond to any degree). Then, it counts the errors corresponding to substitutions of harmonic degrees when it is possible (e.g. in C, $A: min$ instead of C corresponds to $I \sim vi$). This section shows an analysis of the results using this second module. First, it determines if the target chord is diatonic (i.e. belongs to the harmony of the key), as presented in Table 3. If this is the case, the notion of incorrect degree for the predicted chord is relevant and the percentages of errors corresponding to substitutions of degrees is computed (Table 4).

Model	$I \sim IV$	$I \sim V$	$IV \sim V$	$I \sim vi$	$IV \sim ii$	$I \sim iii$
A_0-D_0	17.41	14.04	4.54	4.22	5.41	2.13
A_0-D_1	17.02	13.67	3.33	4.08	6.51	3.49
A_0-D_2	16.16	13.60	3.08	5.65	6.25	3.66
A_1-D_0	17.53	13.72	3.67	5.25	4.65	3.50
A_1-D_1	15.88	13.82	3.48	4.95	6.26	3.46
A_1-D_2	16.73	13.45	3.36	4.70	5.75	2.97
A_2-D_0	16.90	13.51	3.68	4.45	5.06	3.32
A_2-D_1	16.81	13.60	3.85	4.57	5.37	3.59
A_2-D_2	16.78	12.96	3.84	5.19	7.01	3.45

Table 4. Errors ($> 2\%$) corresponding to degrees substitutions (% of the subset of errors on diatonic targets).

A first interesting fact presented in Table 3 is that 36.99% to 45.87% of the errors occur when the target chord is non-diatonic. It also shows, for the three alphabets, that using D_1 or D_2 instead of D_0 makes the fraction of non-diatonic errors decrease (Table 3, particularly A_0), which means that the errors are more likely to stay in the correct key. Surprisingly, high percentages of errors are associated to errors $I \sim V$ (up to 14.04%), $I \sim IV$ (up to 17.41%), or $IV \sim V$ (up to 4.54%) in Table 4. These errors are not usual substitutions, and $IV \sim V$ and $I \sim IV$ have respectively 0 and 1 pitch in common. In most of the cases, these percentages tend to decrease on alphabets A_1 or A_2 and when using musical distances (particularly D_2). Conversely, it increases the amount of errors in the right part of Table 4 containing usual substitutions: once again we observe that the more precise the musical representations are, the more the harmonic functions tend to be correct.

6. CONCLUSION

We presented a novel approach taking advantage of musical prior knowledge underlying the labeling alphabets into ACE statistical models. To this end, we applied reductions on different chord alphabets and we used different distances to train the same type of model. Then, we conducted a quantitative and qualitative analysis of the classification results.

First, we conclude that training the model using distances reflecting the relationships between chords improves the results both quantitatively (classification scores) and qualitatively (in terms of harmonic functions). Second, it appears that working first on large alphabets and reducing the chords during the test phase does not significantly improve the classification scores but provides a qualitative improvement in the type of errors. Finally, ACE could be improved by moving away from its binary classification paradigm. Indeed, MIREX evaluations focus on the nature of chords but a large amount of errors can be explained by inclusions or usual substitution rules. Our evaluation method therefore provides an interesting notion of *musical quality* of the errors, and encourages to adopt a functional approach or even to introduce a notion of equivalence classes. It could be adapted to the ACE problem downstream and upstream: in the classification processes as well as in the methodology for labeling the datasets.

7. ACKNOWLEDGMENTS

The authors would like to thank the master's students who contributed to the implementation: Alexis Font, Grégoire Locqueville, Octave Roulleau-Thery, and Téo Sanchez. This work was supported by the DYCI2 project ANR-14-CE2 4-0002-01 funded by the French National Research Agency (ANR), the MAKIMOno project 17-CE38-0015-01 funded by the French ANR and the Canadian Natural Sciences and Engineering Research Council (STPG 507004-17), the ACTOR Partnership funded by the Canadian Social Sciences and Humanities Research Council (895-2018-1023).

8. REFERENCES

- [1] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Audio chord recognition with recurrent neural networks. In *International Symposium on Music Information Retrieval*, pages 335–340, 2013.
- [2] C. Cannam, E. Benetos, M. Mauch, M. E. P. Davies, S. Dixon, C. Landone, K. Noland, and D. Stowell. Mirex 2015: Vamp plugins from the centre for digital music. In *Proceedings of the Music Information Retrieval Evaluation eXchange (MIREX)*, 2015.
- [3] T. Cho. *Improved techniques for automatic chord recognition from music audio signals*. PhD thesis, New York University, 2014.
- [4] T. Cho, R. J. Weiss, and J. P. Bello. Exploring common variations in state of the art chord recognition systems. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 1–8, 2010.
- [5] R. Cohn. Neo-riemannian operations, parsimonious tri-chords, and their "tonnetz" representations. *Journal of Music Theory*, 41(1):1–66, 1997.
- [6] C. Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, 2010.
- [7] C. Harte, M. B. Sandler, S. A. Abdallah, and E. Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *International Symposium on Music Information Retrieval*, volume 5, pages 66–71, 2005.
- [8] C-Z. A. Huang, D. Duvenaud, and K. Z. Gajos. Chordriple: Recommending chords to help novice composers go beyond the ordinary. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 241–250. ACM, 2016.
- [9] E. J. Humphrey and J. P. Bello. Rethinking automatic chord recognition with convolutional neural networks. In *Machine Learning and Applications (ICMLA)*, 2012 11th International Conference on, volume 2, pages 357–362. IEEE, 2012.
- [10] E. J. Humphrey and J. P. Bello. Four timely insights on automatic chord estimation. In *International Symposium on Music Information Retrieval*, pages 673–679, 2015.
- [11] E. J. Humphrey, T. Cho, and J. P. Bello. Learning a robust tonnetz-space transform for automatic chord recognition. In *Acoustics, Speech and Signal Processing (ICASSP)*, 2012 IEEE International Conference on, pages 453–456. IEEE, 2012.
- [12] J. Jiang, W. Li, and Y. Wu. Extended abstract for mirex 2017 submission: Chord recognition using random forest model. *MIREX evaluation results*, 2017.
- [13] F. Korzeniowski and G. Widmer. Feature learning for chord recognition: The deep chroma extractor. *arXiv preprint arXiv:1612.05065*, 2016.
- [14] F. Korzeniowski and G. Widmer. A fully convolutional deep auditory model for musical chord recognition. In *Machine Learning for Signal Processing (MLSP)*, 2016 IEEE 26th International Workshop on, pages 1–6. IEEE, 2016.
- [15] J. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.
- [16] K. Lee. Automatic chord recognition from audio using enhanced pitch class profile. In *International Computer Music Conference*, 2006.
- [17] H-L. Lou. Implementing the viterbi algorithm. *IEEE Signal Processing Magazine*, 12(5):42–52, 1995.
- [18] S. Madjiheurem, L. Qu, and C. Walder. Chord2vec: Learning musical chord embeddings. In *Proceedings of the Constructive Machine Learning Workshop at 30th Conference on Neural Information Processing Systems (NIPS2016)*, Barcelona, Spain, 2016.
- [19] B. McFee and J. P. Bello. Structured training for large-vocabulary chord recognition. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR2017)*. ISMIR, 2017.
- [20] M. McVicar, R. Santos-Rodríguez, Y. Ni, and T. De Bie. Automatic chord estimation from audio: A review of the state of the art. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(2):556–575, 2014.
- [21] L. Oudre, Y. Grenier, and C. Févotte. Template-based chord recognition: Influence of the chord types. In *International Symposium on Music Information Retrieval*, pages 153–158, 2009.
- [22] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. W. Ellis, and C. C. Raffel. mir_eval: A transparent implementation of common mir metrics. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, 2014.

- [23] A. Rehding. *Hugo Riemann and the birth of modern musical thought*, volume 11. Cambridge University Press, 2003.
- [24] A. Schoenberg and L. Stein. *Structural functions of harmony*. Number 478. WW Norton & Company, 1969.
- [25] Y. Wu, X. Feng, and W. Li. Mirex 2017 submission: Automatic audio chord recognition with miditrained deep feature and blstm-crf sequence decoding model. *MIREX evaluation results*, 2017.
- [26] X. Zhou and A. Lerch. Chord detection using deep learning. In *Proceedings of the 16th International Symposium on Music Information Retrieval Conference*, volume 53, 2015.

A PREDICTIVE MODEL FOR MUSIC BASED ON LEARNED INTERVAL REPRESENTATIONS

Stefan Lattner^{1,2}, Maarten Grachten^{1,2}, Gerhard Widmer¹

¹ Institute of Computational Perception, JKU Linz

² Sony Computer Science Laboratories (CSL), Paris, France

ABSTRACT

Connectionist sequence models (e.g., RNNs) applied to musical sequences suffer from two known problems: First, they have strictly “absolute pitch perception”. Therefore, they fail to generalize over musical concepts which are commonly perceived in terms of *relative distances* between pitches (e.g., melodies, scale types, modes, cadences, or chord types). Second, they fall short of capturing the concepts of repetition and musical form. In this paper we introduce the *recurrent gated autoencoder* (RGAE), a recurrent neural network which learns and operates on *interval representations* of musical sequences. The relative pitch modeling increases generalization and reduces sparsity in the input data. Furthermore, it can learn sequences of copy-and-shift operations (i.e. chromatically transposed copies of musical fragments)—a promising capability for learning musical repetition structure. We show that the RGAE improves the state of the art for general connectionist sequence models in learning to predict monophonic melodies, and that ensembles of relative and absolute music processing models improve the results appreciably. Furthermore, we show that the relative pitch processing of the RGAE naturally facilitates the learning and the generation of sequences of copy-and-shift operations, wherefore the RGAE greatly outperforms a common absolute pitch recurrent neural network on this task.

1. INTRODUCTION

The objective of sequence models for music prediction is to predict (the probability of) musical events at the next time step, given some prior musical context. In the (most common) case of predicting note events, this task involves finding relationships between past and future occurrences of absolute pitches. However, many music theoretical constructs that might help to find such relationships are defined in relative terms, such as diatonic scale steps, and cadences. The discrepancy between the relative nature of many regularities in music and the absolute pitch representation is problematic for modeling tasks, because it leads to

high sparsity in the input data, increased model sizes, and altogether reduced generalization in music modeling.

To remedy these problems, musical input sequences can be transposed to a common key before training, augmented by random transpositions during training, or, in case of symbolic monophonic music, transformed into interval representations before training. In this work, we propose a sequence model which *learns* both interval representations from absolute pitch sequences and temporal dependencies between these intervals. By learning not only the intervals between two successive notes, but all intervals within a window of n pitches, the model is more robust to diatonic transposition and can also learn repetition structure. More precisely, a recurrent neural network (RNN) is employed on top of a gated autoencoder (GAE), which we refer to as *recurrent gated autoencoder* (RGAE). The GAE portion learns the intervals between its input and its target pitches and represents them in its latent space. The RNN portion operates on these interval representations, to learn their temporal dependencies. The implicit transformation to intervals allows this architecture to operate directly on absolute musical textures, without the need for data pre-processing. Besides, relative pitch modeling reduces the sparsity in the data and the representations learned by the GAE are transposition-invariant. Therefore, the RGAE requires less temporal connections than a common RNN while achieving higher prediction accuracy.

Also, operating on the intervals of input sequences brings added value to sequence modeling. By allowing the model to relate its prediction with events using specific time lags, it can learn copy-and-shift operations. In the space of intervals, such operations are performed by repeatedly applying a constant interval to events occurring a constant time lag in the past. Moreover, the RNN portion of the architecture can learn sequences of such copy-and-shift operations (i.e., “structure schemes”), which can then be realized as musical notes by the GAE.

This ability is promising for music modeling, where musical form defines the self-similarity within a piece, and repeated sections often occur as a transposed (i.e., shifted in the pitch dimension) version of the initial section. Musical form is challenging to learn with common sequence models, like RNNs. They are specialized in learning the statistics of musical textures and are “blind” towards similarity and (transposed) repetition (i.e., there is no content-independent “repetition neuron”). As a result, when sampling music using such models, repeated fragments occur



© Stefan Lattner, Maarten Grachten, Gerhard Widmer. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Stefan Lattner, Maarten Grachten, Gerhard Widmer. “A predictive model for music based on learned interval representations”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

either due to chance or as a phenomenon of an entanglement with a learned texture. In contrast, the ability of RGAEs to learn copy-and-shift operations may allow to represent musical form explicitly, and to realize learned schemes as musical textures in music prediction and music generation tasks.

We show that the RGAE is competitive with state-of-the-art models in a music sequence learning task. Furthermore, we demonstrate that the RGAE, due to its relative pitch processing, is complementary to absolute pitch models, by combining their predictions to obtain improved accuracy. Lastly, we show that the RGAE is particularly suited for learning sequences of copy-and-shift operations. It can learn to recognize and continue pre-defined “structure schemes”, abstracted from the actual texture, with which the scheme is realized.

In Section 2, we provide an overview of related models and related publications. In Section 3, the GAE and the proposed extensions to the RGAE are described, as well as the baseline RNN used for comparison and combined prediction. General training details concerning the GAE are given in Section 4. The two experiments conducted, including the data used, training details and discussion for each experiment separately, are presented in Section 5. Section 6 concludes the paper and provides further directions.

2. RELATED WORK

GAEs are bi-linear models utilizing *multiplicative interactions* to learn correlations between or within data instances. They were introduced by [15] as a derivative of the gated Boltzmann machines (GBMs) [17, 18], as standard learning criteria became applicable through the development of denoising autoencoders [28]. In music, bi-linear models were applied to learn co-variances within spectrogram data for music similarity estimation [25], and for learning musical transformations in the symbolic domain [11].

The GAE was utilized for learning the derivatives of sequences in [16] (between subsequent frames in movies of rotated 3D objects), and to predict accelerated motion by stacking two layers to learn second-order derivatives [19]. This method is very similar to the one proposed here, but we use different dimensionalities between input and output, and we do not assume constant transformations but rather learn *sequences of transformations* using an RNN.

Probabilistic n-gram models, specialized on learning to predict monophonic pitch sequences include IDyOM [23], and [10], both employ multiple features of the musical surface. In this paper, we do not compare the RGAE with these models, as they are more specialized on the musical domain, by explicit selection of (computed) features. We compare the RGAE to the currently best performing general connectionist sequence model, the RTDRBM [1]. Its architecture is similar to the well-known RTRBM proposed in [27], but it employs a different cost function.

For structured sequence generation, Markov chains together with pre-defined repetition structure schemes were employed in [4], where specific methods for handling tran-

sitions between repeating segments were proposed; in [20], where an approach to a controlled creation of variations was introduced; in [5], where chords were generated, obeying a pre-defined repetition structure. In [12], a convolutional restricted Boltzmann machine was employed, and different structural properties were imposed using differentiable soft-constraints and gradient descent optimization. A constrained variable neighborhood search to generate polyphonic music obeying a tension profile and the repetition structure from a template piece was proposed in [7]. In [6], Markov chains and evolutionary algorithms were used to generate repetition structure for Electronic Dance Music.

3. MODELS

3.1 Gated Autoencoder

A GAE learns first-order derivatives between its input and its output. In musical sequences, this amounts to learning pitch intervals, which are represented as distinct codes in its latent space. In reconstruction, it applies learned interval codes to pitches in order to transpose them. Its ability to learn and to perform musical transformations is, however, not limited to single intervals. For example, it was shown in [11], that more complex musical transformations like diatonic transposition can be learned by a GAE and can be applied to an unseen material. Intervals are encoded in the latent space of the GAE, denoted as mappings

$$\mathbf{m}_{t+1} = \sigma_q(\mathbf{W}_m(\mathbf{Q}\mathbf{x}_{t-n}^t \cdot \mathbf{V}\mathbf{x}_{t+1})), \quad (1)$$

where \mathbf{x}_{t+1} is a binary vector encoding active notes at time step $t+1$ as on-bits, \mathbf{x}_{t-n}^t contain the concatenated vectors of the last n time steps, \mathbf{Q} , \mathbf{V} and \mathbf{W}_m are weight matrices, and σ_q is the softplus non-linearity. The operator \cdot (indicated as a triangle in Figure 1) depicts the Hadamard product of the filter responses $\mathbf{Q}\mathbf{x}_{t-n}^t$ and $\mathbf{V}\mathbf{x}_{t+1}$, denoted as *factors*. This operation allows the model to *relate* its inputs, making it possible to learn interval representations.

GAEs are often trained by minimizing the symmetric error when reconstructing the output from the input and vice versa. In the proposed RGAE architecture, we use predictive training and just learn to reconstruct the target \mathbf{x}_{t+1} from the input \mathbf{x}_{t-n}^t and the mapping \mathbf{m}_{t+1} as

$$\tilde{\mathbf{x}}_{t+1} = \sigma_g(\mathbf{V}^\top(\mathbf{W}_m^\top \mathbf{m}_{t+1} \cdot \mathbf{Q}\mathbf{x}_{t-n}^t)), \quad (2)$$

where σ_g is the sigmoid non-linearity. The GAE portion of the RGAE is pre-trained by minimizing the binary cross-entropy loss of the reconstruction as

$$\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = -\frac{1}{N} \sum_{n=1}^N \left[x_n \log_2 \tilde{x}_n + (1-x_n) \log_2 (1-\tilde{x}_n) \right]. \quad (3)$$

3.2 Recurrent Gated Autoencoder

The proposed model is a combination of a gated autoencoder (GAE) and a recurrent neural network (RNN) as depicted in Figure 1. The GAE learns relative pitch (i.e., interval) representations of the musical surface, and the RNN learns their temporal dependencies.

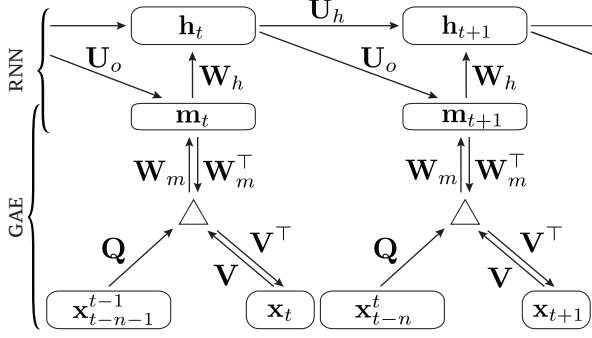


Figure 1: Schematic illustration of the proposed recurrent gated autoencoder architecture. Arrows represent weight matrices, rounded rectangles represent vectors. The triangles depict the Hadamard product. The specifics of the gated recurrent unit are omitted for better clarity.

We use gated recurrent units (GRUs) [2] for the RNN portion of the RGAE. This type of units have been shown to be often as efficient as long short-term memory units (LSTMs, [9]) while being conceptually simpler [3]. It is intuitively clear that any RNN variant can be potentially attached on a GAE. The input to the RNN at time t is the GAE’s mapping \mathbf{m}_t , resulting in the following specification:

$$\mathbf{z}_t = \sigma_g(\mathbf{W}_z \mathbf{m}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z), \quad (4)$$

$$\mathbf{r}_t = \sigma_g(\mathbf{W}_r \mathbf{m}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_r), \quad (5)$$

$$\mathbf{h}_t = \mathbf{z}_t \cdot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \cdot \sigma_h(\mathbf{W}_h \mathbf{m}_t + \mathbf{U}_h (\mathbf{r}_t \cdot \mathbf{h}_{t-1}) + \mathbf{b}_h), \quad (6)$$

where \mathbf{h}_t is the hidden state at time t , \mathbf{z}_t is the update gate vector, \mathbf{r}_t is the reset gate vector, and \mathbf{W} , \mathbf{U} and \mathbf{b} are parameter matrices and vectors. The RNN predicts the next mapping of the GAE as

$$\tilde{\mathbf{m}}_{t+1} = \sigma_q(\mathbf{U}_o \mathbf{h}_t), \quad (7)$$

which is used to reconstruct the target configuration at $t+1$ as

$$\tilde{\mathbf{x}}_{t+1} = \sigma_s(\mathbf{V}^\top (\mathbf{W}_m^\top \tilde{\mathbf{m}}_{t+1} \cdot \mathbf{Q} \mathbf{x}_{t-n}^t)). \quad (8)$$

Here, we use the softmax non-linearity σ_s , as the data the RGAE is trained on is monophonic. The full architecture is trained with Backpropagation through time (BPTT) to minimize the *categorical* cross-entropy loss for the reconstructed target as

$$\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = -\frac{1}{N} \sum_{n=1}^N x_n \log_2 \tilde{x}_n. \quad (9)$$

When the RGAE is applied to polyphonic music, in Equation 8 the sigmoid non-linearity, together with the binary cross-entropy loss (cf. Equation 3) has to be used.

3.3 Baseline RNN

As a baseline, we employ an RNN with GRUs to directly operate on the data. Accordingly, Equations 4, 5, and 6 are

adapted to take \mathbf{x}_t instead of \mathbf{m}_t as input. Consequently, the prediction of the baseline RNN amounts to

$$\tilde{\mathbf{x}}_{t+1} = \sigma_s(\mathbf{U}_o \mathbf{h}_t), \quad (10)$$

where the softmax non-linearity is applied, making the categorical cross-entropy loss (cf. Equation 9) applicable in training.

4. GATED AUTOENCODER PRE-TRAINING

Due to the relatively high number of parameters in its GAE portion, the RGAE is prone to overfitting. To circumvent this, and to establish robust interval representations, we pre-train the GAE first, using the cross-entropy of the reconstruction as the cost function (cf. Equation 3). In the second training iteration, we train the RNN portion of the GAE to minimize the cross-entropy error of the architecture’s prediction (cf. Equation 9). The datasets may differ between the training iterations as long as the included relations are identical (e.g. “intervals of western tonal music”). Consequently, the GAE parameters trained on one dataset can be used for prediction tasks on several datasets. Fine-tuning the whole architecture in the last few epochs of predictive training can make up for possible bias.

In the following, we describe how the GAE is pre-trained in our experiments. Details varying between the experiments are given later in the experiments section (cf. Section 5).

4.0.1 Enforcing Transposition-Invariance

A property of interval representations in music is transposition invariance (i.e., transposing the melody does not change the representation). Although training the GAE as described in Section 3.1 naturally tends to lead to similar mapping codes for input target pairs that have the same interval relationships, the training does not explicitly enforce such similarities and consequently the mappings may not be maximally transposition invariant. Therefore, when pre-training the GAE, we explicitly support the learning of transposition-invariant codes. First, we define a transposition function $shift(\mathbf{x}, \delta)$, which shifts the bits of a vector \mathbf{x} of length M by δ pitches:

$$shift(\mathbf{x}, \delta) = (x_{(0+\delta) \bmod M}, \dots, x_{(M-1+\delta) \bmod M})^\top, \quad (11)$$

where $shift(\mathbf{x}_{t-n}^t, \delta)$ denotes the transposition of each single time step vector *before* concatenation and linearization.

The altered training is then as follows: First, the mapping code \mathbf{m}_{t+1} of an input/target pair is inferred as shown in Equation 1. Then, \mathbf{m}_{t+1} is used to reconstruct a *transposed* version of the target from an equally *transposed* input (modifying Equation 2) as

$$\tilde{\mathbf{x}}'_{t+1} = \sigma_g(\mathbf{V}^\top (\mathbf{W}_m^\top \mathbf{m}_{t+1} \cdot \mathbf{Q} shift(\mathbf{x}_{t-n}^t, \delta))), \quad (12)$$

with $\delta \in [-30, 30]$. Finally, we penalize the error between the reconstruction of the transposed target and the actual transposed target (i.e., employing Equation 3) as

$$\mathcal{L}(shift(\mathbf{x}_{t+1}, \delta), \tilde{\mathbf{x}}'_{t+1}). \quad (13)$$

The transposition distance δ is randomly chosen for each training batch. This method amounts to both, a form of guided training and data augmentation.

4.0.2 Pre-training and Architecture

We use 512 units in the factor layer and 64 units in the mapping layer of the GAE. On the latter, sparsity regularization [14] is applied. The deviation of the norms of the columns of both weight matrices \mathbf{U} and \mathbf{V} from their average norm is penalized. Furthermore, we restrict these norms to a maximum value. The learning rate is reduced from 0.001 to 0 during training, and RMSProp [8] is used.

5. EXPERIMENTS

5.1 Experiment 1: Folk Song Prediction

We test the RGAE and RNN in a sequence learning task using the data described in Section 5.1.1. In order to make the results comparable, we use the same experiment setup as in [1, 22].

5.1.1 Data

The EFSC subset (comprising a total of 54,308 note events) of the Essen Folk Song Collection (EFSC) [24] constitutes the data for the actual training and evaluation. It consists of 119 Yugoslavian folk songs, 91 Alsatian folk songs, 93 Swiss folk songs, 104 Austrian folk songs, the German subset *kinder* (213 songs), and 237 songs of the Chinese subset *shanxi*. The melodies are represented as series of pitches ignoring note durations.

For pre-training the GAE portion of the RGAE, we use a polyphonic Mozart piano music dataset ([29], comprising 13 piano sonatas with more than 106,000 notes) in piano-roll representation (i.e., using a regular time grid of 1/8th note resolution, and an active note can span several time steps). We pre-train on that data because polyphonic music acts as a better regularizer for learning interval representations than monophonic music.

5.1.2 Training and Architecture

We use only 16 hidden units in the RNN portion of the RGAE. The look-back window of the GAE is $n = 8$ pitches, and we apply 50% dropout on the input in pre-training and when training the whole architecture. We pre-train the GAE for 250 epochs on the Mozart piano pieces (cf. Section 5.1.1). Subsequently, the RNN portion is trained for 110 epochs on the interval representations (i.e., mappings provided by the GAE) of the EFSC datasets. In the last 10 epochs the whole architecture is fine-tuned.

The baseline RNN with 50 hidden units is trained for 70 epochs on the EFSC data. The learning rate scheme is adopted from that described in Section 4.0.2 for all models.

5.1.3 Combining Model Predictions

We hypothesize that the RNN and the RGAE are complementary in how they process musical sequences. For example, the RNN may have better stability in remembering absolute reference pitches, like the tonic of a piece, and

is superior in modeling prior probabilities, to keep predictions in a plausible pitch range. In contrast, the RGAE can make use of structural cues indicating repetitions and can generalize better due to relative pitch processing. There are several possibilities to combine the predictions of statistical models. Next to the ad-hoc approach of merely averaging their outputs, we can also use information about the certainty of the models and weight their outputs accordingly. A measure for the certainty of a prediction is given by the Shannon entropy [26]:

$$H(p) = - \sum_{a \in \mathcal{A}} p(a) \log_2 p(a), \quad (14)$$

where $p(a \in \mathcal{A}) = P(\mathcal{X} = a)$ is a probability mass function over a discrete alphabet \mathcal{A} . The method which worked best in our experiments is calculating the entropy-weighted geometric mean of both predictions, as proposed in [21]:

$$p(t) = \frac{1}{R} \prod_{m \in M} p_m(t)^{w_m}, \quad (15)$$

where $p_m(t)$ is the predicted distribution of model m at time t , $w_m = H_{\text{relative}}(p_m)^{-b}$ is the weight of model m , non-linearly scaled using a bias b (set to 0.5 in our experiments), and R is a normalization constant. The relative entropy $H_{\text{relative}}(p_m)$ for model m is given by

$$H_{\text{relative}}(p_m) = \frac{H(p_m)}{H_{\text{max}}(p_m)}, \quad (16)$$

where $H_{\text{max}}(p_m) > 0$ is the entropy of the probability mass uniformly distributed over the alphabet (indicating maximal uncertainty of the model).

5.1.4 Evaluation

Since the datasets are rather small, a fixed training/test set split would lead to a poor estimation of the performance of the models. Therefore, and in accordance with [1, 22], a 10-fold cross validation is performed for each dataset and the categorical cross-entropy loss (cf. Equation 9) is reported.

5.1.5 Results and Discussion

The results are shown in Table 1. The current state-of-the-art results for general connectionist sequence models on the datasets are achieved by the RTDRBM model introduced in [1]. The results show that the RGAE slightly outperforms the RTDRBM and is clearly superior to the baseline RNN. Note that the RGAE only has 16 units for learning temporal dependencies (the GAE portion mainly transforms absolute pitch input to relative pitch representations). This compactness suggests that the relative processing of music indeed supports generalization by reducing the sparsity in the data.

When combining the predictions of the RGAE with an absolute pitch model (i.e., RNN or RTDRBM) based on the entropy-weighted geometric mean (cf. Section 5.1.3), a more substantial improvement is achieved than when combining the two absolute pitch models. This result shows

Data	RNN (GRU)	RTDRBM [1]	RGAE	RNN + RTDRBM	RNN + RGAE	RTDRBM + RGAE
Alsatian folk songs	2.890	2.897	2.872	2.844	2.788	2.771
Yugoslavian folk songs	2.717	2.655	2.676	2.617	2.586	2.530
Swiss folk songs	2.954	2.932	2.895	2.851	2.831	2.769
Austrian folk songs	3.185	3.259	3.171	3.163	3.070	3.085
German folk songs	2.358	2.301	2.305	2.257	2.233	2.184
Chinese folk songs	2.725	2.685	2.752	2.612	2.650	2.595
Average	2.805	2.788	2.779	2.724	2.693	2.656

Table 1: Cross-Entropies of the 10-fold cross validation in the prediction task for different data sets and different models. When combining the RGAE with an absolute pitch model (i.e., RNN, RTDRBM), results improve substantially. The results suggest that absolute and relative pitch models are complementary in the aspects they learn about music and can be effectively used in an ensemble method.

that absolute and relative processing of music are complementary and can, therefore, be effectively used together in an ensemble method.

5.2 Experiment 2: Copy-and-Shift Operations

This experiment shall be seen as a proof-of-concept for the RGAEs ability to learning sequences of copy-and-shift operations (i.e., structure schemes). We oppose our model to an RNN with GRUs, which is known to have difficulties to learn tasks in the form “whatever has been generated before, now create a (shifted) copy of it”. The hypothesis is that the RGAE, due to its modeling of intervals, is superior in solving this task. It has shown in previous studies that it can learn content-invariant transformations between data instances [16], a necessary capability for learning content-invariant structure schemes.

5.2.1 Data

In order to obtain a controlled setup for testing the model performances, we construct data obeying different recurring (chromatic) transposition patterns. To this end, the EFSC dataset is transformed into a piano-roll representation with a resolution of 1/8th note. From that, short fragments of length 4, 8, and 16 (\leq the length of the receptive field of the input to the models) are randomly sampled (rests are omitted). It is necessary that the RGAE has access to all past events with which the prediction should be related. Choosing longer fragment lengths than the lengths of the receptive fields yields considerably worse results, also for the baseline RNN, which already performs weakly in this setup. The fragments are copied and transposed according to some pre-defined transposition schemes (cf. Table 2). For each of the 10 schemes and fragment lengths, 26 sequences (512 time steps each, resulting in 133 120 time steps) are generated, where 20 sequences are used for training, 5 sequences are used for testing and 1 for evaluation. This results in a total of 600 sequences for training, 150 sequences for testing and 30 sequences for evaluation.

5.2.2 Training and Architecture

The lookback window of the RGAE is $n = 16$ time steps, the RNN portion has 64 units, and we do not use dropout on the input. For the baseline RNN, we also input the 16 preceding time steps, as this supports copy operations by

Transposition Schemes
$\{+5, +5, +5, \dots\}$
$\{+7, +7, +7, \dots\}$
$\{-5, -5, -5, \dots\}$
$\{-7, -7, -7, \dots\}$
$\{+12, -12, +12, \dots\}$
$\{+3, -3, +3, \dots\}$
$\{+4, -4, +4, \dots\}$
$\{+9, -9, +9, \dots\}$
$\{+4, -8, +4, -8, \dots\}$
$\{-4, +8, -4, +8, \dots\}$

Table 2: The different relative transposition schemes used in Experiment 2.

freeing up memory in the hidden units. The baseline RNN model size (512 units) is selected by starting from 64 units and always doubling that number until no substantial improvement occurs on the evaluation set.

The GAE portion of the RGAE is pretrained for 50 epochs on the structured sequences described above. Subsequently, the RGAE is trained for 50 epochs, holding the parameters of the GAE fixed. As the data of the pretraining does not differ from the sequences in the prediction task, finetuning is not necessary.

The baseline RNN is trained for 60 epochs. Again, for both models the learning rate scheme described in Section 4.0.2 is employed. Note that in this task, we always randomly transpose the input to the models in all training phases. Therefore, we need no dropout on the input of the RGAE, and the baseline RNN does not overfit, despite its high number of parameters.

5.2.3 Evaluation

The models have to learn to continue sequences from the test set after exposition to the first 64 time steps of each sequence. The experiment is different to typical prediction tasks in that possibly incorrect predictions are fed back to the models, causing errors to accumulate. To obtain more stable continuations, we do not sample from the predicted distributions of the models, but instead, treat the experiment as a classification task and choose the pitch with the highest predicted probability. Accordingly, the precision is merely the percentage of correctly predicted pitches over time. In addition, we quantify how many sequences are correctly continued until the end by considering all se-

Model	Pr (%)	> 99%	CE	# Params
RNN	41.38	6.67	10.10	~ 2 300 000
RGAE	99.43	92.00	0.16	~ 600 000

Table 3: Results of the structure learning task. Average precision (Pr), percentage of continuations above 99% precision, cross-entropy (CE) and number of parameters of the respective model.

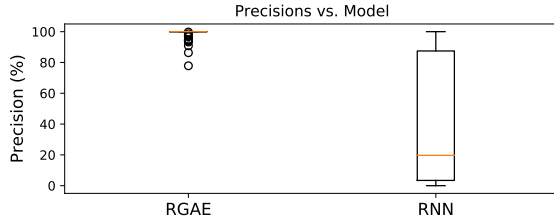


Figure 2: Distribution of precisions for continuation of sequential copy-and-shift operations in the test set of size 150. The median is marked with a orange line, the boxes indicate the interquartile range, and circles indicate outliers.

quences with an overall precision above 99% as correctly continued. Furthermore, like in Experiment 1, the categorical cross-entropy loss (cf. Equation 9) is computed.

5.2.4 Results and Discussion

Table 3 shows the quantitative results of the experiment, and Figure 2 shows a box plot comparing the precisions of the two models. With an average precision of 99.43% percent, where 92% of all examples are flawlessly continued, the RGAE shows remarkable stability in continuing the structure scheme realizations. The cross-entropy of the RGAE is about two orders of magnitude lower than that of the RNN. In Figure 3, a specific example of this sequence continuation task is depicted. Note that the hidden unit activations of the RGAE are more regular because they only represent copy-and-shift operations instead of the musical texture itself (as it is the case for the RNN). The most challenging part for the RGAE is counting, in order to change the copy operation (i.e., transposition distance) at the right time (in fact, at most of the incorrectly continued sequences, the RGAE miscounted by one time step). It is important to note that the hidden unit activations of the RNN portion are identical for identical schemes, because they operate on transformations between events, rather than on the events themselves (i.e., they are largely content-invariant).

6. CONCLUSION AND FUTURE WORK

The principle of modeling sequences of first-order derivatives in music is a compelling concept with the potential to solve two persistent problems in MIR: Learning transposition-invariant interval representations, and learning representations of (chromatically transposed) repetition structure. The proposed model is conceptually simple and can be trained as a generative model in sequence learning tasks.

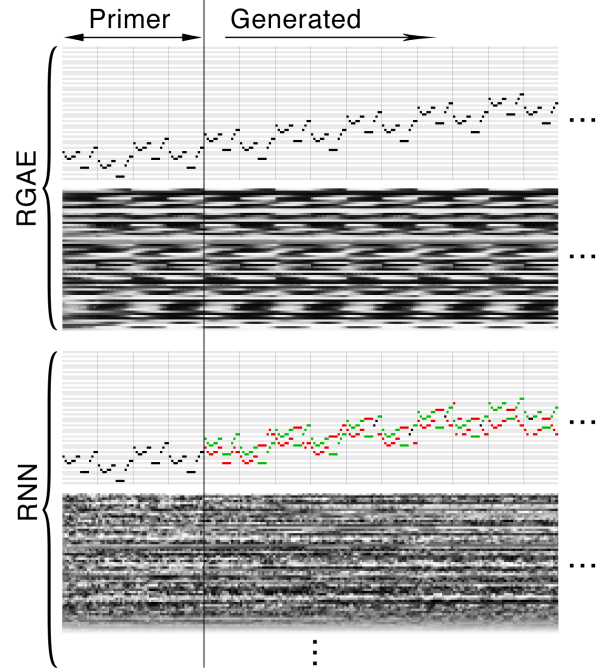


Figure 3: Generated structure schemes and hidden unit activations of the RGAE and the RNN models after input of a primer indicating the $\{-4, +8, -4, +8, \dots\}$ scheme, realized with melodies of length 16 not contained in the train set. Black notes indicate correct continuation, green notes indicate false negatives, red notes indicate false positives. Hidden units activations of the RNN are pruned due to space limitation.

Moreover, the RGAE can act as a building block for more complex architectures, in order to extend its capabilities. For example, the temporal lookback window could be greatly extended by employing the RGAE on top of a (dilated) convolutional network, enabling it to learn higher-level repetition structure. In another variant, an RGAE could be employed on top of an RNN. Applied to music, the RNN would provide the RGAE with representations of important, absolute reference pitches (e.g., the tonic of a scale, or the root note of a chord), and the RGAE could learn sequences of intervals in relation to them. Another interesting architecture would involve stacking more than one RGAE on top of one another to learn higher-order derivatives, for example, variations between mutually transposed parts in music.

The RGAE, however, is not limited to the symbolic, monophonic, domain of music. We show in [13] that a GAE can also operate in the spectral domain of audio and in polyphonic symbolic music. Finally, we note that the RGAE is general enough to be applicable to other domains where the derivatives of functions are of higher importance than their absolute course. Possible applications include modeling temporal progressions of changes in loudness, tempo, mood, information density curves, and other musical properties, modeling moving or rotating objects, camera movements in video recordings, and signals in the time domain.

7. ACKNOWLEDGMENTS

This research was supported by the EU FP7 (project Lrn2Cre8, FET grant number 610859), and the European Research Council (project CON ESPRESSIONE, ERC grant number 670035). We thank Srikanth Cherla for providing us with the source code of the RTDRBM model [1].

8. REFERENCES

- [1] Srikanth Cherla. *Neural Probabilistic Models for Melody Prediction, Sequence Labelling and Classification*. PhD thesis, City, University of London, 2016.
- [2] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103, 2014.
- [3] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [4] Tom Collins, Robin C. Laney, Alistair Willis, and Paul H. Garthwaite. Developing and evaluating computational models of musical style. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 30(1):16–43, 2016.
- [5] Darrell Conklin. Chord sequence generation with semi-otc patterns. *Journal of Mathematics and Music*, 10(2):92–106, 2016.
- [6] Arne Eigenfeldt and Philippe Pasquier. Evolving structures for electronic dance music. In *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013*, pages 319–326. ACM, 2013.
- [7] Dorien Herremans and Elaine Chew. MorpheuS: Automatic music generation with recurrent pattern constraints and tension profiles. In *Proceedings of the IEEE Region 10 Conference (TENCON), Singapore, November 22-25, 2016*, pages 282–285. IEEE, 2016.
- [8] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent, 2012.
- [9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [10] Jonas Langhabel, Robert Lieck, Marc Toussaint, and Martin Rohrmeier. Feature discovery for sequential prediction of monophonic music. In Sally Jo Cunningham, Zhiyao Duan, Xiao Hu, and Douglas Turnbull, editors, *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 649–656, 2017.
- [11] Stefan Lattner and Maarten Grachten. Learning transformations of musical material using gated autoencoders. In *Proceedings of the 2nd Conference on Computer Simulation of Musical Creativity, CSMC 2017, Milton Keynes, UK, September 11-13, 2017*, 2017.
- [12] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. Imposing higher-level structure in polyphonic music generation using convolutional restricted Boltzmann machines and constraints. *Journal of Creative Music Systems*, 3(1), 2018.
- [13] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. Learning transposition-invariant interval features from symbolic music and audio. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*.
- [14] Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng. Sparse deep belief net model for visual area V2. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 873–880. Curran Associates, Inc., 2007.
- [15] Roland Memisevic. Gradient-based learning of higher-order image features. In *IEEE International Conference on Computer Vision (ICCV), 2011*, pages 1591–1598. IEEE, 2011.
- [16] Roland Memisevic and Georgios Exarchakis. Learning invariant features by harnessing the aperture problem. In *ICML (3)*, pages 100–108, 2013.
- [17] Roland Memisevic and Geoffrey Hinton. Unsupervised learning of image transformations. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR.*, pages 1–8. IEEE, 2007.
- [18] Roland Memisevic and Geoffrey E Hinton. Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Computation*, 22(6):1473–1492, 2010.
- [19] Vincent Michalski, Roland Memisevic, and Kishore Konda. "modeling deep temporal dependencies with recurrent grammar cells". In *Advances in neural information processing systems*, pages 1925–1933, 2014.
- [20] François Pachet, Sony CSL Paris, Alexandre Papadopoulos, and Pierre Roy. Sampling variations of sequences for structured music generation. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pages 167–173, 2017.
- [21] Marcus Pearce, Darrell Conklin, and Geraint Wiggins. Methods for combining statistical models of music. In *International Symposium on Computer Music Modeling and Retrieval*, pages 295–312. Springer, 2004.

- [22] Marcus Pearce and Geraint Wiggins. Improved methods for statistical modelling of monophonic music. *Journal of New Music Research*, 33(4):367–385, 2004.
- [23] Marcus Thomas Pearce. *The construction and evaluation of statistical models of melodic structure in music perception and composition*. PhD thesis, City University London, 2005.
- [24] Helmut Schaffrath. The Essen Folksong Collection in Kern Format. In David Huron, editor, *Database containing , folksong transcriptions in the Kern format and a -page research guide computer database*. Menlo Park, CA, 1995.
- [25] Jan Schlueter and Christian Osendorfer. Music similarity estimation with the mean-covariance restricted Boltzmann machine. In *10th International Conference on Machine Learning and Applications and Workshops (ICMLA), 2011*, volume 2, pages 118–123. IEEE, 2011.
- [26] Claude Elwood Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July 1948.
- [27] Ilya Sutskever, Geoffrey E. Hinton, and Graham W. Taylor. The recurrent temporal restricted Boltzmann machine. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1601–1608. Curran Associates, Inc., 2008.
- [28] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.
- [29] Gerhard Widmer. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence*, 146(2):129–148, 2003.

AN END-TO-END FRAMEWORK FOR AUDIO-TO-SCORE MUSIC TRANSCRIPTION ON MONOPHONIC EXCERPTS

Miguel A. Román
U.I. for Computing Research
University of Alicante
Alicante, Spain
mroman@dlsi.ua.es

Antonio Pertusa
U.I. for Computing Research
University of Alicante
Alicante, Spain
pertusa@ua.es

Jorge Calvo-Zaragoza
PRHLT Research Center
Universitat Politècnica de València
Valencia, Spain
jcalvo@prhlt.upv.es

ABSTRACT

In this work, we present an end-to-end framework for audio-to-score transcription. To the best of our knowledge, this is the first automatic music transcription approach which obtains directly a symbolic score from audio, instead of performing separate stages for piano-roll estimation (pitch detection and note tracking), meter detection or key estimation. The proposed method is based on a Convolutional Recurrent Neural Network architecture directly trained with pairs of spectrograms and their corresponding symbolic scores in Western notation. Unlike standard pitch estimation methods, the proposed architecture does not need the music symbols to be aligned with their audio frames thanks to a Connectionist Temporal Classification loss function. Training and evaluation were performed using a large dataset of short monophonic scores (incipits) from the RISM collection, that were synthesized to get the ground-truth data. Although there is still room for improvement, most musical symbols were correctly detected and the evaluation results validate the proposed approach. We believe that this end-to-end framework opens new avenues for automatic music transcription.

1. INTRODUCTION

Automatic Music Transcription (AMT) is a very relevant field within the Music Information Retrieval (MIR) community. This task can be defined as the automated process of converting an audio recording into any kind of musically-meaningful structured format. The usefulness of this process is very broad, especially for MIR algorithms such as content-based music search, symbolic music similarity, or symbolic musicological analysis.

However, this is a challenging task and state-of-the-art methods currently obtain a performance significantly below a human expert. In order to obtain a complete score

from a waveform, it is necessary to perform pitch detection, note onset/offset detection, loudness estimation and quantization, instrument recognition, extraction of rhythmic information, and time quantization [2].

Most music transcription systems focus on two of these stages: pitch detection, where pitches at each time frame of the audio are estimated, and note tracking [32], where the estimations of the previous step are discretized into sequences of 3-tuples (onset, offset, pitch). The output in this case is a piano-roll, that is, a two-dimensional representation of notes across time [2]. Multiple pitch estimation techniques include spectrogram factorization methods [1, 3, 28] and discriminative approaches, which perform frame-by-frame pitch estimation using statistical models [10], signal processing methods [23, 35], or machine learning techniques [4] including deep neural networks [17, 27, 30]. Some works also integrate musical language models into the pitch estimation process to resolve output ambiguities [27, 34].

Supervised learning approaches for piano-roll estimation require the ground truth to be aligned for training. Matching pitches frame by frame with their corresponding waveform samples is a time-consuming task and, although there are some efforts in this direction with datasets such as MAPS [10], RWC [11] or MusicNet [29], currently there are no very large AMT corpora. Beyond the difficulty of performing an accurate annotation, frame-by-frame estimation has some additional issues to be taken into account. For example, when a whole note is played using a plucked string instrument such as a guitar, the quick decay of its harmonic amplitudes produces frames with a very low intensity at the end of the note, causing ambiguities when labeling the offset frames.

In addition, as pointed out in [2], AMT algorithms are usually developed independently to carry out individual tasks such as multiple pitch detection, beat tracking and instrument recognition. Some existing AMT methods, such as the ones proposed in [19–21], also include rhythm estimation and time quantization. Still, the challenge remains to combine the outputs of the individual tasks to perform joint estimation of all parameters, in order to avoid the cascading of errors when algorithms are combined sequentially.

In this work we intend to open a new framework to address the AMT task. Our proposal is to consider end-to-



© Miguel A. Román, Antonio Pertusa, Jorge Calvo-Zaragoza. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Miguel A. Román, Antonio Pertusa, Jorge Calvo-Zaragoza. “An End-to-End framework for Audio-to-Score Music Transcription on monophonic excerpts”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

end machine learning strategies, with which this task can be carried out holistically. In other words, we aim at using a waveform as input, and directly obtaining a music score at the output taking into account all its components (pitches, note durations, time signature, key signature, etc.) jointly.

The task of directly estimating a symbolic score from audio is certainly different from that of estimating a piano-roll. While piano-roll estimation aims to extract what has been played from the audio as exact as possible, in the score transcription task the goal is to obtain a symbolic representation from what the musician read, which includes abstracting away some information such as loudness.

For this, we address score estimation using Deep Neural Networks. We specifically consider the use of a Convolutional Recurrent Neural Network, which is responsible of both processing the input spectrogram to extract meaningful features and predict an output sequence that represents the music contained in a given audio recording. Thanks to the Connectionist Temporal Classification (CTC) loss function, this kind of networks can be trained in terms of pairs (input, output), without needing of dividing the process into smaller stages or providing framewise annotations. The idea is that the prediction is forced to be encoded in terms of actual music-notation elements.

It is important to emphasize that the objective of this work is not to outperform the accuracy of previous approaches, but to propose a framework with which to address the AMT task. In order to demonstrate the feasibility of this formulation, our experiments are restricted to a constrained scenario, using audio recordings from monophonic scores that were synthesized using a piano. We are aware that the main challenge in AMT is to deal with polyphonic real music. In a future work we plan to extend the proposed approach to detect polyphonic scores, although its effectiveness with sound mixtures is yet to be studied.

The evaluation results in this constrained scenario validates the proposed framework and show that the the proposed approach obtains reliable results, correctly detecting most musical symbols.

The rest of the paper is organized as follows: the corpus used for evaluation is described in Section 2; the holistic neural framework proposed for the AMT task is described in Section 3; the series of experiments carried out are detailed in Section 4; and finally, the conclusions of the current work are summarized in Section 5, pointing out some interesting avenues for future work as well.

2. DATASET

In order to get the ground truth for our framework, we used the RISM¹ collection [26], which currently contains more than one million incipits (short monophonic music excerpts). This corpus is very useful for music retrieval tasks because of its size and the fact that it contains real music written by human composers [31]. Spectrograms from synthesized incipits are the inputs to our method, and

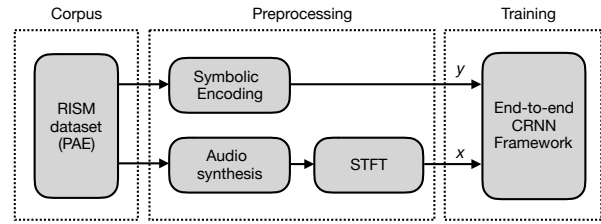


Figure 1: Data acquisition for training. RISM incipits are converted into our music notation format and magnitude spectrograms (Short-Time Fourier Transform, STFT) are also computed from synthesized versions of the incipits. The inputs of the proposed framework (x) are the symbolic data and the outputs are the spectrograms (y). Frame-by-frame alignment is not necessary.

their corresponding symbolic scores are the outputs. The scheme of the proposed method can be seen in Figure 1.

2.1 Preprocessing

RISM incipits are formatted in Plaine & Easie Code (PAE). We randomly selected a subset of 71,400 incipits in Western notation and converted them into the music notation format that can be seen in Table 1, where each symbol is encoded using a single character. This notation is oriented to represent the music as a language, similarly to what a speech recognition system does. Following this analogy, we consider a music note as a *word* (for example, C#4♯) containing several *characters* from an alphabet set Σ that can be seen in Table 1, and which is separated to other words by blank spaces. Rests are represented in the same way, with a word consisting of the rest symbol and its duration. In addition to notes and rests, the alphabet set includes clefs, key and time signatures, measure bars and note ties. Every musical symbol in Table 1 is encoded for our framework using a single element (one character).

In order to get the audio files, the RISM PAE incipits were converted into Music Encoding Initiative (MEI) format, and then translated again into MIDI using Meico², which unlike Verovio³ takes into account the key signature.

The synthesis from MIDI files was performed using timidity with the piano program of the default soundfont, obtaining monoaural audio files at 16kHz. Then, magnitude spectrograms were calculated using a 64ms (1024 samples) Hamming window with a 16ms hop (256 samples). All incipits were synthesized using random tempo values in the range [96-144] bpm in order to make the network work with different speeds.

3. FRAMEWORK

We describe in this section the neural model that allows us to face the AMT task directly from an audio signal to a sequence of meaningful symbols.

¹ The complete set of RISM incipits can be downloaded from <https://opac.rism.info/index.php?id=8&L=1&id=8>

² <https://github.com/cemfi/meico>

³ <http://www.verovio.org/index.xhtml>

Class	Symbol	Count	Histogram
Global	Blank	1,526,051	
Clef	G2	39,337	
	F4	4,414	
	C1	22,468	
	C3	1,981	
	C4	3,200	
Key	D \flat M	112	
	A \flat M	1,065	
	E \flat M	6,815	
	B \flat M	8,950	
	FM	11,599	
	CM	15,488	
	GM	10,309	
	DM	10,861	
	AM	4,933	
	EM	1,216	
	BM	52	
Pitch	A	87,323	
	B	88,004	
	C	95,190	
	D	100,014	
	E	80,780	
	F	75,579	
	G	84,953	
	\flat	70,557	
	\sharp	55,471	
	Rest	89,635	
Octave	2	2,937	
	3	44,170	
	4	274,590	
	5	284,081	
	6	6,065	
Duration	\circ	8,686	
	d	52,541	
	e	172,933	
	f	226,395	
	g	72,124	
	h	6,711	
	\cdot	72,453	
Time	Tie	10,393	
	4/4	27,855	
	2/2	13,848	
	3/4	11,595	
	2/4	7,569	
	6/8	4,950	
	3/8	2,916	
	3/2	1,199	
	12/8	592	
	6/4	417	
	4/2	305	
	9/8	154	
	Barline	245,239	

Table 1: Symbols of the alphabet Σ . Notes are encoded using words of three to five symbols (for example, C \sharp 4 e).

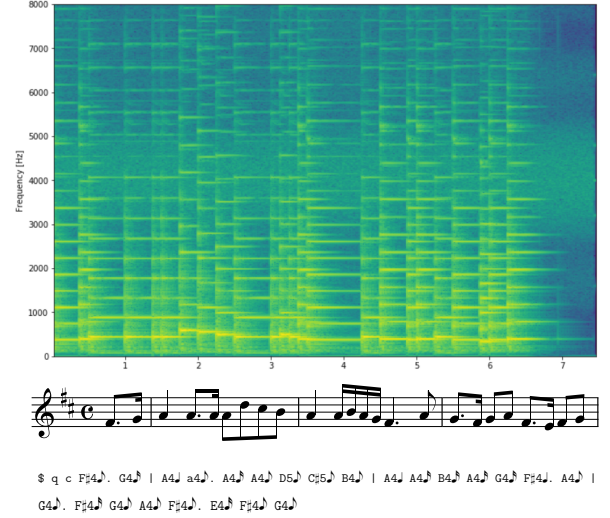


Figure 2: Example of a magnitude spectrogram (x) synthesized from a RISM incipit (center). The symbolic encoding representation used for the CRNN (y) is shown below, where the character ‘\$’ is the G2 clef, ‘q’ is the key signature DM, the symbol ‘c’ is used to encode 4/4 and ‘|’ represents the barline. Similarly to speech recognition, words are separated by blank spaces.

Formally, let $\mathcal{X} = \{(x_1, y_1), (x_2, y_2), \dots\}$ be our end-to-end application domain, where x_i is an audio recording represented by its magnitude spectrogram, and y_i denotes its corresponding ground-truth sequence from a fixed alphabet set Σ .

The problem of AMT can be reformulated as retrieving the most likely sequence of symbols \hat{y} given an input spectrogram x . That is:

$$\hat{y} = \arg \max_{y \in \Sigma^*} P(y|x) \quad (1)$$

We formulate this statistical framework by means of Recurrent Neural Networks (RNN), as they allow handling sequences [12]. Ultimately, therefore, the RNN will be responsible of producing the sequence of musical symbols that fulfills Eq. 1. Nevertheless, on top of it, we add a Convolutional Neural Network (CNN), which learns how to process the input signal to represent it in a meaningful way for the task at issue [36]. Since both types of networks consist of feed-forward operations, the training stage can be carried out jointly by simply connecting the output of the last layer of the CNN with the input of the first layer of the RNN, which leads to a Convolutional Recurrent Neural Network (CRNN). A similar topology was previously applied to drum transcription in [33], although not in an end-to-end fashion.

Our work is conducted over a supervised learning scenario. Therefore, it is assumed that we can make use of a set $\mathcal{T} \subset \mathcal{X}$ with which to train the model. Initially, the traditional training mechanism for a CRNN needs to be provided with the expected output for each frame of the input. As introduced above, for each recording the training set only contains its corresponding sequence of expected

symbols, without any kind of explicit information about their location within the input signal. This scenario can be solved by means of the so-called Connectionist Temporal Classification (CTC) loss function [13].

Given an input x , CTC provides a means to optimize the CRNN parameters in order to directly output its correct sequence y . In other works, CTC directly optimizes $P(y|x)$. Since the ground-truth is not aligned at the frame level, that is, it is unknown the alignment between the frames of the recurrent part and the output symbols, CTC integrates over all possible alignments. It only considers monotonic alignments (left-to-right constraint), which is a valid assumption in our task.

Although optimizing the aforementioned probability is computationally expensive, CTC performs a local optimization using an Expectation-Maximization algorithm similar to that used for training Hidden Markov Models [24]. However, given that CTC integrates over all possible alignments, its main limitation is that the cost of the optimization procedure grows rapidly with the length of the sequences.

Note that CTC is used only for training. At the inference stage, the CRNN still predicts a symbol for each frame of the recurrent block. To indicate a separation between symbols, or to handle those frames in which there is no symbol, CTC considers an additional symbol in the alphabet that indicates this situation (*blank* symbol).

3.1 Implementation details

Finding the best instantiation of a CRNN for the case of AMT is out of the scope of this work, but we are inspired by the *Deep Speech 2* [8] topology, which was especially designed for the task of Automatic Speech Recognition (ASR). Although ASR and AMT are different tasks they are related, and so the use of this architecture allows us to obtain valuable results without having to make an exhaustive search of the best neural topology.

Nonetheless, we made small modifications to the original architecture in order to adjust its behavior to AMT. The specification of our neural topology is detailed in Table 2. It consists of 2 convolutional layers and 3 recurrent layers. Convolutional layers are composed of convolutional filters followed by Batch Normalization [16], and the non-linear hard hyperbolic tangent (HardTanh) activation function [14]. Furthermore, bi-directional recurrent layers are configured as Gated Recurrent Units (GRU) [7], with Batch Normalization as well. On top of the last recurrent output, a fully-connected layer is placed with as many neurons as symbols of the vocabulary (plus 1, because of the *blank* symbol). The use of the *softmax* activation allows us to interpret the output of this last layer as a posterior probability over the vocabulary [6].

The training stage is carried out by providing pairs of spectrograms with their corresponding unaligned sequence of musical symbols. The optimization procedure follows stochastic gradient descent (SGD) [5] with Nesterov momentum of 0.9, gradient L2 Norm clipping of 400, and a mini-batch size of 20 samples, which modifies the network

Input($1024 \times T$)
Convolutional block
Conv($32, 41 \times 11, 2 \times 2$), BatchNorm(), HardTanh()
Conv($32, 21 \times 11, 2 \times 1$), BatchNorm(), HardTanh()
Recurrent block
B-GRU(1024), BatchNorm()
B-GRU(1024), BatchNorm()
B-GRU(1024), BatchNorm()
Dense($ \Sigma + 1$), Softmax()

Table 2: Instantiation of the CRNN used in this work for audio-to-score AMT, consisting of 2 convolutional layers and 3 recurrent layers. Notation: Input($h \times w$) means an input spectrogram of height h and width w ; Conv($n, k_h \times k_w, s_h \times s_w$) denotes a convolution operator of n filters, kernel size of $k_h \times k_w$, and stride of $s_h \times s_w$; BatchNorm() denotes a batch normalization procedure; HardTanh() represents the *hard hyperbolic tangent* activation; B-GRU(n) means a bi-directional Gated Recurrent Units of n neurons; Dense(n) denotes a fully-connected layer of n neurons; and Softmax() represents the *softmax* activation function. Σ denotes the character-wise alphabet considered.

weights to minimize the CTC loss function through back-propagation. The learning rate was initially set to 0.0003, but it was annealed by a factor of 1.1 after each epoch to favor convergence. The model was trained during 20 epochs, fixing the weights according to the best result over the validation set.

Once the CRNN is trained with the previous procedure, it can be used to output a discrete symbol sequence from a given spectrogram. The model yields character-level predictions in each frame. In order to provide an actual symbol sequence, it is necessary to both collapse repeating characters and discarding *blank* characters. Since there could be several frame-level sequences that result in the same sequence of musical symbols, the final decoding is conducted by a beam search procedure [37], with a beam width set to 10.

4. EXPERIMENTS

4.1 Setup

The proposed framework is evaluated using the corpus described in Section 2.1.

Experiments are performed dividing the available data into three independent partitions: 49,980 samples (118.03 hours) for training, 10,710 samples (25.34 hours) for validation, and 10,710 samples (25.36 hours) for the test set, which is used to evaluate the actual performance.

Given the differences with existing AMT approaches, our results are not directly comparable with any previous work. Likewise, there are no standard evaluation metrics with which to evaluate this framework.

Here, we propose a series of metrics especially considered for evaluating the presented approach. In particular,

we are inspired by other tasks, like ASR or Optical Character Recognition (OCR), that are also formulated expecting a sequence of symbols as output. Analogously to these tasks, we also assume that the output consists of individual *characters* (pitches, durations, alterations, ...) that build complete *words* (such as notes). Therefore, the performance can be evaluated in terms of Character Error Rate (CER) and Word Error Rate (WER). These metrics are defined as the number of elementary editing operations (insertion, deletion, or substitution) to convert the hypotheses of the system into the ground-truth sequences, at the character and word level, respectively. They compute this cost in a normalized way according to the length of the ground-truth sequences. Even assuming that these metrics are not optimal for the task of AMT, we hope that they allow us to validate the approach and draw reasonable conclusions from our experimental results.

In order to get some baseline results that can be compared to other works, we also applied the evaluation metric used in [19] for piano-roll alignment tasks. The total number of notes in the ground truth is denoted by N_{GT} , that of estimated notes by N_{est} . The number of notes with pitch errors is denoted by N_p , that of extra notes by N_e , and that of missing notes by N_m . The number of matched notes is defined as $N_{match} = N_{GT} - N_m = N_{est} - N_e$. Then we define the pitch error rate as $E_p = N_p/N_{GT}$, extra note rate as $E_e = N_e/N_{est}$, and missing note rate as $E_m = N_m/N_{GT}$. Onset/offsets errors are also reported in [19]. As we are dealing with note durations instead of onsets/offsets, we include an alternative error metric E_d which is calculated similarly to the pitch error E_p but using note duration errors, denoted by N_d . Thus, we define the duration error rate as $E_d = N_d/N_{GT}$.

4.2 Results

Figure 3 shows the evolution of the errors during the training process. As can be seen, the convergence is fast and the best results on the validation set are obtained at epoch 18, reporting a CER of 5.53 and a WER of 15.98. In the test set, a CER of 5.36 and a WER of 15.67 are obtained. These results are very similar to those from the validation set, thus proving that there is no over-fitting and the model generalizes well.

After an in-depth analysis of the test set transcriptions obtained, we observed that the majority of errors are due to wrong time signatures, barline locations, and clefs. This result was expected in our prior analysis, as even for a human it would be difficult to identify them based on the short audio excerpts we provide to our model (the average number of music measures of the audio excerpts is 4.4). Furthermore, there are some time signatures that contain the same number of notes per measure and therefore they require more musical context to identify them correctly (e.g. 4/4 and 2/2 time signatures), as shown in Figure 4. In other cases, one of these specific errors causes the appearance of many others, as seem to happen with the time signature in the example of Figure 5. In order to address these ambiguities, normalization techniques could be em-

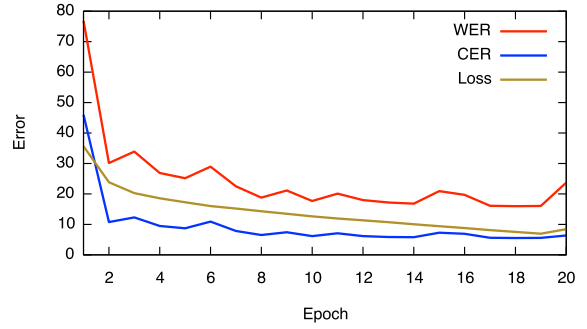


Figure 3: Evolution curves of the CTC loss, CER, and WER over the validation set with respect to the training epoch. The lowest WER (15.98) and CER (5.53) figures are obtained at epoch 18.

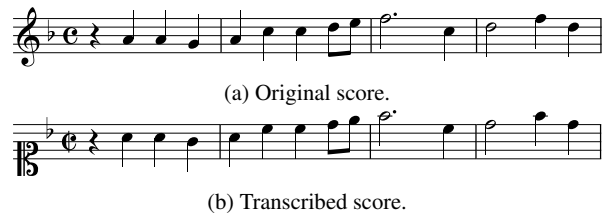


Figure 4: Example of transcription performance. Note that the two mistakes made (clef and time signature) belong to music notation ambiguities.

ployed (for instance, changing all 2/2 by their equivalent notation in 4/4).

In spite of all these difficulties, some samples are perfectly recognized, as the one depicted in Figure 6.

We provide the results of the evaluation metric proposed in [19] for estimated notes, and for estimated notes and rests combined (in this case, E_p does not change). As can be seen in Table 3, the error rates are quite low compared to [19], but this is due to the fact that our audio files are monophonic and synthesized. In addition, most transcription errors are due to wrong estimations of time signatures, subsequently yielding wrong barline locations as previously explained.

5. CONCLUSIONS

In this work, we propose a new formulation of AMT in the form of an audio-to-score task. In summary, the advantages of this formulation over piano-roll estimation are: 1) it is not required to have a frame-by-frame annotation aligned with the audio, therefore potentially more data

Table 3: Note pitch error rate (E_p), missing symbol rate (E_m), extra symbol rate (E_e) and symbol duration error rate (E_d) considering only notes and notes plus rests.

	E_p	E_m	E_e	E_d
Notes	0.99%	2.63%	1.81%	0.71%
Notes+Rests	0.99%	4.94%	2.51%	1.23%

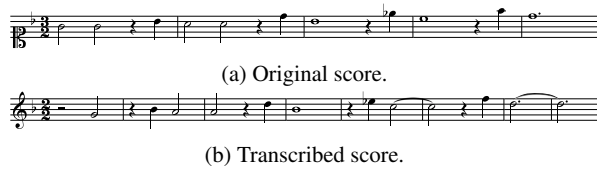


Figure 5: Example of a transcription with several mistakes. Here, the unusual time signature 3/2 (wrongly detected) propagates the errors to the notes.



Figure 6: Example of a correctly transcribed score.

could be acquired for training; 2) the obtained outputs are musically meaningful; 3) the frame-by-frame annotation ambiguities are avoided, although on the other hand there are music notation ambiguities to deal with; 4) the task is addressed holistically instead of using a pipeline of individual processes, avoiding the cascading of errors when they are combined sequentially, and 5) musical models are implicitly inferred as it occurs with language models in speech recognition.

We validated the proposed framework using a CRNN with a CTC loss function trained on RISM incipits, correctly predicting around 84% of symbols for monophonic scores synthesized with a piano sound at different tempos. It is important to note that some symbols such as barlines, rests, ties, time signatures or key signatures were not explicitly present in spectrograms but they were correctly inferred from the context.

A qualitative analysis of the performance reported that many errors occurred because of music notation ambiguities. Although they decreased the WER and CER figures, "wrong" outputs are musically correct and equivalent to the ground-truth scores in most cases.

As a future work, we are planning first to extend it for polyphonic sources, and also to perform instrument recognition. In order to deal with polyphony, a chord could be considered as a "word" containing "syllabus" (the individual notes), for example: $C4\downarrow E4\downarrow G4\downarrow$. An additional symbol could be added to indicate the instrument (for example, $PC4\downarrow$ could represent a quarter note of pitch C4 played on Piano).

As pointed out in [18], previous experiments on deep neural networks dealing with framewise multiple pitch detection showed that unseen combinations are hard to detect. A partial solution to this problem might involve a modification of the loss function for the network to disentangle individual notes explicitly and learn to decompose a (nonlinear) mixture of signals into its constituent parts. We believe that, unlike what happens in this framewise detection, CTC loss may be able to break the observed glass-ceiling, given that ASR methods using this architecture are capable of generalizing to detect unseen words from its constituent (character) elements. Nonetheless, additional experiments on AMT should confirm this hypothesis.

Synthesized scores were used to perform the experiments, although ideally real data should be evaluated. For this, we are planning to use datasets such as Lakh [25], which contains audio files with their corresponding MIDI files. Given the computational cost of CTC, the proposed framework needs to use short segments. Therefore, it is necessary to have aligned barlines to split both the audio and the corresponding score ground truth into smaller pieces. This could be done using a score following method [9, 22]. This preprocessing could introduce some errors due to wrong alignments, but there is a more suitable alternative: to train the CRNN using full scores along with their complete real audio files, which is the ultimate goal of the proposed framework. This is possible and could be done by considering the recently proposed *online* CTC [15] function, which efficiently adapts to any sequence length.

Another obvious future work is to find a more adequate network architecture and evaluate alternative hyperparameters to increase the accuracy. CNN and RNN topologies evaluated in previous AMT works [17, 27] should be investigated for this task.

In conclusion, in this work we show that it is feasible to perform end-to-end transcription from monophonic audio files to scores. We are fully aware that experiments were made in a very controlled and simplified environment and there is still much work to do in order to perform a complete transcription. But we believe that the proposed framework opens a new exciting research area given the huge amount of data that could potentially be used for training, and its practical utility for musicians who could obtain directly a score from audio.

6. ACKNOWLEDGEMENT

This work was supported by the University of Alicante through grant GRE-16-04 and its University Institute for Computing Research (IUI), and by the Spanish Ministerio de Economía, Industria y Competitividad through HispaMus project (TIN2017-86576-R) and Juan de la Cierva - Formación grant (Ref. FJCI-2016-27873).

7. REFERENCES

- [1] E. Benetos and S. Dixon. A shift-invariant latent variable model for automatic music transcription. *Computer Music Journal*, 36(4):81–94, 2012.
- [2] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic Music Transcription: Challenges and Future Directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.
- [3] E. Benetos and T. Weyde. An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *16th International Conference on Music Information Retrieval*, pages 701–707, 2015.
- [4] S. Böck and M. Schedl. Polyphonic piano note transcription with recurrent neural networks. In *IEEE In-*

- ternational Conference on Acoustics, Speech and Signal Processing*, pages 121–124, 2012.
- [5] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.
 - [6] H. Bourlard and C. Wellekens. Links Between Markov Models and Multilayer Perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1167–1178, 1990.
 - [7] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantic and Structure in Statistical Translation*, pages 103–111, 2014.
 - [8] D. Amodei et al. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin. In *33rd International Conference on Machine Learning*, pages 173–182, 2016.
 - [9] R. B. Dannenberg and C. Raphael. Music score alignment and computer accompaniment. *Communications of the ACM*, 49(8):38–43, 2006.
 - [10] V. Emiya, R. Badeau, and B. David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Trans. on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.
 - [11] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC Music Database: Popular, Classical and Jazz Music Databases. In *3rd International Conference on Music Information Retrieval*, pages 287–288, 2002.
 - [12] A. Graves. *Supervised Sequence Labelling with recurrent neural networks*. PhD thesis, Technical University Munich, 2008.
 - [13] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *23rd International Conference on Machine Learning*, International Conference on Machine Learning, pages 369–376. ACM, 2006.
 - [14] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems 29*, pages 4107–4115, 2016.
 - [15] K. Hwang and W. Sung. Online Sequence Training of Recurrent Neural Networks with Connectionist Temporal Classification. *CoRR*, abs/1511.06841, 2015.
 - [16] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July*, pages 448–456, 2015.
 - [17] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. Widmer. On the Potential of Simple Framework Approaches to Piano Transcription. In *17th International Conference on Music Information Retrieval*, 2016.
 - [18] R. Kelz and G. Widmer. An experimental analysis of the entanglement problem in neural-network-based music transcription systems. *arXiv preprint arXiv:1702.00025*, 2017.
 - [19] E. Nakamura, E. Benetos, K. Yoshii, and S. Dixon. Towards complete polyphonic music transcription: Integrating multi-pitch detection and rhythm quantization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2018.
 - [20] E. Nakamura, K. Yoshii, and S. Dixon. Note Value Recognition for Piano Transcription Using Markov Random Fields. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25:1542–1554, 2017.
 - [21] E. Nakamura, K. Yoshii, and S. Sagayama. Rhythm Transcription of Polyphonic Piano Music Based on Merged-Output HMM for Multiple Voices. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25:794–806, 2017.
 - [22] N. Orio, S. Lemouton, and D. Schwarz. Score following: State of the art and new developments. In *Proc. of the 2003 Conference on New interfaces for Musical Expression*, pages 36–41. National University of Singapore, 2003.
 - [23] A. Pertusa and J. M. Iñesta. Efficient methods for joint estimation of multiple fundamental frequencies in music signals. *EURASIP Journal on Advances in Signal Processing*, 2012(1):27, Feb 2012.
 - [24] L. Rabiner and B.-H. Juang. *Fundamentals of speech recognition*. Prentice hall, 1993.
 - [25] C. Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016.
 - [26] RISM. Répertoire International des Sources Musicales, 2017.
 - [27] S. Sigtia, E. Benetos, and S. Dixon. An End-to-end Neural Network for polyphonic piano music transcription. *IEEE Transactions on Audio, Speech and Language Processing*, 24(5):927–939, 2016.
 - [28] P. Smaragdis and J. C. Brown. Non-negative Matrix Factorization for Polyphonic Music Transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, 2003.
 - [29] J. Thickstun, Z. Harchaoui, and S. Kakade. Learning features of music from scratch. In *International Conference on Learning Representations (ICLR)*, 2017.

- [30] J. Thickstun, Z. Harchaoui, D. Foster, and S. M. Kakade. Invariances and data augmentation for supervised music transcription. *arXiv preprint arXiv:1711.04845*, 2017.
- [31] R. Typke, M. Den Hoed, J. De Nooijer, F. Wiering, and R. C. Veltkamp. A ground truth for half a million musical incipits. *Journal of Digital Information Management*, pages 34–39, 2005.
- [32] J. J. Valero-Mas, E. Benetos, and J. M. Ñesta. A supervised classification approach for note tracking in polyphonic piano transcription. *Journal of New Music Research*, pages 1–15, 2018.
- [33] R. Vogl, M. Dorfer, G. Widmer, and P. Knees. Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks. In *18th International Conference on Music Information Retrieval*, pages 150–157, 2017.
- [34] Q. Wang, R. Zhou, and Y. Yan. Polyphonic Piano Transcription with a Note-Based Music Language Model. *Applied Sciences*, 8(3), 2018.
- [35] C. Yeh, A. Robel, and X. Rodet. Multiple fundamental frequency estimation of polyphonic music signals. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3. IEEE, 2005.
- [36] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *13th European Conference on Computer Vision*, pages 818–833, September 2014.
- [37] T. Zenkel, R. Sanabria, F. Metze, J. Niehues, M. Sperber, S. Stüker, and A. Waibel. Comparison of decoding strategies for CTC acoustic models. In *18th Annual Conference of the International Speech Communication Association*, pages 513–517, 2017.

EVALUATING AUTOMATIC POLYPHONIC MUSIC TRANSCRIPTION

Andrew McLeod

University of Edinburgh

A.McLeod-5@sms.ed.ac.uk

Mark Steedman

University of Edinburgh

steedman@inf.ed.ac.uk

ABSTRACT

Automatic Music Transcription (AMT) is an important task in music information retrieval. Prior work has focused on multiple fundamental frequency estimation (multi-pitch detection), the conversion of an audio signal into a time-frequency representation such as a MIDI file. It is less common to annotate this output with musical features such as voicing information, metrical structure, and harmonic information, though these are important aspects of a complete transcription. Evaluation of these features is most often performed separately and independent of multi-pitch detection; however, these features are non-independent. We therefore introduce *MV2H*, a quantitative, automatic, joint evaluation metric based on musicological principles, and show its effectiveness through the use of specific examples. The metric is modularised in such a way that it can still be used with partially performed annotation—for example, when the transcription process has been applied to some transduced format such as MIDI (which may itself be the result of multi-pitch detection). The code for the evaluation metric described here is available at <https://www.github.com/apmcleod/MV2H>.

1. INTRODUCTION

Automatic Music Transcription (AMT) involves converting an acoustic musical signal into some form of music notation. The process has generally been divided into two steps: first, multi-pitch detection, which is the conversion of the signal into a piano-roll notation (such as a MIDI file) by detecting which pitches are present at each time; and second, the conversion of that piano-roll notation into a musical score by annotating it with further musical information. Readers can refer to [2] for an overview of AMT.

The first step, multi-pitch detection, has been the focus of a great amount of research in AMT. The second step involves many subtasks of musical analysis, including voice separation, metrical alignment, note value detection, and harmonic analysis. Each of these has been the subject of research both directly from the acoustic signal and from other input formats such as MIDI. They are usually performed separately, though some recent work has attempted

to analyse subsets of them jointly. For example, [27] estimates both chords and downbeats directly from acoustic input. [33] performs voice streaming, metrical alignment, and harmonic analysis jointly from symbolic input. However, even in the case of these joint models, evaluation is performed separately for each subtask. Rather than simply taking an average of a model’s score on each subtask, there is a need for a standardised way to compute the joint score in a way that reflects overall AMT performance.

In this paper, we introduce *MV2H* (from **M**ulti-pitch detection, **V**oice separation, **M**etrical alignment, **n**ote **V**alue detection, and **H**armonic analysis), a metric to quantitatively evaluate AMT systems that perform both multi-pitch detection and musical analysis. The metric can be used for AMT systems that do not perform all aspects of a full musical analysis—for example, those that perform multi-pitch detection and meter detection, but nothing else. One of the main principles of the new metric is that of disjoint penalties: that mistakes should only be penalised once. That is, if an error in one part of the transcription causes a mistake in another part, that error should not be counted twice. For example, if a pitch is missed during multi-pitch detection, the metric should not further penalise missing that note from the voice separation results.

Based on this principle, we do not include errors related to the proper typesetting of a transcription in our metric, and we do not even require a typeset musical score to perform our evaluation. Most typesetting decisions come down to the proper analysis of the underlying piece. For example, if metrical alignment is performed properly, beaming comes naturally. Likewise, stem directions can follow from voice separation and pitch spelling is a consequence of a proper harmonic analysis. For details related to the proper typesetting of music and its relation to the underlying music analysis, see [14].

2. EXISTING METRICS

Each of the separate tasks involved in the full AMT process has been the subject of much prior research, and there are existing metrics for each of them. This section gives a brief overview of the most widely used metrics for each subtask.

2.1 Multi-pitch Detection

Multi-pitch detection is evaluated both at the frame level and at the note level depending whether a given model includes some form of note tracking or not. As the goal of this paper is to define a metric which is useful for a com-



© Andrew McLeod, Mark Steedman. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: Andrew McLeod, Mark Steedman. “Evaluating Automatic Polyphonic Music Transcription”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

plete AMT system, the note-level evaluation metrics are most applicable here, and readers interested in the frame-based evaluation, an accuracy metric, should refer to [28].

For the note-level metric, a note is defined by its pitch, onset time, and offset time. [1] defines two different precision, recall, and F-measures for note-level multi-pitch detection. For the first, they define true positives as those notes detected whose pitch lies within a quartertone of that of a ground truth note, and whose onset time is within 50 ms of the same ground truth note's onset time, regardless of offset time. Spuriously detected notes are each assigned a false positive, and ground truth notes which are not matched by a detected note are each assigned a false negative. The second metric they propose is identical, with the additional constraint that a detected note's offset time must be accurate to within a certain threshold for it to be considered a true positive. Both of these metrics are used by both the Music Information Retrieval Evaluation Exchange (MIREX) [25] and the *mir_eval* package [29].

For our purposes, we care mostly about onset time and pitch (to the nearest semitone) as these aspects are most directly relevant to the underlying musical score. Offset time, on the other hand, is applicable as far as it relates to note value, and is discussed in Section 2.4. Our multi-pitch detection metric will therefore be based most closely on the first multi-pitch F-measure, which doesn't account for offset time.

2.2 Voice Separation

Voice separation refers to the separation of the notes of a piece of music into perceptual streams called voices. There is no standardised definition of what constitutes a voice, and a full discussion can be found in [3]. In this work, we restrict each voice to be monophonic. This aligns with the majority of work on voice separation, and is beneficial in AMT in that it allows simpler processing of monophonic data to occur in the later musical analysis steps.

There is no standardised metric for evaluating voice separation performance. [5] defines *Average Voice Consistency* (AVC), which returns an average of the percentage of notes in each voice which have been assigned to the correct voice. (A note is said to be assigned to the correct voice if its ground truth voice is the most common one for notes assigned to its voice.) This metric has a problem in that if a model assigns each note to a distinct voice, it achieves a perfect AVC of 100%. For acoustic input, [19, 30] use a similar metric, with the addition that spuriously detected notes automatically count as incorrect.

[17] defines two metrics: *soundness*, which measures the percentage of consecutive notes in an assigned voice which belong to the same ground truth voice; and *completeness*, which measures the percentage of consecutive notes in a ground truth voice which were assigned to the same voice. Finally, [12] defines a precision, recall, and F-measure evaluation, in which the problem of voice assignment is treated as a graph problem where each note is represented by a node, and two nodes are connected by an edge if and only if they are consecutive notes in an assigned

voice. The values are calculated by counting the number of correct edges (true positives), spurious edges (false positives), and omitted edges (false negatives).

Each of these metrics would penalise an AMT system for any spurious notes, so for our proposed metric, we will need to use a modified version of one of them (or design a new metric) in order to enforce the principle of disjoint penalties.

2.3 Metrical Alignment

Metrical alignment is most often approached as one of three different tasks: downbeat tracking, beat tracking, or metrical structure detection. Downbeat tracking and beat tracking each involve identifying points in time, and thus can theoretically be evaluated using the same metrics, which are summarised by [8, 9]. F-measure [11] (which downbeat tracking work uses almost exclusively), is calculated by counting the number of (down)beats within some window length (usually 70 ms) of an annotated (down)beat. [4] proposes a similar metric, where accuracy is calculated instead using a Gaussian window around each annotated beat. [13] proposes a binary metric which is 1 if the beats are correctly tracked for at least 25% of the piece, and 0 otherwise. *P-score* [18], is the proportion of tracked beats which correctly match an annotated beat, normalised by either the number of tracked beats or the number of annotated beats (whichever is greater). Finally, [15] proposes metrics based on the longest continuously tracked section of music. All of the above are used to some extent in beat-tracking, and all are used by both *mir_eval* [29] and MIREX. [21, 23] In addition, evaluation is also often presented at twice and half the annotated beat length, to handle models which may track a beat at the wrong metrical level.

By comparison, the evaluation of metrical structure detection is far less sophisticated. Meter detection is the organisation of the beats of a given musical performance into a sequence of trees at the bar level, in which each node represents a single note value. The structure of each of these trees is directly related to the music's time signature, where the head of each tree splits into a number of nodes equal to the number of beats per bar, and each of these beat nodes splits into a number of nodes equal to the number of sub-beats per beat. Thus, each time signature uniquely describes a single metrical tree structure as defined by the number of beats per bar and sub-beats per beat in that time signature. The most basic evaluation is to simply report the proportion of musical excerpts for which the model guesses the correct metrical structure and phase (such that each tree aligns correctly with a single bar). Another approach is to simply report the proportion of musical excerpts for which the model correctly classifies the meter as duple or triple [10]. Both of these metrics are simplistic, and fail to take into account some idea of partially correct metrical structure trees.

Two metrics have been used for metrical structure detection evaluation which contain within them an evaluation of beat tracking and downbeat tracking, making them ideal for an evaluation of a joint model. [31] proposes a metric

which takes into account the level on the metrical tree at which each note lies in order to capture some idea of partial correctness. However, since it is based on detected notes, it is not robust to errors in multi-pitch detection. [20] introduces an F-measure metric where each level of the detected metrical structure is assigned a true positive if it matches any level of the ground truth metrical structure (even if it is not the same level). A false positive is given for any level of the detected metrical structure which clashes with a metrical grouping in the ground truth, and a false negative for any metrical level in the ground truth which remains unmatched by a level of the detected metrical structure. As it is based solely on metrical groupings, rather than notes, it is robust to multi-pitch detection errors, and would not violate our principle of disjoint penalties. However, it was designed for use with metronomic input, and would therefore need to be adapted for our purposes of evaluating a complete AMT system on live performance data.

2.4 Note Value Detection

Note value detection (identifying a note as a quarter note, eighth note, dotted note, tied note, etc.) is not a widely researched problem, related to a combination of note offset time and metrical alignment. [26] describes two metrics for the task. One, error rate, is simply the percentage of notes whose transcribed value is incorrect. The other, scale error, takes into account the magnitude of the error as well (relative to the metrical grid), in log space such that errors from long notes do not dominate the calculation.

However, since the measured note values are reported relative to the underlying meter, they violate our property of disjoint penalties and we must design a new measure of note value detection accuracy for our metric.

2.5 Harmonic Analysis

Harmonic analysis involves both key detection, a classification problem of identifying one of twelve tonic notes, each with two possible modes (major or minor—alternate mode detection has not been widely researched); and chord tracking, identifying a sequence of chords and times given an audio recording. The possible chords to identify range from simply identifying the correct root note, to determining major or minor, identifying seventh chords, and even identifying different chord inversions.

The standard key detection evaluation, used by both *mir_eval* [29] and MIREX [24], is to assign a score of 1.0 to the correct key, 0.5 to a key which is a perfect fifth too high, 0.3 to the relative major or minor of the correct key, 0.2 to the parallel major or minor of the correct key, and 0.0 otherwise.

The standard chord tracking evaluation is *chord symbol recall* (CSR)—described by [16], and used by both MIREX [22], and *mir_eval* [29]—defined as the proportion of the input for which the annotated chord matches the ground truth chord. There can be varying levels of specificity for what exactly constitutes a match, since different sets of possible chords can be used as described above.

2.6 Joint Metric

For the joint evaluation of AMT performance, [7] presents a system to transcribe MIDI input into a musical score (thus including errors from typesetting), and evaluate it using five human evaluators. The evaluators were asked to: “1) Rate the pitch notation with regard to the key signature and the spelling of notes. 2) Rate the rhythmic notation with regard to the time signature, bar lines, and rhythmic values. 3) Rate the notation with regard to stems, voicing, and placement of notes on staves,” each on a scale of 1 to 10. The three questions roughly correspond with four of our sections above: 1) harmonic analysis; 2) metrical alignment, note value detection; and 3) voice separation.

[6] describes an automatic metric for the same task, similar to string edit distance, taking into account the ordering of 12 different aspects of a musical score: barlines, clefs, key signatures, time signatures, notes, note spelling, note durations, stem directions, groupings, rests, rest duration, and staff assignment.

While this metric is a great step towards an automatic evaluation of AMT performance, it violates our principle of disjoint penalties. A single mistake in metrical alignment can manifest itself in the time signature, rest durations, note durations, and even additional notes (tied notes are counted as separate objects in the metric).

It appears that both of the above metrics measure something slightly different from what we want. They measure the readability of a score produced by an AMT system, while we really want a metric which measures the accuracy of the analysis performed by the AMT system, a slightly different task. To our knowledge, no metric exists which measures the accuracy of the analysis performed by a complete AMT system in the way we desire.

3. NEW METRIC

Our proposed metric, *MV2H*, draws from existing metrics where possible, though we take care to ensure that our principle of disjoint penalties is not violated. Essentially, we calculate a single score for each aspect of the transcription, and then combine them all into the final joint metric.

3.1 Multi-pitch Detection

For multi-pitch detection, we use an F-measure very similar to the one by [1] described above, counting a detected note as a true positive if its detected pitch (in semitones) is correct and its onset lies within 50 ms of the ground truth onset time. All other detected notes are false positives, and any unmatched ground truth notes are false negatives. Note offset time does not factor into our evaluation; rather, see Section 3.4 for a discussion on the related problem of note value detection.

3.2 Voice Separation

For voice separation, we use an F-measure very similar to [12], taking care not to violate our principle of disjoint penalties. Specifically, we don’t want to penalise any model in voice separation for multi-pitch detection errors.



Figure 1: An example transcription of the ground truth bar (left) is shown (right). The voice connection between the last two notes in the lower voice counts as a true positive, even though they are not consecutive in the ground truth.

Recall that the F-measure is calculated as a binary classification problem where for each ordered pair of notes, we must decide if they occur consecutively in the same voice or not. To address the disjoint penalties violation, we alter this slightly. We first remove from both the ground truth voices and the detected voices any notes which have not been matched as a true positive. Then, we perform the same F-measure calculation with the new voices.

As an illustration of this, see Figure 1. In the transcribed music, the last two notes in the lower voice are both matched with a ground truth note (in pitch and onset time), but are not immediately sequential in the ground truth voice. However, because the intervening note was not correctly transcribed, the link between these two notes counts as a true positive. (The second note in the transcribed lower voice does indeed count as an error.) This new F-measure calculation is equivalent to the standard voice separation F-measure when multi-pitch detection is performed perfectly.

3.3 Metrical Alignment

For metrical alignment, we would like to use a metric similar to that from [20] which has some idea of the partial correctness of a metrical alignment. However, as it is designed for use mainly on metronomic data where a metrical hypothesis cannot move in and out of phase throughout a piece, a few adjustments must be made to adapt it for use on live performance data. We call our newly designed evaluation metric the metrical F-measure. It takes into account every grouping at three levels of the metrical hierarchy throughout an entire piece: the sub beat level, the beat level, and the bar level.

For each hypothesised grouping at these metrical levels, we check if it matches a ground truth grouping at any level. A hypothesised grouping is said to match a ground truth grouping if its beginning and ending times are each within 50 *ms* of the beginning and ending times of that particular ground truth grouping, regardless of the metrical level of either grouping.¹ Each matched pair of groupings within a piece count as a true positive, while any unmatched hypothesis groupings count as false positives, and any unmatched ground truth groupings count as false negatives. The metrical F-measure of a piece is then calculated

¹ We use a 50 *ms* threshold, rather than the more common 70 *ms*, because it was shown by [8] that 50 *ms* corresponds more exactly with human judgement for beat tracking. However, this threshold may need to be tuned for different genres as regular syncopation can tend to misalign notes with the metrical grid in certain genres more than others [32].



Figure 2: An example transcription of the ground truth bar (left) is shown (right). Those notes which are assigned a note value score are coloured. Of those, the C (assuming treble clef) is assigned a score of 0.5, while the others are assigned a score of 1.

as the harmonic mean of precision and recall as usual.

3.4 Note Value Detection

It is difficult to disentangle note value detection from multi-pitch detection, voice separation, and metrical alignment in order to include it in our evaluation without violating our principle of disjoint penalties. Clearly, note value should only be regarded if the note has been counted as a true positive in the multi-pitch detection evaluation. Less obviously, we also disregard any detected note which is not followed in its transcribed voice by the correct note. Additionally, note value depends directly on meter such that any note value accuracy metric must measure note value relative to time rather than the underlying metrical grid.

Therefore, we define a note value score which measures only a subset of the detected notes: those which both (1) correspond with a true positive multi-pitch detection; and (2) correspond with a true positive ground truth voice segment as described in the previous paragraph. Each note which matches those two criteria is assigned a score according to the accuracy of its normalised duration (that is, the duration corresponding to its note value rather than its performed duration). Specifically, each note is counted as correct and assigned a score of 1 if its normalised duration is within 100 *ms* of the normalised duration of the corresponding ground truth note.² Otherwise, its score is calculated as in Equation 1, where dur_{gt} is the ground truth note's normalised duration and dur_{det} is the detected note's normalised duration. This score is 1 when the durations match exactly and scales linearly on both sides to a score of 0 for a note with 0 duration or a note with twice the ground truth note's duration. The overall note value score is calculated as the arithmetic mean of the scores of those notes which are assigned a score.

$$score = \max\left(0, 1 - \frac{|dur_{gt} - dur_{det}|}{dur_{gt}}\right) \quad (1)$$

Figure 2 illustrates this note value score. Only those notes which are coloured are considered for the note value score. Notice that the two C's (assuming treble clef) on the downbeat are not considered due to errors in voice separation. Likewise, the last two notes in the lower voice are also not counted against note value score due to note detection errors, even though they count as true positives for

² We use 100 *ms* here to allow for a 50 *ms* error in both onset and offset time, although this value again may need to be tuned for different genres.

the voice separation F-measure. Of the coloured notes, the C would be assigned a score of around 0.5 (depending on exact timing), since its value duration is off by exactly half of the ground truth note's value duration. The others would receive scores of 1. Thus, the final note value score would be the average of 1, 1, 1, and 0.5, or about 0.875.

3.5 Harmonic Analysis

For harmonic analysis, we use the standard key detection and CSR metrics described above, as neither one violates our principle of disjoint penalties since they are based on time rather than notes or the metrical alignment. For now, we take the set of possible chords to include a major and minor version for each root note, but not sevenths or inversions, although the full collection of chords should be used for the final version of our metric.

To combine the two into a single harmonic analysis score, we take the arithmetic mean of the two values, since they are both on the range [0–1]. Models which only perform one of the above tasks may simply use that task's score as their harmonic analysis score.

3.6 Joint Metric

We now have five values to combine into a single number: the multi-pitch detection F-measure, the voice separation F-measure, the metrical F-measure, the note value detection accuracy score, and the harmonic analysis mean. All of these values are on the range [0–1] such that a value of 1 results from a perfect transcription in that aspect. We consider three different approaches for their combination: harmonic mean, geometric mean, and arithmetic mean.

Harmonic mean is most useful when there is potential for one of the values involved to be significantly larger than the others, and thus dominate the overall result. F-measure, for example, is the harmonic mean between precision and recall, and is used so that models cannot receive a high F-measure by simply tuning their model to have a very high recall or precision; rather, both recall and precision must be relatively high in order for their harmonic mean to also be high. This is not relevant in our case as there is no way for a model to tune itself towards one very high score at the expense of the others as is the case with some binary classification problems.

Geometric mean is most useful when the values involved are on different scales. Then, a given percent change in one of the values will result in the same change in mean as the same percent change to another of the values. This property is not necessary for us because all of our values lie on the same range.

Arithmetic mean is a simple calculation that weights each of the values involved equally. This property is desirable for us because, for a complete transcription, all five aspects of an analysis must be correct. Furthermore, due to our property of disjoint penalties, we have kept the five values involved disjoint, and a model must fairly perform well on all aspects in order for its overall score to be high.

Therefore, for the final joint metric, *MV2H* (for **M**ulti-pitch detection, **V**oice separation, **M**etrical alignment, note

Value detection, and **H**armonic analysis), we take the arithmetic mean of the five previously calculated values. We also want the metric to be usable no matter what subset of analyses is performed, for example, for models which run on MIDI input and therefore do not perform multi-pitch detection. In these cases, we advise using our metric and simply taking the arithmetic mean of only those scores which correspond with analyses performed. In future work, we will investigate whether a linear combination of the five values involved, perhaps weighting some more strongly than others, aligns more exactly with human judgements than the current arithmetic mean.

4. EXAMPLES

To illustrate the effectiveness and appropriateness of our metric, we present in Figure 3 two example transcriptions of the first four bars of Bach's Minuet in G, each exhibiting different errors. Figure 3a shows the ground truth transcription (where the chord progression is shown beneath the staff), and the example transcriptions are shown below. We make two assumptions: (1) ground truth voices are separated by clef (plus the bottom two notes in the initial chord, which each belong to their own voice); and (2) The sub beats of each transcription align in time with the sub beats of the ground truth.

Figure 3b shows an example transcription which is good in general, with just a few mistakes, mostly related to the metrical alignment. First, for the multi-pitch detection F-measure, we can see that the transcription has 20 true positives, 3 false negatives (a G on the second beat in the first bar, a C on the second beat of the third bar, and the final G in the fourth bar), and 0 false positives, resulting in an F-measure of 0.93. For voice separation, this transcription is generally good, making a single bad assignment in the second bar, resulting in 3 false positives (the connections to and from the incorrect assignment, as well as the incorrect connection in the treble clef), 3 false negatives (the correct connections to and from the misclassified note, as well as the correct connection in the bass clef), and a voice separation F-measure of 0.83. Notice that the missed G in the upper voice in the treble clef of the first bar does not result in a penalty for voice assignment due to our principle of disjoint penalties. For metrical alignment, we can see that this transcription is notated in $\frac{6}{8}$ time, correctly grouping all sub beats (eighth notes) and bars, yielding 28 true positives, but incorrectly grouping three sub beats together into dotted quarter note beats, yielding 8 false positives and 12 false negatives. This results in a metrical F-measure of 0.74. For note value detection, 14 notes are counted: all of the bass clef notes and all of the eighth notes in the first bar, only the high D in the second bar, the low C and all of the eighth notes in the third bar, and the high G and the low B in the fourth bar. Notice that the initial high D isn't counted because the next note in its voice has not been detected. Similarly, neither the G on the second beat of the second bar nor any of the bass clef notes in the second bar are counted due to voice separation errors. Of the 14 notes, 13 of them are assigned the correct note value (even the

(a) Ground truth

(b) Transcription 1

(c) Transcription 2

Figure 3: Two different example transcriptions of the first four bars of Bach’s Minuet in G.

first bass chord, since its incorrect typesetting and the ties are related to the incorrect metrical alignment—the note value still ends at the correct point in time). One note (the C in the bass clef on the downbeat of the third bar) is assigned a value score of 0.5 (since its value duration is half of the correct value duration). This results in a note value detection score of 0.96. The harmonic analysis in this transcription is entirely correct, resulting in a harmonic score of 1.0. Thus, the *MV2H* of the first transcription is 0.89. This makes sense because the transcription is quite good in general, but a few mistakes are made, the most glaring of which is the metrical alignment (its lowest score).

Figure 3c shows another example transcription which is again good in general, this time with a few more errors in multi-pitch detection, as well as a poor harmonic analysis. For multi-pitch detection, it contains 17 true positives, 4 false positives, and 6 false negatives, resulting in an F-measure of 0.77. This number is 0.16 lower than that the previous transcription’s corresponding F-measure, and this makes sense intuitively: the first transcription does seem to have resulted from a more accurate multi-pitch detection than the second. For voice separation, this second transcription contains no errors. Some erroneous notes are placed into one voice or the other, but all of the correctly detected notes are also correctly separated into voices, resulting in a perfect voice separation F-measure of 1.0. Likewise the metrical alignment is performed perfectly, resulting in a metrical F-measure of 1.0. For note value detection, we look at all of the true positive note detections except (1) the initial D on the downbeat of the first bar, (2) the B in the bass clef of the first bar, (3) the C in the bass clef of the third bar, and (4) the high F at the end of the third bar. (All of these exceptions are due to missed note detections of the following note in each voice.) All of the remaining notes have been assigned the correct value, resulting in a note value detection score of 1.0. For the harmonic analysis, the model has incorrectly transcribed the excerpt in D major, resulting in a key score of 0.5.

Transcription	1	2
Multi-pitch	0.93	0.77
Voice	0.83	1.0
Meter	0.74	1.0
Note Value	0.96	1.0
Harmonic	1.0	0.5
<i>MV2H</i>	0.89	0.85

Table 1: The resulting evaluation scores from each of the example transcriptions from Figure 3.

Likewise, the model has incorrectly labelled the chord progression as D-G-G-G, rather than G-G-C-G. Thus, it has transcribed the correct chord for half of the transcription, resulting in a CSR of 0.5, and a harmonic score of 0.5. The *MV2H* of the second transcription is therefore 0.85: slightly worse than the first transcription, but still good.

The scores of both transcriptions are summarised in Table 1, and intuitively, they make sense. Both seem good overall, though they both contain errors. The first transcription has an incorrectly notated meter (although its bars and sub beats still align correctly) and a few other smaller mistakes related to multi-pitch detection, voice separation, and note value detection. The second transcription, on the other hand, correctly aligns the meter, and makes its only errors in its harmonic analysis (which is quite poor), and in multi-pitch detection (it is worse than the first model in this regard). Given these examples, for applications which need a good all-around transcription, we would recommend the system which produced the first transcription. However, applications which emphasise metrical structure detection or voice separation should consider using the system which produced the second transcription instead.

5. CONCLUSION

As research moves towards the goal of a complete AMT system, an automatic, standardised, quantitative metric for the task will become a necessity. To that end, we have proposed a joint metric, *MV2H*, which measures multi-pitch detection, voice separation, metrical alignment, note value detection, and harmonic analysis and summarises them in a single number. Our metric is based on the property of disjoint penalties: that a model should not be penalised twice for errors which come from a single mistake or misinterpretation. While our metric may not be the final standardised metric used for the task, we believe that it should become part of the discussion, and that the principles that guided us through its creation should continue to be addressed by any future proposed metrics.

Future work will evaluate our metric on a wider corpus of realistic transcriptions. In particular, we will investigate how well our metric aligns with human judgements, testing a weighted average of the five values involved, rather than using the arithmetic mean. A more advanced multi-pitch detection metric, for example one which weights errors according to their perceptual salience, could be another avenue for improvements.

6. ACKNOWLEDGEMENTS

This work was partially funded by EU ERC H2020 Advanced Fellowship GA 742137 SEMANTAX and a Google faculty award.

7. REFERENCES

- [1] Mert Bay, Andreas F. Ehmann, and J. Stephen Downie. Evaluation of Multiple-f0 Estimation and Tracking Systems. In *ISMIR*, pages 315–320, 2009.
- [2] Emmanouil Benetos, Simon Dixon, Dimitrios Gianneoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, July 2013.
- [3] Emiliós Cambouropoulos. Voice And Stream: Perceptual And Computational Modeling Of Voice Separation. *Music Perception*, 26(1):75–94, September 2008.
- [4] A. Taylan Cemgil, Bert Kappen, Peter Desain, and Henkjan Honing. On tempo tracking: Tempogram Representation and Kalman filtering. *Journal of New Music Research*, 29(4):259–273, December 2000.
- [5] Elaine Chew and Xiaodan Wu. Separating Voices in Polyphonic Music: A Contig Mapping Approach. In *Computer Music Modeling and Retrieval*, pages 1–20, 2004.
- [6] Andrea Cogliati and Zhiyao Duan. A metric for music notation transcription accuracy. In *ISMIR*, pages 407–413, 2017.
- [7] Andrea Cogliati, David Temperley, and Zhiyao Duan. Transcribing Human Piano Performances into Music Notation. In *ISMIR*, pages 758–764, 2016.
- [8] Matthew E. P. Davies and Sebastian Böck. Evaluating the Evaluation Measures for Beat Tracking. In *ISMIR*, pages 637–642, 2014.
- [9] Matthew E. P. Davies, Norberto Degara, and Mark D. Plumbley. Evaluation Methods for Musical Audio Beat Tracking Algorithms. *Queen Mary University of London, Centre for Digital Music, Technical Report C4DM-TR-09-06*, 2009.
- [10] W. Bas De Haas and Anja Volk. Meter Detection in Symbolic Music Using Inner Metric Analysis. In *ISMIR*, pages 441–447, 2016.
- [11] Simon Dixon. Evaluation of the Audio Beat Tracking System BeatRoot. *Journal of New Music Research*, 36(1):39–50, March 2007.
- [12] Ben Duane and Bryan Pardo. Streaming from MIDI using constraint satisfaction optimization and sequence alignment. In *Proceedings of the International Computer Music Conference*, pages 1–8, 2009.
- [13] Masataka Goto and Yoichi Muraoka. Issues in Evaluating Beat Tracking Systems. In *Workshop on Issues in AI and Music*, pages 9–16, 1997.
- [14] Elaine Gould. *Behind bars : the definitive guide to music notation*. Faber Music, 2011.
- [15] Stephen W. Hainsworth. *Techniques for the Automated Analysis of Musical Audio*. PhD thesis, University of Cambridge, 2003.
- [16] Christopher Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, Queen Mary University of London, 2010.
- [17] Phillip Kirlin and Paul Utgoff. VOISE: Learning to Segregate Voices in Explicit and Implicit Polyphony. In *ISMIR*, pages 552–557, 2005.
- [18] M. F. McKinney, D. Moelants, Matthew E. P. Davies, and Anssi Klapuri. Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms. *Journal of New Music Research*, 36(1):1–16, March 2007.
- [19] Andrew McLeod, Rodrigo Schramm, Mark Steedman, and Emmanouil Benetos. Automatic transcription of polyphonic vocal music. *Applied Sciences*, 7(12), 2017.
- [20] Andrew McLeod and Mark Steedman. Meter detection in symbolic music using a lexicalized pcfg. In *Proceedings of the 14th Sound and Music Computing Conference*, pages 373–379, 2017.
- [21] MIREX. Audio beat tracking. http://www.music-ir.org/mirex/wiki/2017:Audio_Beat_Tracking, 2017. Accessed: 2017-07-18.
- [22] MIREX. Audio chord estimation. http://www.music-ir.org/mirex/wiki/2017:Audio_Chord_Estimation, 2017. Accessed: 2017-07-18.
- [23] MIREX. Audio downbeat estimation. http://www.music-ir.org/mirex/wiki/2017:Audio_Downbeat_Estimation, 2017. Accessed: 2017-07-18.
- [24] MIREX. Audio key detection. http://www.music-ir.org/mirex/wiki/2017:Audio_Key_Detection, 2017. Accessed: 2017-07-18.
- [25] MIREX. Multiple fundamental frequency estimation & tracking. http://www.music-ir.org/mirex/wiki/2017:Multiple_Fundamental_Frequency_Estimation_%26_Tracking, 2017. Accessed: 2017-07-18.
- [26] Eita Nakamura, Kazuyoshi Yoshii, and Simon Dixon. Note value recognition for piano transcription using markov random fields. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(9):1846–1858, sep 2017.

- [27] Hélène Papadopoulos and Geoffroy Peeters. Joint Estimation of Chords and Downbeats From an Audio Signal. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(1):138–152, January 2011.
- [28] Graham E Poliner and Daniel P. W. Ellis. A Discriminative Model for Polyphonic Piano Transcription. *EURASIP Journal on Advances in Signal Processing*, 2007(1):154–162, 2007.
- [29] Colin Raffel, Brian Mcfee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel P. W. Ellis, C Colin Raffel, Brian Mcfee, and Eric J. Humphrey. mir_eval: A Transparent Implementation of Common MIR Metrics. In *ISMIR*, 2014.
- [30] Rodrigo Schramm, Andrew McLeod, Mark Steedman, and Emmanouil Benetos. Multi-pitch detection and voice assignment for a cappella recordings of multiple singers. In *ISMIR*, pages 552–559, Suzhou, October 2017.
- [31] David Temperley. An Evaluation System for Metrical Models. *Computer Music Journal*, 28(3):28–44, September 2004.
- [32] David Temperley. Communicative pressure and the evolution of musical styles. *Music Perception*, 21:313–337, 2004.
- [33] David Temperley. A Unified Probabilistic Model for Polyphonic Music Analysis. *Journal of New Music Research*, 38(1):3–18, March 2009.

ONSETS AND FRAMES: DUAL-OBJECTIVE PIANO TRANSCRIPTION

Curtis Hawthorne^{*†} Erich Elsen^{*} Jialin Song^{*†}
Adam Roberts Ian Simon Colin Raffel Jesse Engel Sageev Oore Douglas Eck
Google Brain Team, Mountain View, CA, USA

[†] Please direct correspondence to: fjord@google.com

ABSTRACT

We advance the state of the art in polyphonic piano music transcription by using a deep convolutional and recurrent neural network which is trained to jointly predict onsets and frames. Our model predicts pitch onset events and then uses those predictions to condition framewise pitch predictions. During inference, we restrict the predictions from the framewise detector by not allowing a new note to start unless the onset detector also agrees that an onset for that pitch is present in the frame. We focus on improving onsets *and* offsets together instead of either in isolation as we believe this correlates better with human musical perception. Our approach results in over a 100% relative improvement in note F1 score (with offsets) on the MAPS dataset. Furthermore, we extend the model to predict relative velocities of normalized audio which results in more natural-sounding transcriptions.

1. INTRODUCTION

Automatic music transcription (AMT) aims to create a symbolic music representation (e.g., MIDI) from raw audio. Converting audio recordings of music into a symbolic form makes many tasks in music information retrieval (MIR) easier to accomplish, such as searching for common chord progressions or categorizing musical motifs. Making a larger collection of symbolic music available also broadens the scope of possible computational musicology studies [8].

Piano music transcription is a task considered difficult even for humans due to its inherent polyphonic nature. Accurate note identifications are further complicated by the way note energy decays after an onset, so a transcription model needs to adapt to a note with varying amplitude and harmonics. Nonnegative matrix factorization (NMF) is an

early popular method used in the task of polyphonic music transcription [19]. With recent advancements in deep learning, neural networks have attracted more and more attention from the AMT community [13, 18]. In particular, the success of convolutional neural networks (CNN) for image classification tasks [21] has inspired the use of CNNs for AMT because two-dimensional time-frequency representations (e.g., constant-Q transform [5]) are common input representations for audio. In [13], the authors demonstrated the potential for a single CNN-based acoustic model to accomplish polyphonic piano music transcription. [18] considered an approach inspired by common models used in speech recognition where a CNN acoustic model and a Recurrent Neural Network (RNN) language model are combined. In this paper, we investigate improving the acoustic model by focusing on note onsets.

Note onset detection looks for only the very beginning of a note. Intuitively, the beginning of a piano note is easier to identify because the amplitude of that note is at its peak. For piano notes, the onset is also percussive and has a distinctive broadband spectrum. Once the model has determined onset events, we can condition framewise note detection tasks on this knowledge. Previously, [6, 27] demonstrated the promise of modeling onset events explicitly in both NMF and CNN frameworks. In this work, we demonstrate that a model conditioned on onsets achieves state of the art performance by a large margin for all common metrics measuring transcription quality: frame, note, and note-with-offset.

We also extend our model to predict the relative velocity of each onset. Velocity captures the speed with which a piano key was depressed and is directly related to how loud that note sounds. Including velocity information in a transcription is critical for describing the expressivity of a piano performance and results in much more natural-sounding transcriptions.

2. DATASET AND METRICS

We use the MAPS dataset [9] which contains audio and corresponding annotations of isolated notes, chords, and complete piano pieces. Full piano pieces in the dataset consist of both pieces rendered by software synthesizers and recordings of pieces played by a Yamaha Disklavier player piano. We use the set of synthesized pieces as the training split and the set of pieces played on the Disklavier as the test split, as proposed in [18]. When constructing

^{*} Equal contribution.

[†] Work done as a Google Brain intern.



© Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, Douglas Eck. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, Douglas Eck. "Onsets and Frames: Dual-Objective Piano Transcription", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

these datasets, we also ensured that the same music piece was not present in more than one set. Not including the Disklavier recordings, individual notes, or chords in the training set is closer to a real-world testing environment because we often do not have access to recordings of a testing piano at training time. Testing on the Disklavier recordings is also more realistic because many of the recordings that are most interesting to transcribe are ones played on real pianos.

When processing the MAPS MIDI files for training and evaluation, we first translate “sustain pedal” control changes into longer note durations. If a note is active when sustain goes on, that note will be extended until either sustain goes off or the same note is played again. This process gives the same note durations as the text files included with the dataset.

The metrics used to evaluate a model are frame-level and note-level metrics including precision, recall, and F1 score. We use the *mir_eval* library [16] to calculate note-based precision, recall, and F1 scores. As is standard, we calculate two versions of note metrics: one requiring that onsets be within ± 50 ms of ground truth but ignoring offsets and one that also requires offsets resulting in note durations within 20% of the ground truth or within 50ms, whichever is greater. Frame-based scores are calculated using the standard metrics as defined in [2]. We also introduce a new note metric for velocity transcription that is further described in Section 3.1. Both frame and note scores are calculated per piece and the mean of these per-piece scores is presented as the final metric for a given collection of pieces.

Our goal is to generate piano transcriptions that contain all perceptually relevant performance information in an audio recording without prior information about the recording environment such as characterization of the instrument. We need a numerical measure that correlates with this perceptual goal. Poor quality transcriptions can still result in high frame scores due to short spurious notes and repeated notes that should be held. Note onsets are important, but a piece played with only onset information would either have to be entirely staccato or use some kind of heuristic to determine when to release notes. A high note-with-offset score will correspond to a transcription that sounds good because it captures the perceptual information from both onsets and durations. Adding a velocity requirement to this metric ensures that the dynamics of the piece are captured as well. More perceptually accurate metrics may be possible and warrant further research. In this work we focus on improving the note-with-offset score, but also achieve state of the art results for the more common frame and note scores and extend the model to transcribe velocity information as well.

3. MODEL CONFIGURATION

Framewise piano transcription tasks typically process frames of raw audio and produce frames of note activations. Previous framewise prediction models [13, 18] have treated frames as both independent and of equal impor-

tance, at least prior to being processed by a separate language model. We propose that some frames are more important than others, specifically the onset frame for any given note. Piano note energy decays starting immediately after the onset, so the onset is both the easiest frame to identify and the most perceptually significant.

We take advantage of the significance of onset frames by training a dedicated note onset detector and using the raw output of that detector as additional input for the framewise note activation detector. We also use the thresholded output of the onset detector during the inference process, similar to concurrent research described in [24]. An activation from the frame detector is only allowed to start a note if the onset detector agrees that an onset is present in that frame.

Our onset and frame detectors are built upon the convolution layer acoustic model architecture presented in [13], with some modifications. We use *librosa* [15] to compute the same input data representation of mel-scaled spectrograms with log amplitude of the input raw audio with 229 logarithmically-spaced frequency bins, a hop length of 512, an FFT window of 2048, and a sample rate of 16kHz. We present the network with the entire input sequence, which allows us to feed the output of the convolutional frontend into a recurrent neural network (described below).

The onset detector is composed of the acoustic model, followed by a bidirectional LSTM [17] with 128 units in both the forward and backward directions, followed by a fully connected sigmoid layer with 88 outputs for representing the probability of an onset for each of the 88 piano keys.

The frame activation detector is composed of a separate acoustic model, followed by a fully connected sigmoid layer with 88 outputs. Its output is concatenated together with the output of the onset detector and followed by a bidirectional LSTM with 128 units in both the forward and backward directions. Finally, the output of that LSTM is followed by a fully connected sigmoid layer with 88 outputs. During inference, we use a threshold of 0.5 to determine whether the onset detector or frame detector is active.

Training RNNs over long sequences can require large amounts of memory and is generally faster with larger batch sizes. To expedite training, we split the training audio into smaller files. However, when we do this splitting we do not want to cut the audio during notes because the onset detector would miss an onset while the frame detector would still need to predict the note’s presence. We found that 20 second splits allowed us to achieve a reasonable batch size during training of at least 8, while also forcing splits in only a small number of places where notes are active. When notes are active and we must split, we choose a zero-crossing of the audio signal. Inference is performed on the original and un-split audio file.

Our ground truth note labels are in continuous time, but the results from audio processing are in spectrogram frames. So, we quantize our labels to calculate our training loss. When quantizing, we use the same frame size as

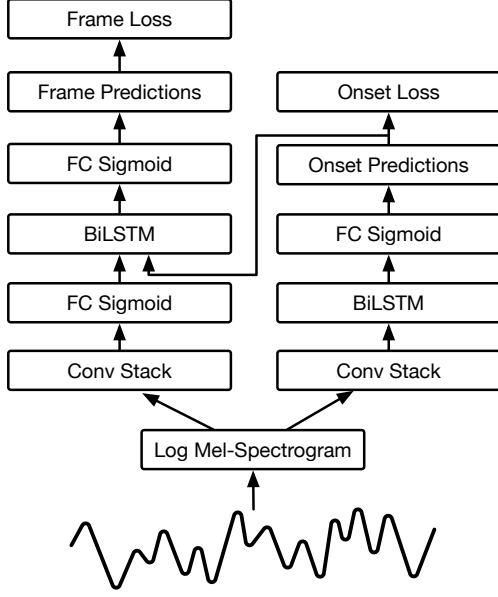


Figure 1. Diagram of Network Architecture

the output of the spectrogram. However, when calculating metrics, we compare our inference results against the original, continuous time labels.

Our loss function is the sum of two cross-entropy losses: one from the onset side and one from the note side.

$$L_{total} = L_{onset} + L_{frame} \quad (1)$$

$$L_{onset} = \sum_{p=p_{min}}^{p_{max}} \sum_{t=0}^T CE(\mathbf{I}_{onset}(p, t), \mathbf{P}_{onset}(p, t)) \quad (2)$$

where p_{min}/p_{max} denote the MIDI pitch range of the piano roll, T is the number of frames in the example, $\mathbf{I}_{onset}(p, t)$ is an indicator function that is 1 when there is a ground truth onset at pitch p and frame t , $\mathbf{P}_{onset}(p, t)$ is the probability output by the model at pitch p and frame t and CE denotes cross entropy. The labels for the onset loss are created by truncating note lengths to $\min(\text{note.length}, \text{onset.length})$ prior to quantization. We performed a coarse hyperparameter search over onset.length (we tried 16, 32 and 48ms) and found that 32ms worked best. In hindsight this is not surprising as it is also the length of our frames and so almost all onsets will end up spanning exactly two frames. Labeling only the frame that contains the exact beginning of the onset does not work as well because of possible mis-alignments of the audio and labels. We experimented with requiring a minimum amount of time a note had to be present in a frame before it was labeled, but found that the optimum value was to include any presence.

In addition, within the frame-based loss term L_{frame} , we apply a weighting to encourage accuracy at the start of the note. A note starts at frame t_1 , completes its onset at t_2 and ends at frame t_3 . Because the weight vector assigns higher weights to the early frames of notes, the model is incentivized to predict the beginnings of notes accurately,

thus preserving the most important musical events of the piece. First, we define a raw frame loss as:

$$L_{frame} = \sum_{p=p_{min}}^{p_{max}} \sum_{t=0}^T CE(\mathbf{I}_{frame}(p, t), \mathbf{P}_{frame}(p, t)) \quad (3)$$

where $\mathbf{I}_{frame}(p, t)$ is 1 when pitch p is active in the ground truth in frame t and $\mathbf{P}_{frame}(p, t)$ is the probability output by the model for pitch p being active at frame t . Then, we define the weighted frame loss as:

$$L_{frame}(l, p) = \begin{cases} cL'_{frame}(l, p) & t_1 \leq t \leq t_2 \\ \frac{c}{t-t_2}L'_{frame}(l, p) & t_2 < t \leq t_3 \\ L'_{frame}(l, p) & \text{elsewhere} \end{cases} \quad (4)$$

where $c = 5.0$ as determined with coarse hyperparameter search.

3.1 Velocity Estimation

We further extend the model by adding another stack to also predict velocities for each onset. This stack is similar to the others and consists of the same layers of convolutions. This stack does not connect to the other two. The velocity labels are generated by dividing all the velocities by the maximum velocity present in the piece. The smallest velocity does not go to zero, but rather to $\frac{v_{min}}{v_{max}}$. The stack is trained with the following loss averaged across a batch:

$$L_{vel} = \sum_{p=p_{min}}^{p_{max}} \sum_{t=0}^T \mathbf{I}_{onset}(p, t) (v_{label}^{p,t} - v_{predicted}^{p,t})^2 \quad (5)$$

At inference time the output is clipped to $[0, 1]$ and then transformed to a midi velocity by the following mapping:

$$v_{midi} = 80v_{predicted} + 10 \quad (6)$$

The final mapping is arbitrary, but we found this leads to pleasing audio renderings.

While various studies have considered the estimation of dynamics (note intensities or velocities) in a recording given the score [10, 22, 26], to our knowledge there has been no work in the literature considering estimation of dynamics alongside pitch and timing information. As a result, as Benetos et al. [3] noted in their review paper in 2013, “evaluating the performance of current [automatic music transcription] systems for the estimation of note dynamics has not yet been addressed.” To evaluate our velocity-aware model, we therefore propose an additional criterion for the note-level precision, recall, and F1 scores.

Evaluating velocity predictions is not straightforward because unlike pitch and timing, velocity has no absolute meaning. For example, if two transcriptions contained identical velocities except that they were offset or scaled by a constant factor, they would be effectively equivalent despite reporting completely different velocities for every note. To address these issues, we first re-scale all of the ground-truth velocities in a transcription to be in the range

[0, 1]. After notes are matched according to their pitch and onset/offset timing, we assemble pairs of the reference (ground-truth) and estimated velocities for matched notes, referred to as v_r and v_e respectively. We then perform a linear regression to estimate a global scale and offset parameter such that the squared difference between pairs of reference and estimated velocities is minimized:

$$m, b = \arg \min_{m, b} \sum_{i=1}^M \|v_r(i) - (mv_e(i) + b)\|^2 \quad (7)$$

where M is the number of matches (i.e. number of entries in v_r and v_e). These scalar parameters are used to re-scale the entries of v_e to obtain

$$\hat{v}_e = \{mv_e(i) + b, i \in 1, \dots, M\} \quad (8)$$

Finally, a match i is now only considered correct if, in addition to having its pitch and timing match, it also satisfies $|\hat{v}_e(i) - v_r(i)| < \tau$ for some threshold τ . We used $\tau = 0.1$ in all of our evaluations. The precision, recall, and F1 scores are then recomputed as normal based on this newly filtered list of matches.

4. EXPERIMENTS

We trained our onsets and frames model using TensorFlow [1] on the training dataset described in Section 2 using a batch size of 8, a learning rate of .0006, and a gradient clipping L2-norm of 3. A hyperparameter search was conducted to find the optimal learning rate. We use the Adam optimizer [14] and train for 50,000 steps. Training takes 5 hours on 3 P100 GPUs. The same hyperparameters were used to train all models, including those from the ablation study, except when reproducing the results of [18] and [13], where hyperparameters from the respective papers were used. The source code for our model is available at <https://goo.gl/magenta/onsets-frames-code>.

For comparison, we reimplemented the models described in [13, 18] to ensure evaluation consistency. We also compared against the commercial software Melodyne version 4.1.1.011¹. We would have liked to compare against AnthemScore² as described in [25] as well, but because it produces a MusicXML score with quantized note durations instead of a MIDI file with millisecond-scale timings, an accurate comparison was not possible.

Results from these evaluations are summarized in Table 1. Our onsets and frames model not only produces better note-based scores (which only take into account onsets), it also produces the best frame-level scores and note-based scores that include offsets.

An example input spectrogram, note and onset output posteriorgrams, and inferred transcription for a recording from outside of the training set is shown in Figure 2. The importance of restricting frame activations based on onset predictions during inference is clear: The second-to-bottom image (“Estimated Onsets and Notes”) shows the

results from the frame and onset predictors. There are several examples of notes that either last for only a few frames or that reactivate briefly after being active for a while. Frame results after being restricted by the onset detector are shown in magenta. Many of the notes that were active for only a few frames did not have a corresponding onset detection and were removed, shown in cyan. Cases where a note briefly reactivated were also removed because a corresponding second onset was not detected.

Despite not optimizing for inference speed, our network performs 70× faster than real time on a Tesla K40c. The MIDI files resulting from our inference experiments are available at <https://goo.gl/magenta/onsets-frames-examples>.

5. ABLATION STUDY

To understand the individual importance of each piece in our model, we conducted an ablation study. We consider removing the onset detector entirely (i.e., using only the frame detector) (a), not using the onset information during inference (b), making the bi-directional RNNs uni-directional (c,d), removing the RNN from the onset detector entirely (e), pre-training the onset detector rather than jointly training it with the frame detector (f), weighting all frames equally (g), sharing the convolutional features between both detectors (h), removing the connection between the onset and frame detectors during training (i), using a Constant Q-Transform (CQT) input representation instead of mel-scaled spectrograms (j), and finally removing all the LSTMs and sharing the convolutional features (k).

These results show the importance of the onset information – not using the onset information during inference (b) results in a significant 18% relative decrease in the note onset score and a 31% relative decrease in the note-with-offset score while increasing the frame score slightly. Despite the increased frame score, the output sounds significantly worse. To our ears, the decrease in transcription quality is best reflected by the note-with-offset scores.

The model which does not have the onset detector at all (a) – consisting of convolutions followed by a bi-directional RNN followed by a frame-wise loss – does the worst on all metrics, although it still outperforms the baseline model from [13]. The other ablations indicate a small impact for each component (< 6%). It is encouraging that forward-only RNNs have only a small accuracy impact as they can be used for online piano transcription.

We tried many other architectures and data augmentation strategies not listed in the table, none of which resulted in any improvement. Significantly, augmenting the training audio by adding normalization, reverb, compression, noise, and synthesizing the training MIDI files with other synthesizers made no difference. We believe these results indicate a need for a much larger training dataset of real piano recordings that have fully accurate label alignments. These requirements are not satisfied by the current MAPS dataset because only 60 of its 270 recordings are from real pianos, and they are also not satisfied by MusicNet [23] because its alignments are not fully accurate (e.g.,

¹ <http://www.celemony.com/en/melodyne>

² <https://www.lunaverus.com/>

	Frame			Note			Note w/ offset			Note w/ offset & velocity		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Sigtia et al., 2016 [18]	71.99	73.32	72.22	44.97	49.55	46.58	17.64	19.71	18.38	—	—	—
Kelz et al., 2016 [13]	81.18	65.07	71.60	44.27	61.29	50.94	20.13	27.80	23.14	—	—	—
Melodyne (decay mode)	71.85	50.39	58.57	62.08	48.53	54.02	21.09	16.56	18.40	10.43	8.15	9.08
Onsets and Frames	88.53	70.89	78.30	84.24	80.67	82.29	51.32	49.31	50.22	35.52	30.80	35.39

Table 1. Precision, Recall, and F1 Results on MAPS configuration 2 test dataset (ENSTDkCl and ENSTDkAm full-length .wav files). Note-based scores calculated by the *mir_eval* library, frame-based scores as defined in [2]. Final metric is the mean of scores calculated per piece. MIDI files used to calculate these scores are available at <https://goo.gl/magenta/onsets-frames-examples>.

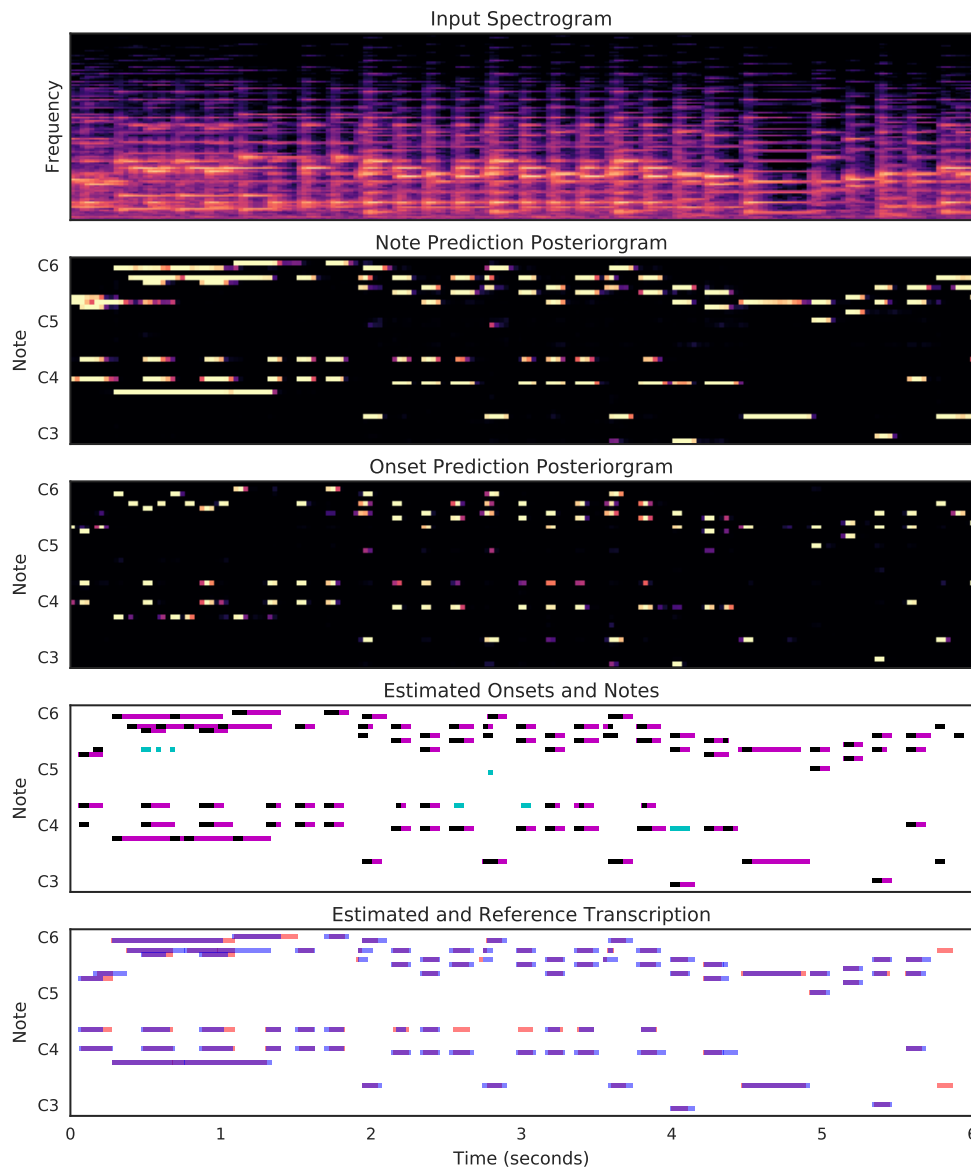


Figure 2. Inference on 6 seconds of MAPS.MUS-mz_331_3.ENSTDkCl.wav (a recording which is not in the training set). From top to bottom, we show the log-magnitude mel-frequency spectrogram input, the frame-wise note probability and onset probability “posteriorgrams” produced by our model, the corresponding estimated onsets and notes after thresholding, and finally the resulting estimated transcription produced by our model alongside the reference transcription. In the onset and notes plot (second from the bottom), onset predictions are shown in black. Notes with a corresponding onset prediction are shown in magenta and notes which are filtered out because no onset was predicted for the note are shown in cyan. In the bottom plot, the estimated transcription is shown in blue and the reference is shown in red. Figure best viewed in color.

there is an audible time difference between piano audio and MIDI at 1:24 in sequence 2533). Other approaches, such as seq2seq [20] may not require fully accurate alignments.

	F1		
	Frame	Note	Note with offset
Onset and Frames	78.30	82.29	50.22
(a) Frame-only LSTM	76.12	62.71	27.89
(b) No Onset Inference	78.37	67.44	34.15
(c) Onset forward LSTM	75.98	80.77	46.36
(d) Frame forward LSTM	76.30	82.27	49.50
(e) No Onset LSTM	75.90	80.99	46.14
(f) Pretrain Onsets	75.56	81.95	48.02
(g) No Weighted Loss	75.54	80.07	48.55
(h) Shared conv	76.85	81.64	43.61
(i) Disconnected Detectors	73.91	82.67	44.83
(j) CQT Input	73.07	76.38	41.14
(k) No LSTM, shared conv	67.60	75.34	37.03

Table 2. Ablation Study Results.

6. NEED FOR MORE DATA, MORE RIGOROUS EVALUATION

The most common dataset for evaluation of piano transcription tasks is the MAPS dataset, in particular the ENSTDkCl and ENSTDkAm renderings of the MUS collection of pieces. This set has several desirable properties: the pieces are real music as opposed to randomly-generated sequences, the pieces are played on a real physical piano as opposed to a synthesizer, and multiple recording environments are available (“close” and “ambient” configurations). The main drawback of this dataset is that it contains only 60 recordings. To best measure transcription quality, we believe a new and much larger dataset is needed. However, until that exists, evaluations should make full use of the data that is currently available.

Many papers, for example [7, 12, 18, 27], further restrict the data used in evaluation by using only the “close” collection and/or only the first 30 seconds or less of each file. We believe this method results in an evaluation that is not representative of real-world transcription tasks. For example, evaluating on only the “close” collection raises our note F1 score from 82.29 to 84.34, and evaluating on only the first 30 seconds further raises it to 86.38. For comparison, [27] achieved a note F1 score of 80.23 in this setting. The model in [12] is also evaluated using 30s clips from the “close” collection, but it was additionally trained on data from the test piano. This method limits the generalizability of the model but produced a note F1 score of 85.06.

In addition to the small number of the MAPS Disklavier recordings, we have also noticed several cases where the Disklavier appears to skip some notes played at low velocity. For example, at the beginning of the Beethoven Sonata No. 9, 2nd movement, several Ab notes played with MIDI velocities in the mid-20s are clearly missing from the audio (<https://goo.gl/magenta/onsets-frames-examples>). More analysis is needed to determine how frequently missed notes

occur, but we have noticed that our model performs particularly poorly on notes with ground truth velocities below 30.

Finally, we believe that more strict metrics should be adopted by the community. As discussed in Section 2, frame and note onset scores are not enough to determine whether a transcription has captured all musically relevant information from a performance. We present several audio examples at <https://goo.gl/magenta/onsets-frames-examples> to illustrate this point. Of the metrics currently available, we believe that the note-with-offset and velocity is the best way to compare models going forward.

Similarly, the current practice of using a 50ms tolerance for note onset correctness allows for too much timing jitter. An audio example illustrating this point is available at the above URL. We suggest future work should evaluate models with tighter timing requirements. Much work remains to be done here because as observed in [4], achieving high accuracy is increasingly difficult as timing precision is increased, in part due to the limited timing accuracy of the datasets currently available [11]. When we trained our model at a resolution of 24ms, our scores using the existing 50ms metrics were not always as high: Frame 76.87, Note F1 82.54, Note-with-offset 49.99. Audio examples of this higher resolution model are also available at the above URL. In the examples, the higher time resolution is evident, but the model also produces more extraneous notes.

7. CONCLUSION AND FUTURE WORK

We presented a jointly-trained onsets and frames model for transcribing polyphonic piano music which yields significant improvements by using onset information. This model transfers well between the disparate train and test distributions. The current quality of our model’s output is on the cusp of enabling downstream applications such as symbolic MIR and automatic music generation. To further improve the results we need to create a new dataset that is much larger and more representative of various piano recording environments and music genres for both training and evaluation. Combining an improved acoustic model with a language model is a natural next step. Another direction is to go beyond traditional spectrogram representations of audio signals.

It is very much worth listening to the examples of transcription. Consider Mozart Sonata K331, 3rd movement. Our system does a good job in terms of capturing harmony, melody, rhythm, and even dynamics. If we compare this transcription to the other systems, the difference is quite audible. We have also successfully used the model to transcribe recordings from the Musopen.org website that are completely unrelated to our training dataset. The model even works surprisingly well transcribing a harpsichord recording. Audio examples are available at <https://goo.gl/magenta/onsets-frames-examples>.

8. ACKNOWLEDGEMENTS

We would like to thank Hans Bernhard for helping with data collection and Brian McFee and Justin Salamon for consulting on the velocity metric design.

9. REFERENCES

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
- [2] Mert Bay, Andreas F Ehmann, and J Stephen Downie. Evaluation of multiple-f0 estimation and tracking systems. In *ISMIR*, pages 315–320, 2009.
- [3] Emmanouil Benetos, Simon Dixon, Dimitrios Gianneoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.
- [4] Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *Acoustics, speech and signal processing (ICASSP), 2012 IEEE international conference on*, pages 121–124. IEEE, 2012.
- [5] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.
- [6] Tian Cheng, Matthias Mauch, Emmanouil Benetos, Simon Dixon, et al. An attack/decay model for piano transcription. In *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*. ISMIR, 2016.
- [7] Andrea Cogliati, Zhiyao Duan, and Brendt Wohlberg. Piano transcription with convolutional sparse lateral inhibition. *IEEE Signal Processing Letters*, 24(4):392–396, 2017.
- [8] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, pages 637–642, 2010.
- [9] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.
- [10] Sebastian Ewert and Meinard Müller. Estimating note intensities in music recordings. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 385–388, 2011.
- [11] Sebastian Ewert and Mark Sandler. Piano transcription in the studio using an extensible alternating directions framework. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):1983–1997, 2016.
- [12] Lufei Gao, Li Su, Yi-Hsuan Yang, and Tan Lee. Polyphonic piano note transcription with non-negative matrix factorization of differential spectrogram. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 291–295. IEEE, 2017.
- [13] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. *arXiv preprint arXiv:1612.05153*, 2016.
- [14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [15] Brian McFee. librosa: v0.4.3, May 2016.
- [16] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. mir_eval: A transparent implementation of common mir metrics. In *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.
- [17] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
- [18] Siddharth Sigita, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(5):927–939, 2016.
- [19] Paris Smaragdis and Judith C Brown. Non-negative matrix factorization for polyphonic music transcription. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on*, pages 177–180. IEEE, 2003.
- [20] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
- [21] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.
- [22] Wai Man Szeto, Kin Hong Wong, and Chi Hang Wong. Finding intensities of individual notes in piano music. In *Computer Music Modeling and Retrieval*, 2005.

- [23] John Thickstun, Zaid Harchaoui, and Sham Kakade. Learning features of music from scratch. In *International Conference on Learning Representations (ICLR)*, 2017.
- [24] Carl Thomé and Sven Ahlbäck. Polyphonic pitch detection with convolutional recurrent neural networks. *Music Information Retrieval Evaluation eXchange (MIREX)*, 2017.
- [25] Daylin Troxel. Music transcription with a convolutional neural network 2016. *Music Information Retrieval Evaluation eXchange (MIREX)*, 2016.
- [26] Sam Van Herwaarden, Maarten Grachten, and Bas De Haas. Predicting expressive dynamics in piano performances using neural networks. In *Proceedings of the 15th Conference of the International Society for Music Information Retrieval*, pages 45–52, 2014.
- [27] Qi Wang, Ruohua Zhou, and Yonghong Yan. A two-stage approach to note-level transcription of a specific piano. *Applied Sciences*, 7(9):901, 2017.

PLAYER VS TRANSCRIBER: A GAME APPROACH TO DATA MANIPULATION FOR AUTOMATIC DRUM TRANSCRIPTION

Carl Southall, Ryan Stables and Jason Hockman

DMT Lab, Birmingham City University, Birmingham, United Kingdom

{carl.southall, ryan.stables, jason.hockman}@bcu.ac.uk

ABSTRACT

State-of-the-art automatic drum transcription (ADT) approaches utilise deep learning methods reliant on time-consuming manual annotations and require congruence between training and testing data. When these conditions are not held, they often fail to generalise. We propose a game approach to ADT, termed player vs transcriber (PvT), in which a player model aims to reduce transcription accuracy of a transcriber model by manipulating training data in two ways. First, existing data may be augmented, allowing the transcriber to be trained using recordings with modified timbres. Second, additional individual recordings from sample libraries are included to generate rare combinations. We present three versions of the PvT model: AugExist, which augments pre-existing recordings; AugAddExist, which adds additional samples of drum hits to the AugExist system; and Generate, which generates training examples exclusively from individual drum hits from sample libraries. The three versions are evaluated alongside a state-of-the-art deep learning ADT system using two evaluation strategies. The results demonstrate that including the player network improves the ADT performance and suggests that this is due to improved generalisability. The results also indicate that although the Generate model achieves relatively low results, it is a viable choice when annotations are not accessible.

1. INTRODUCTION

Automatic music transcription (AMT) systems generate a symbolic representation of the instrumentation within an audio recording. There are multiple educational, analytical and creative fields that would benefit from fast and accurately produced music notation. Automatic drum transcription (ADT) systems form a subset of AMT systems which produce notation solely focused on drum instrumentation.

1.1 Background

At present, high ADT accuracies have been achieved for audio files containing either just drums or polyphonic mixtures [4, 7, 9–11, 18]. Following the comprehensive

ADT literature review in [22], current state-of-the-art ADT systems utilise either deep learning (DL) or non-negative matrix factorisation (NMF). NMF approaches perform instrument-specific onset detection through an iterative simultaneous update of basis and activation functions via factorisation of an input spectrogram. Recent NMF approaches have introduced specialised update methods (i.e., fixed, adaptive and semi-adaptive) based on the expected end-use application of the algorithm [2] or incorporated additional basis functions to capture harmonic content within polyphonic recordings [23]. Alternatively, DL approaches perform instrument-specific onset detection through supervised frame-based classification. The first DL systems to achieve high ADT performance incorporated recurrent neural networks [14, 19, 20]. More recent approaches now include convolutional neural networks [21] and soft attention mechanisms [15]. As in many fields, augmentation of data (i.e., pitch shifting) during training has aided performance [12, 20].

1.2 Motivation

Evaluations undertaken in [22] and the MIREX 2017 Drum Transcription Task¹ highlight that state-of-the-art ADT accuracies are achieved by supervised DL approaches. However, success of DL systems is reliant on training data achieved through a time-consuming manual annotation process [24]. Also, if there is a mismatch between training and testing data, these systems will fail to generalise [22]. We propose a game approach to ADT influenced by generative adversarial networks [5], termed *player vs transcriber* (PvT). In an attempt to undermine the accuracy of a transcriber model (i.e., an existing supervised ADT approach), a player model seeks to exploit poorly defined areas of the feature space through a manipulation of training data. This is achieved through learned data manipulation variables in the player network, which are used to define the manipulation coordinates of the transform. Additionally, the player model is able to manipulate the data depending on its content, where existing methods for augmentation typically rely on a set of global variables [8].

The remainder of this paper is structured as follows: Section 2 presents the PvT model and Section 3 provides an overview of the undertaken evaluation. The results and discussion are presented in Section 4 and the conclusions and future work are presented in Section 5.



© Carl Southall, Ryan Stables and Jason Hockman. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Carl Southall, Ryan Stables and Jason Hockman. “Player Vs Transcriber: A Game Approach To Data Manipulation For Automatic Drum Transcription”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ http://www.music-ir.org/mirex/wiki/2017:Drum_Transcription_Results

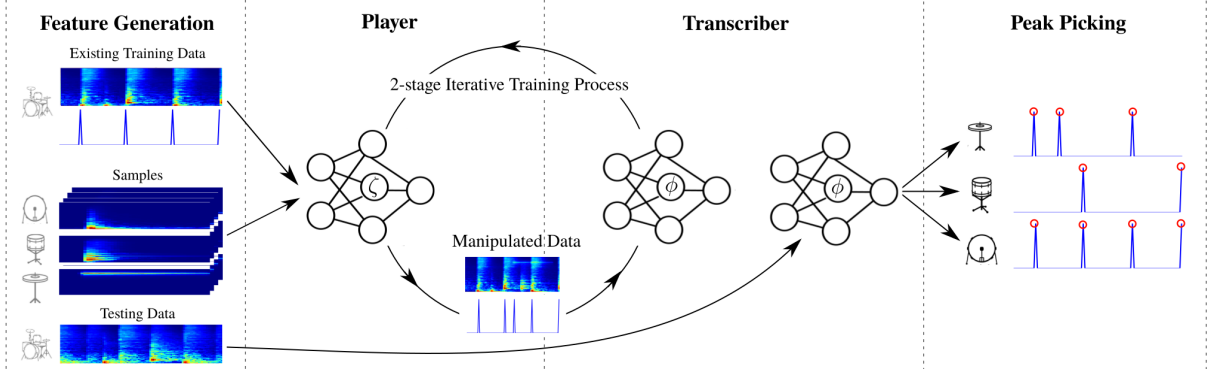


Figure 1. Overview of the player vs transcriber (PvT) game approach to automatic drum transcription. The PvT model is achieved through four stages: feature generation, player model, transcriber model and peak-picking.

2. METHOD

Figure 1 provides an overview of the proposed PvT system, which is achieved through four stages: Feature generation, data manipulation by a player, activation function creation by a transcriber, and framewise classification by peak picking. At the core of the system is an iterative process of data manipulation (i.e., data augmentation and sample addition) and activation function generation between the player and transcriber. Both models examine the loss functions related to the activation functions created by the transcriber. Here, the two models have diametrically opposed goals; while the player seeks to maximize the loss, the transcriber attempts to minimize the loss. Once the loss function has been optimized during training, testing data may be evaluated by the transcriber and drum events are found through peak picking.

2.1 Feature Generation

Input audio (16-bit .wav file sampled at 44.1kHz) is segmented into T frames using a Hanning window of m samples ($m = 2048$) with a $\frac{m}{2}$ hopsize. A logarithmic frequency representation of each of the frames is created using a similar process to [21] using the madmom Python library [1]. The magnitudes of a discrete Fourier transform are converted to a logarithmic scale (20Hz–20kHz) using twelve triangular filters per octave, resulting in a $84 \times T$ logarithmic spectrogram x and corresponding target y .

2.2 Player

The aim of the player is to exploit weaknesses within the transcriber through a manipulation of the training data. This is achieved using two processes as demonstrated in Figure 2: data augmentation (Section 2.2.1), which alters the frequential content of existing data; and sample addition (Section 2.2.2), which adds recordings of individual drum hits from drum samples (Section 2.2.3) to the pre-existing training examples. To ensure that the process is *end-to-end* and that the player network can be trained using back propagation, the process must be differentiable. To this end, the entire process is designed around network defined variables θ (Sections 2.2.4 and 2.2.5) and avoids operations such as *argmax*.

2.2.1 Data Augmentation

The data augmentation stage is based on existing data augmentation approaches [8], however also aims to portray changes in instrumentation and performance techniques by manipulating the frequency content of pre-existing data. This is achieved using three network-generated variables (θ^p , θ^n and θ^g), in which θ^p and θ^n are used within an pseudo-equaliser function and θ^g as an overall gain. Time-step t of the augmented segment x_{aug} is calculated using:

$$x_{aug}^t = \text{ReLU}(x^t + (s(\theta^p)x^tv) - (s(\theta^n)x^tv))g), \quad (1)$$

where v and s , the softmax functions are used to prevent over augmentation by limiting maximum augmentation to either 1 or -1. The rectified linear unit function (ReLU) ensures non-negativity and g is determined using:

$$g = \theta_g(1 - \min_g) + \min_g, \quad (2)$$

where \min_g is a hyperparameter that determines the minimum possible gain.

2.2.2 Sample Addition

The sample addition stage aims to reduce transcription accuracy by adding new drum hits to the augmented existing training data using drum samples c (Section 2.2.3). This is achieved by generating a new spectrogram q and corresponding target u and adding them to the existing augmented training spectrogram x_{aug} and target y :

$$x = x_{aug} + q_\omega, \quad (3)$$

$$y = y + u_\omega. \quad (4)$$

ω is the sample number and the total sample number hyperparameter Ω determines how many of each sample class are added. Each sample is added in an iterative process with the latest version of y and x_{aug} used in the update equations. To create q and u , four network determined variables are used: θ^{ps} , θ^{ns} and θ^{gs} which are variables used to augment each sample using the previously explained augmentation process and θ^l , which is used to determine the location of the additional drum samples. For all four network-determined variables, if $\Omega > 1$ a different variable is used for each sample (i.e., $\theta^l = [\theta_1^l, \theta_2^l]$). The

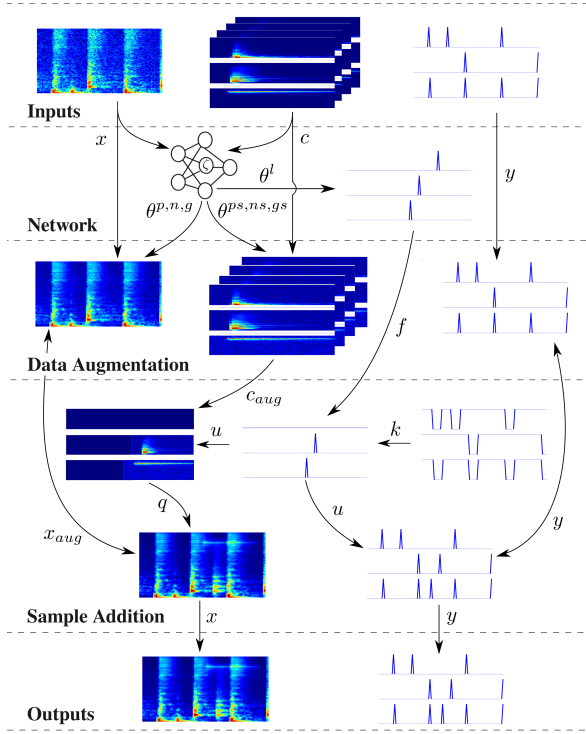


Figure 2. Overview of player model (Section 2.2) with a data augmentation stage that alters frequency content of pre-existing data and a sample addition stage that adds new drum hits based on generated locations.

generated target u is created using two variables: f , an activation function derived from the network determined parameter θ^l and k , a variable that ensures there are no overlaps between the same drum class. The activation function f for each individual drum sample is derived using:

$$i_\omega = \frac{\theta_\omega^l}{\max(\theta_\omega^l)}, \quad (5)$$

$$f_\omega = \text{ReLU}(i_\omega + \epsilon - \max(i_\omega)) \frac{1}{\epsilon}, \quad (6)$$

where ϵ is used to prevent undefined numbers. k is calculated from the current target y using a minimum distance between possible locations hyperparameter d :

$$h_\omega^t = \text{mean}(y^{t-d} : y^{t+d}), \quad (7)$$

$$n_\omega = \frac{\max(h_\omega)}{h_\omega} (h_\omega - \epsilon), \quad (8)$$

$$k_\omega = 1 - \text{ReLU}\left(\frac{n_\omega}{\max(n_\omega)}\right). \quad (9)$$

u is then generated by performing element wise multiplication on the two activation functions f and k :

$$u_\omega = k_\omega \odot f_\omega. \quad (10)$$

To calculate the new spectrogram q , a matrix consisting of all of the possible spectrograms for all T time steps e is created using:

$$z_\omega = \text{pad}(c_{aug\omega}, T, T), \quad (11)$$

$$c_\omega^t = z_\omega^{t+b} : z_\omega^{t+b+T}, \quad (12)$$

where c_{aug} is the augmented current sample spectrogram and $\text{pad}(c, 1, 1)$ means zero pad c by 1 in both directions in the time-step dimension. The spectrogram with the sample in the chosen location is then calculated using:

$$q_\omega = \sum_{t=1}^T c_\omega^t u_\omega^t. \quad (13)$$

It is worth noting that the proposed sample addition technique is capable of learning to not add any samples by putting the samples in locations where they overlap other locations and so will be removed.

2.2.3 Drum Samples

For drum samples (i.e., isolated drum events) to be utilised within the player model they must be segmented and undergo the same processing as the input features x . Segmentation of drum events from within larger audio files was achieved automatically through an automatic drum transcription method [13], and subsequently verified manually. Each sample is then cut to a pre-determined sample length with b frames before the onset. In this work a sample length of 50 is used with $b = 10$. The segmented samples are then converted to logarithmic spectrograms c through the process presented in Section 2.1. More information regarding the extraction of samples is given in Section 3.3.

2.2.4 Variations

From the augmentation and sample addition processes we create three different versions of the player model. The first version, termed *AugExist* augments existing training data using only the augmentation stage. The second version, termed *AugAddExist* uses both stages to augment existing training data and add drum samples to the augmented data. The final version, termed *Generate* generates entirely new training data by initializing x and y with zeros (i.e., no existing training data).

2.2.5 Player Network

The player neural network generates θ using a convolutional neural network consisting of two 3x3 kernel convolutional layers with max pooling, dropout [17] and batch normalisation [6], followed by a sigmoid fully connected output layer. The first convolutional layer is comprised of 5 channels and the second layer contains 10 channels. The size of the max pooling layer is altered depending on the player parameters (i.e., Ω , v , \min_g) so that the total number of trainable parameters of each model is comparable. The input features are the existing training data x and the different drum instrument samples c concatenated along the time dimensions. Throughout the remainder of this paper, all trainable player parameters are denoted as ζ .

2.3 Transcriber

The transcriber model follows the same system outline proposed in [14]. Input features are fed into a pre-trained neural network, which aims to output an activation function \tilde{y} with spikes in frames where onsets are located. In

this paper, we present a novel system that combines the strength of two recently proposed models. The soft attention mechanism presented in [15] is combined with the convolutional recurrent neural network proposed in [21] to create a convolutional recurrent neural network with a soft attention mechanism output layer. It contains two convolutional layers consisting of 3x3 filters, 3x3 max pooling, dropouts [17] and batch normalization [6], with the first layer consisting of 32 channels and the second is comprised of 64 channels. This is followed by two 20 neuron bidirectional recurrent neural network layers containing long short-term memory cells with peephole connections and a three-neuron sigmoid soft attention mechanism output layer. The attention number is set to 3 [15] and all other unstated variables are the same as the original implementations [15, 21]. Throughout the remainder of this paper, all trainable transcriber parameters are denoted as ϕ .

2.4 Peak Picking

Once the optimisation process is complete, the peak-picking stage classifies frames of \tilde{y} as either containing or not containing an onset. We use the mean threshold (MT) peak-picking technique from [15]. First a threshold τ^t is determined as:

$$\tau^t = \text{mean}(\tilde{y}^{t-\gamma} : \tilde{y}^{t+\gamma}) * \lambda \quad (14)$$

$$\tau^t = \begin{cases} tmax, & \tau > tmax \\ tmin, & \tau < tmin, \end{cases} \quad (15)$$

where λ is a constant, $tmax$ and $tmin$ are the possible maximum and minimum values and γ sets the number of frames used to calculate the mean. The current frame of \tilde{y} is accepted as an onset if it is the maximum of a surrounding number of frames and above the threshold τ :

$$O^t = \begin{cases} 1, & \tilde{y}^t == \max(\tilde{y}^{t-\delta} : \tilde{y}^{t+\delta}) \quad \& \quad \tilde{y}^t > \tau^t \\ 0, & \text{otherwise,} \end{cases} \quad (16)$$

where $O(t)$ represents an onset at time step t and δ is the number of frames on either side of the current frame t used to calculate the maximum.

2.5 Training

The player and transcriber are iteratively trained for one epoch each in a two-stage process, in which the player is first trained by updating ζ and then the transcriber is trained by updating ϕ . Cross entropy is used to minimise the loss function in both instances with $1 - y$ used as the player target and y used as the transcriber target. Both systems are trained using mini-batch gradient descent with the Adam optimiser and an initial learning rate of 0.003. These settings were determined using previous ADT studies [15] and also suited the player network well with no instability observed. Each mini-batch consists of 10 randomly chosen, 100 frame length segments. The data is divided into training, validation and test sets, with the training data used to optimize the systems and the validation used to prevent over fitting and to optimize the player and peak-picking parameters. Training is stopped if there has been no decrease in the transcriber validation loss after 10 epochs.

3. EVALUATION

To identify whether the PvT approach improves ADT performance, we compare it with the current state-of-the-art supervised ADT approach in six evaluation conditions, consisting of the three contexts and two evaluation strategies used in [22]. To determine whether similarity between drum samples and existing training data affects performance, two different sample libraries are utilised.

3.1 Contexts

For the first context, termed *drum transcription of drum-only recordings* (DTD) we utilise the IDMT-SMT-Drums dataset [2]. For the second context, termed *drum transcription in the presence of percussion* (DTP) we utilise the drum-only tracks within the ENST-Drums *minus one* subset [3] and MDB Drums [16]. The third context, termed *drum transcription in the presence of melodic instruments* (DTM), utilises the full polyphonic audio from the ENST-Drums *minus one* subset, MDB-Drums and RBMA-2013 [21].

3.2 Evaluation Strategies

The first strategy, termed *random*, utilises all of the context data and divides the tracks in to 70%, 15% and 15% training, validation and test subsets with three-fold cross validation. The second strategy, termed *subset*, utilises all data for the DTD context and only ENST-Drums for DTP and DTM. This strategy aims to test the generalisability of the systems by utilising the existing subsets within the datasets so that the training and testing data are unrelated.

3.3 Sample Usage

For drum samples, two sample libraries are used to test for the effect of similarity to existing training data. In addition to these, percussive mixtures are also generated using the content of the unobserved drum samples.

Training: The first library, termed *training*, only utilises samples extracted from the training data. For the IDMT-SMT-Drums dataset, a single sample for each observed drum instrument is extracted from each of the 104 tracks. For ENST-Drums the samples are extracted from the included isolated drum files, resulting in a total of 276 samples (21 KD, 146 SD and 109 HH). No samples are extracted from the other datasets; in the DTP and DTM cases the training sample library only consists of ENST-Drum samples.

Collection: The second sample library, termed *collection*, consists of drum samples collected from online resources. The collection samples are included as they represent a dataset with a wider diversity and are not included in the existing training data. In total, there are 445 samples (101 KD, 151 SD and 193 HH).

Polyphonic Instances: For the DTP versions of the Generate system, the player network output is combined with artificial percussive mixture segments created from other percussive samples (i.e., toms and cymbals) extracted

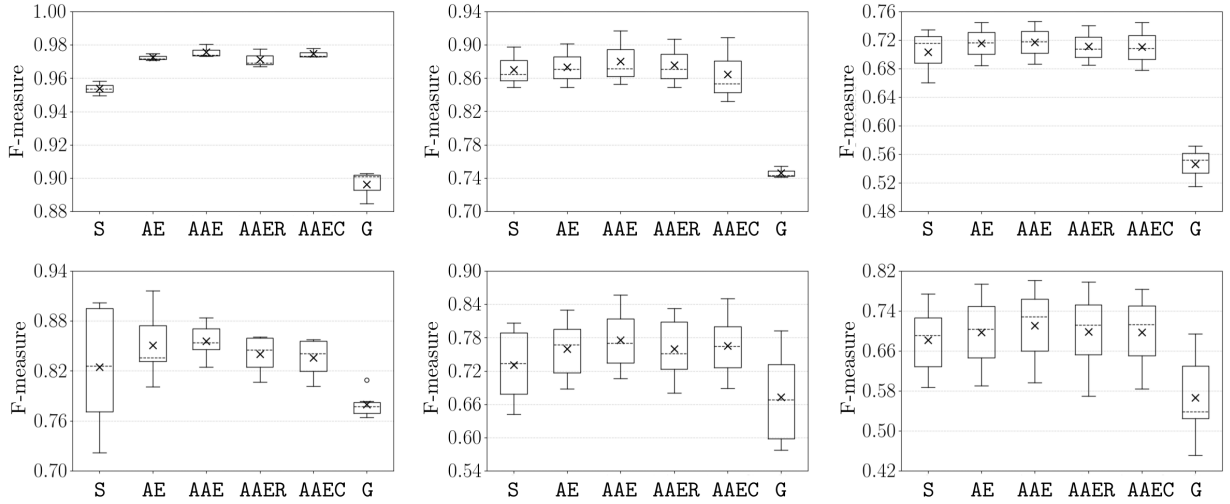


Figure 3. Results for random (top) and subset (bottom) strategies for DTD (left), DTP (middle) and DTM (right) contexts. Crosses denote mean instrument and mean fold F-measure, dashed lines present mean instrument and median fold F-measure and box plots present F-measure range across folds.

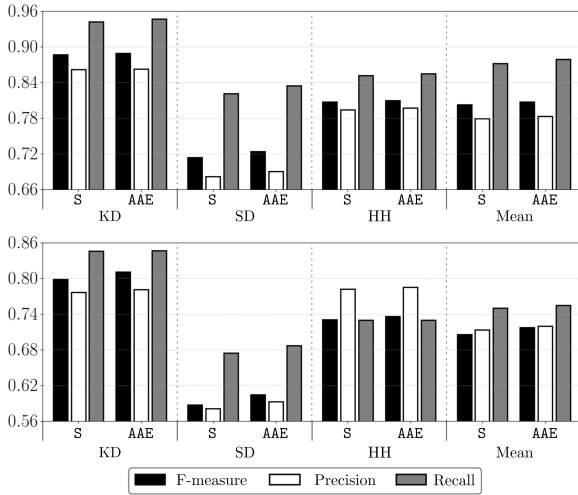


Figure 4. Individual and mean instrument F-measure, precision and recall scores for existing state-of-the-art system (S) and highest performing AugAddExist PvT model (AAE) using random (top) and subset (bottom) strategies.

from the ENST-Drums isolated files. For the DTM versions of the *Generate* system the player network output is combined with the artificial percussive mixtures as well as the accompaniment files from the ENST-Drums and MDB-Drums datasets.

3.4 Evaluation Methodology

In all evaluations, the three proposed PvT models—*AugExist* (AE), *AugAddExist* (AAE) and *Generate* (G)—are compared with the current state-of-the-art supervised ADT approach (S), which consists solely of the transcriber model. Additionally, two versions of the AAE system are evaluated: for the first (AAER), the θ parameters are set to the highest performing random values using a grid search

with the aim of portraying existing data augmentation techniques [8], and the second (AAEC) uses data from the collection samples alone (Section 3.3). The standard precision, recall and F-measure are used as evaluation metrics, with onset candidates being accepted if they fall within 30ms of the ground truth annotations. For all PvT systems d is set to 3 (approx. 30ms) b is set to 10 and hyperparameters v , min_g and Ω are optimized using grid search. To prevent either networks from overpowering the other, the player max pooling sizes are set so that the number of parameters ζ matches that of the transcriber network ϕ (approx. 100,000).

4. RESULTS AND DISCUSSION

4.1 Random and Subset

Figure 3 presents the random and subset results for the six implemented systems in the three contexts. The crosses represent the mean instrument F-measures and the box plots present the median and range across the folds. In all cases the trained versions of the *AugExist* (AE) and *AugAddExist* (AAE) systems achieve a higher mean F-measure and median F-measure than the existing state-of-the-art supervised method (S), with the box plots showing that this improvement is consistent in the majority of the folds. Results from t-tests across folds highlight that the improvements made by all *AugExist* and *AugAddExist* systems in the random DTD evaluation are significant (i.e., $p < 0.05$). Larger improvements are seen in the subset evaluation—designed specifically to test the generalisability of the systems [22]—which suggests that the PvT model does improve generalisability. For all PvT variations, the trained *AugAddExist* player versions (AAE) achieve higher accuracies than the grid search-set *AugAddExist* system (AAER). This demonstrates the worth of utilising a player network to learn the weaknesses of the transcriber model and its ability to manipulate each of the segments based on the content. Although the *Generate*

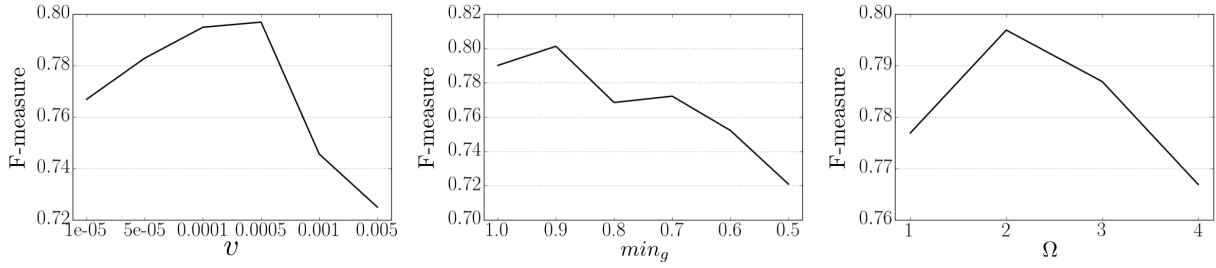


Figure 5. Mean fold, mean instrument and mean training strategy F-measure results of different AugAddExist settings for PvT model hyperparameters v (left), min_g (middle) and Ω (right).

systems do not achieve a higher F-measure they achieve a high accuracy relative to the amount of human input required within the process. Again, this is more apparent in the subset and easier contexts. The lower improvements achieved by the AugAddExist and Generate systems in the DTP and DTM contexts can be attributed to the limitation of samples from just the ENST dataset. This highlights that the more diverse the sample library, the larger the improvement in performance. However, the results from the system trained using the collection sample library (AAEC) show that too much diversity can cause the system to underfit, resulting in a slightly lower improvement being observed. The random- θ and collection sample library versions of the AugExist and Generate versions were also implemented but not included within the results as the trends are the same as the AAER and AAEC systems.

4.2 Individual Instrument

Figure 4 presents the mean fold, mean context individual and mean drum instrument F-measure, precision and recall scores for the supervised and highest performing AugAddExist systems. In all cases the AugAddExist system achieves higher F-measure precision and recall scores for all of the observed drum instruments with 0.015 increase in mean instrument F-measure in random and 0.035 increase in subset. The largest relative improvement is within the snare drum class, which further suggests the increase is due to greater generalisability, as it has proven to be the most difficult instrument to generalise [22].

4.3 Player Settings

Figure 5 presents mean instrument, mean fold and mean training strategy F-measure results for the AugAddExist system with different user defined hyperparameter settings. The left diagram presents results for different values of v , the middle diagram for different values of min_g and the right diagram different values of Ω . $v = 0.0005$ achieved the highest accuracies with lower values not allowing enough manipulation and higher values causing class overlap. $min_g = 0.9$ achieved the highest accuracy overall however, within the DTP and DTM contexts lower values achieved similar results. This is possibly due to the fact that a larger diversity of playing technique is present within those contexts. Adding a maximum of two extra drum hits ($\Omega = 2$) resulted in the highest accuracies with larger values causing too much overlap between instruments.

4.4 Understanding What The Player Does

By observing the player model training it is possible to gain an understanding of poorly defined areas of the feature space within ADT datasets. When the player performs data augmentation, a maximum amount of augmentation is selected most of the time (i.e., the output of the ReLU function in eq. 1 is close to either 1 or -1). This suggests that there are substantial gaps in coverage of training datasets in the feature space for the different instrumentation. Within the sample addition stage the player model consistently attempts to overlap drums with both other drum instruments and other instrumentation within the existing training data. This suggests that the datasets only contain limited observations of overlapping instrument combinations.

5. CONCLUSIONS AND FUTURE WORK

To overcome the requirement of time-consuming manual annotation, we proposed PvT, a game approach to automatic drum transcription. The player model is trained to alter the training data so that the accuracy of the transcriber model is reduced. The three implemented versions of the PvT model—AugExist (AE), AugAddExist (AAE) and Generate (G)—are evaluated alongside the existing supervised state-of-the-art ADT (S) and a grid search-set AugAddExist approach (AAER) using two evaluation strategies and three contexts. The results highlight that the trainable PvT model does improve ADT performance with AugAddExist achieving the highest accuracy in all evaluations. The Generate model also provides a viable option when annotated training data is not accessible. Although two approaches to alter the training data have been implemented, more are possible. Future work could explore trainable methods for moving existing drum events or synthesizing new drum samples within the player network, as there are no structural constraints on what the player can do. For polyphonic cases, the unobserved instrumentation could also be included within the player network so that further combinations can be generated. Another possible direction is to increase the number of observed drum instruments (i.e., including toms and crash cymbals), which is easily done using the Generate model. Open source versions of the PvT model are available online.²

² <https://github.com/CarlSouthall/Player-Vs-Transcriber>

6. REFERENCES

- [1] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: A new Python audio and music signal processing library. In *Proceedings of the ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 2016.
- [2] Christian Dittmar and Daniel Gärtner. Real-time transcription and separation of drum recordings based on NMF decomposition. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 187–194, Erlangen, Germany, 2014.
- [3] Olivier Gillet and Gaël Richard. Enst-drums: an extensive audio-visual database for drum signals processing. In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR)*, 2006.
- [4] Olivier Gillet and Gaël Richard. Transcription and separation of drum signals from polyphonic music. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):529–540, 2008.
- [5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.
- [7] Henry Lindsay-Smith, Skot McDonald, and Mark Sandler. Drumkit transcription via convolutive NMF. In *Proceedings of the International Conference on Digital Audio Effects Conference (DAFx)*, York, United Kingdom, 2012.
- [8] Brian McFee, Eric J Humphrey, and Juan Pablo Bello. A software framework for musical data augmentation. In *ISMIR*, pages 248–254, 2015.
- [9] Marius Miron, Matthew E. P. Davies, and Fabien Gouyon. An open-source drum transcription system for pure data and max MSP. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 221–225, Vancouver, Canada, 2013.
- [10] Axel Röbel, Jordi Pons, Marco Liuni, and Mathieu Lagrange. On automatic drum transcription using non-negative matrix deconvolution and Itakura-Saito divergence. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 414–418, Brisbane, Australia, 2015.
- [11] Mathias Rossignol, Mathieu Lagrange, Grégoire Lafay, and Emmanouil Benetos. Alternate level clustering for drum transcription. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 2023–2027, Nice, France, August 2015.
- [12] Carl Southall. MIREX 2017 drum transcription submissions. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.
- [13] Carl Southall, Nick Jillings, Ryan Stables, and Jason Hockman. ADTWeb: An open source browser based automatic drum transcription system. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.
- [14] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription using bi-directional recurrent neural networks. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 591–597, New York City, United States, 2016.
- [15] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 606–612, Suzhou, China, 2017.
- [16] Carl Southall, Chih-Wei Wu, Alexander Lerch, and Jason Hockman. MDB drums an annotated subset of medleyDB for automatic drum transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.
- [17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [18] Lucas Thompson, Simon Dixon, and Matthias Mauch. Drum transcription via classification of bar-level rhythmic patterns. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 187–192, Taipei, Taiwan, 2014.
- [19] Richard Vogl, Matthias Dorfer, and Peter Knees. Recurrent neural networks for drum transcription. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 730–736, New York City, United States, 2016.
- [20] Richard Vogl, Matthias Dorfer, and Peter Knees. Drum transcription from polyphonic music with recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 201–205, New Orleans, United States, 2017.
- [21] Richard Vogl, Matthias Dorfer, Gerhard Widmer, and Peter Knees. Drum transcription via joint beat and

- drum modeling using convolutional recurrent neural networks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 150–157, Suzhou, China, 2017.
- [22] Chih-Wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, Meinard Müller, and Alexander Lerch. A Review of Automatic Drum Transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(9):1457–1483, 2018.
- [23] Chih-Wei Wu and Alexander Lerch. Drum transcription using partially fixed non-negative matrix factorization with template adaptation. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 257–263, Malaga, Spain, 2015.
- [24] Chih-Wei Wu and Alexander Lerch. Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 613–620, Suzhou, China, 2017.

A FLEXIBLE APPROACH TO AUTOMATED HARMONIC ANALYSIS: MULTIPLE ANNOTATIONS OF CHORALES BY BACH AND PRÆTORIUS

Nathaniel Condit-Schultz
McGill University

Yaolong Ju
McGill University

Ichiro Fujinaga
McGill University

nathaniel.conda-schultz@mcgill.ca yaolong.ju@mail.mcgill.ca ichiro.fujinaga@mcgill.ca

ABSTRACT

Despite being a core component of Western music theory, *harmonic analysis* remains a subjective endeavor, resistant automation. This subjectivity arises from disagreements regarding, among other things, the interpretation of contrapuntal figures, the set of “legal” harmonies, and how harmony relates to more abstract features like tonal function. In this paper, we provide a formal specification of harmonic analysis. We then present a novel approach to computational harmonic analysis: rather than computing harmonic analyses based on one specific set of rules, we compute all possible analyses which satisfy only basic, uncontroversial constraints. These myriad interpretations can later be filtered to extract preferred analyses; for instance, to forbid 7th chords or to prefer analyses with fewer non-chord tones. We apply this approach to two concrete musical datasets: existing encodings of 371 chorales by J.S. Bach and new encodings of 200 chorales by M. Prætorius. Through an online API users can filter and download numerous harmonic interpretations of these 571 chorales. This dataset will serve as a useful resource in the study of harmonic/functional progression, voice-leading, and the relationship between melody and harmony, and as a stepping stone towards automated harmonic analysis of more complex music.

1. INTRODUCTION

Broadly, *harmony* refers to the simultaneous sounding of multiple pitches [22]. However, harmonic theory involves far more than just pitch collections. Rather, harmonic theory describes an abstract syntactic structure in Western tonal music, hierarchically removed from the literal pitches of the musical “surface” [22]. Though harmonic theory is a foundational component of basic music theory, the details of the theory are vague, and deceptively complex [5]. Harmony intertwines low-level sensory distinctions (consonance vs dissonance), short-term musical constructs (counterpoint, voice-leading), and abstract long-range musical

structures (function, form, tonality, etc.), and thus plays a central role in musical experience. Given this complexity, it is no surprise that actual harmonic analysis is highly subjective, and thwarts any attempt to systematize or automate it. This paper attempts to clarify the dimensions of harmonic analysis, identifying the import points of disagreement and ambiguity in harmonic theory. We then present a novel approach to automated harmonic analysis, which allows us to generate a variety of consistent harmonic annotations based on a various assumptions and preferences.

1.1 Theory and Terminology

To avoid confusion with the more general concept of “harmony,” we use the term *sonority* to refer to pitch-class collections. The most basic sonority is the *dyad*—pairs of pitch classes which form *consonant* or *dissonant* intervals.¹ Larger sonorities can be seen as combinatorial compositions of dyads, as each new pitch class forms an interval with every other pitch class in the sonority. Harmonic theory generalizes about various dyad combinations, reducing a huge variety of possible interval combinations to a few categories. The central harmonic categories of Western music are the set of cardinal-three sonorities in which all intervals are consonant (*triads*) and the cardinal-four sonorities which include one dissonant interval (*7th chords*). Other sonorities—the preponderance of possibilities—are unclassified and considered non-syntactic. Some genres (e.g., jazz, music theatre) employ larger sonorities (9th chords, 13th chords, etc.), which necessarily contain more dissonant intervals, as well as dissonant cardinal-three and cardinal-four sonorities (sus4, add9, etc.) [5], but even in these styles the vast majority of sonorities are unclassified.

Traditionally, dissonant harmonic intervals are only used in highly constrained melodic settings: Dissonant notes must “decorate” a neighboring consonant note, typically by moving to/from the consonance by step—a dissonance moving to a consonance by step is said to *resolve* to the consonance. Thus, a basic hierarchical distinction is introduced into music, as “decorative” dissonances are necessarily subservient to “structural” consonances. Traditional theory and pedagogy approaches larger musical textures by applying two-part concepts (parallelism, motion



© Nathaniel Condit-Schultz, Yaolong Ju, Ichiro Fujinaga. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Nathaniel Condit-Schultz, Yaolong Ju, Ichiro Fujinaga. “A Flexible Approach to Automated Harmonic Analysis: Multiple Annotations of Chorales by Bach and Prætorius”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ Here we only consider *generic* intervals, and thus *generic* dissonances. Generically, thirds, fifths, and sixths are consonant, though some specific versions of these intervals (e.g., diminished fifths, augmented thirds) are dissonant.

types, dissonance resolution) to all individual pairs. Unfortunately, larger textures introduce complexities which two-voice theory cannot address: decorative tones may appear in multiple voices at different times, at the same time, or even staggered such that one voice’s decorative dissonance cooccurs with another’s consonant resolution. As a result, the consonant harmonies which undergird musical syntax may never be explicitly sounded as sonorities. The concrete distinction between consonant and dissonant intervals gives way to a nebulous distinction between *chord-tones* which instantiate the local harmony and *non-chord* tones that decorate them [22]. This distinction, is the essential task of harmonic analysis. Traditional “roman numeral” harmonic analysis also requires some interpretation of higher-level tonal structures, including the global key and local modulations. Just as the melodic surface elaborates the underlying harmonic progressions, harmonic progressions elaborate more abstract *functional* (tonic, subdominant, dominant) progressions and prolongations, which in turn articulate the key or progressions between keys. This hierarchy, however, is not clear-cut or discrete: disentangling surface features from increasingly abstract structural progressions is difficult, and the procedure poorly defined.

1.2 Literature

Computational research into harmonic progression and function has been extensive [8, 12, 22–24, 27, 30]. Many researchers have sought to automate harmonic analysis, either using rule-based algorithms [9, 11, 21, 28, 29] or machine learning [13, 18, 19, 26]. Impressive performance has been achieved, though proper evaluation is somewhat difficult given that the “correct answer” is not clear cut. Even if interpretive leeway is allowed, algorithms inevitably struggle with even mildly idiosyncratic or exceptional passages—devising sufficiently complicated rules to cover all possibilities is impossible, and such passages are too rare to be learned by machine learning. Due to these difficulties, many researchers have relied instead on manual annotation by experts, who can make more nuanced decisions and adapt to never-before-seen situations [1, 3, 4, 8, 20]. However, though human analysts may create more accurate data, manual harmonic annotations—even by the same analyst—can be extremely inconsistent [14]. Given the subjectivity of harmonic analysis, the consistency of data annotation may actually more important than a vaguely-defined “accuracy” [6]—inconsistent answers to similar or identical musical patterns will inevitably hamper learning, whether human or machine.

To account for inconsistency and disagreement between theorists, many studies have employed multiple independent annotators [3, 4]. This approach is appropriate to the extent that analytical inconsistency is considered random noise. However, as we will explain, harmonic indeterminacy is not simply a matter of random error, but rather reflects fundamental disagreements concerning the nature, meaning, and purpose of harmonic analysis. Thus, annotation error is not (entirely) stochastic, but rather, is sys-

tematic. What’s more, though multiple independent annotations give us some sense of the scope of disagreement between analysts, they do little to clarify the root causes of these disagreements. Our view is that is preferable to: A) precisely describe the subjective features of harmonic theory; B) study how different theoretical assumptions result in different analyses; and C) evaluate how well different assumptions/models explain patterns in music. The goal of our project is to facilitate these tasks.

1.3 Analytical ambiguity

Harmonic analysis is evidently a useful tool in the description of musical structure and musical experience, yet in practice, harmonic theory is underspecified with regards to many musical passages. Indeed, many prominent theories of music (e.g. Rameau, Riemann, Schenker) differ fundamentally in their approach to harmony. It is often possible to interpret the same passage in a number of ways. Furthermore, the informative distinctions conveyed by different interpretations is unclear. This ambiguity mainly regards four questions:

1. Which harmonies are “legal” structural harmonies? Are sevenths chords true harmonies, or are they always decorative?
2. How do we interpret sonorities that are subsets, supersets, or intersections of each other? Traditional harmonic categories like $\{v, v^7, vii^o\}$ both share many pitch classes and have similar musical function—what, if any, useful information is conveyed by treating them as independent categories?
3. How do we interpret contrapuntally decorative notes which are consonant—i.e., can there be consonant non-chord tones? This issue is especially difficult when multiple voices engage in decorative motion at once, creating “passing chords.”
4. Should harmonic analysis reflect only “surface” features (like dissonance resolutions), or should higher-level structures also play a role? For instance, should, large-scale parallelism inform analyses?—i.e., analyzing two parallel passages in a similar way even if their surface details differ?

Figure 1 illustrates a number of these issues in a concrete musical example. In Figure 1, the three notes colored red form dissonances and therefore *must* be interpreted as non-chord tones. Notes colored blue are consonant, but evince melodic contours similar to the dissonant notes. Throughout this paper, we refer to each new sonority formed whenever any voice articulates a new onset as a sonority “slice”—in Figure 1, slices are numbered above the grand staff.

The consonant passing tones in slices 2 and 8 are especially illustrative. If the passing tone in slice 2 is considered a chord tone, slices 1–2 form the harmonies $I \rightarrow vi_6$, both tonic function chords. If the passing tone in slice 8 is interpreted as a chord tone, the progression $ii \rightarrow vii_6^o$ results—a transition between two different tonal functions

(subdominant and dominant). Given these functional differences, many analysts would mark slices 1–2 as a single I chord but slices 7–8 as $\text{ii} \rightarrow \text{vii}_o^6$. This is especially true since transitions from $\text{ii} \rightarrow \text{I}$ (slice 9) are considered abnormal, while transitions from $\text{vii}_o^6 \rightarrow \text{I}$ are normative. Several slices illustrate the ambiguity regarding 7th chords: Passing tones in slices 6, 18, 20, 22 and 24 might each be interpreted as sevenths, or not. For instance, the G in slice 11 can be seen as the 7th of a ii^6 harmony, or as a suspension. In Bach’s chorale music, chordal 7ths are nearly always treated like dissonances, begging the question: what is the difference between a “chord-tone 7th” and a “non-chord-tone 7th”?

2. CURRENT PROJECT

This paper describes a new approach to automated harmonic analysis, which remains agnostic regarding many of the specific interpretive complexities discussed so far. Rather, we base analyses on only a few, basic, uncontroversial constraints, allowing us to produce numerous interpretations of the same sonorities. Using this approach, we have generated a novel form of harmonic analysis dataset, including numerous harmonic annotations of chorales by Michael Pr torius (1571–1621) and Johann Sebastian Bach (1685–1750). This dataset can serve several useful functions:

1. Researchers can generate specific, *consistent* harmonic analyses, conforming to whatever analytical preferences/assumptions they prefer, for all music in the corpora. These analyses can be used like any other harmonic annotation data—i.e., to study harmonic progression and tonality in general.
2. The dataset includes a set of late-modal (Pr torius) and early-tonal (Bach) music, which are nonetheless largely similar in texture and style. This makes the dataset particularly useful for historical research [8].
3. Finally, by comparing analyses generated with different constraints, we can rigorously explore the ways in which different harmonic theories fit, or don’t fit, real music.

Chorale music is invaluable for teaching and studying harmony, as it features consistent and highly constrained melodic/contrapuntal textures, with few non-chord tones. Bach’s 371 chorales are mainstays of music theory pedagogy and have been the subject of much music information retrieval research [2, 7, 8, 16, 22–24, 27, 31]. Pr torius’ 200 chorales are music of a somewhat similar texture, but have received relatively little attention. Several sets of expert analyses of Bach’s chorales have been published, though only subsets of the chorales have annotations digitally aligned with symbolic music data. Other researchers have generated harmonic annotations—or analogous functional analyses—of the chorales computationally, and used these analyses in research, but have not published their annotations, nor describe them in detail.

3. METHODOLOGY

The approach of this project is to calculate all legal harmonic interpretations of a passage, and to only filter out specific interpretations at a later stage. Our approach is designed specifically for our dataset, and is thus rather “over fit” to chorale music, so it will not generalize well to other music. However, the basic concepts of our approach could be adapted to other tonal music.

Key to our entire endeavor is establishing “basic” constraints on harmonic interpretation. In true music theory form, we formulate these constraints as the following “rules.” There are two types of rules: harmonic rules and melodic rules. Our harmonic rules are as follows:

1. Every sonority slice belongs to one and only one harmony.
2. Every new harmony must be followed by another new harmony on the next stronger metric position—i.e., harmonic rhythm cannot be syncopated. (Some Pr torius chorales contain exceptions to this rule, as the entire rhythmic texture is syncopated.)
3. Only triads (major, minor, diminished, or augmented) and 7th chords (dominant, major, minor, half-diminished, or fully-diminished) are considered legal harmonies. However, subsets of legal harmonies may also appear in music. Complete harmonies are preferred, but cardinal-three subsets of seventh chords (Root-3rd-7th or Root-5th-7th), dyadic subsets of triads (i.e., consonant intervals), and even unisons/octaves are permitted.

Given these definitions of harmony, we can then establish which notes do, or do not, belong to the local harmony. To be a non-chord tone, a note must satisfy the following melodic rules—any note that fails any of these rules *must* be a chord tone:

1. The antecedent and consequent note of each non-chord tone must be consonant (chord tones), excepting the special case of Rule 4g (below).
2. Non-chord tones cannot sustain across metric positions that are stronger than their own metric position.
3. Non-chord tones cannot sustain through changes of harmony. A note cannot start as a non-chord tone and then become a chord tone (though the opposite is possible, in the suspension).
4. Finally, all non-chord tones must match one of these traditional contrapuntal dissonance models:
 - (a) *Passing tone*: approached and departed by step in the same direction.
 - (b) *Neighbor tone*: approached and departed by steps in opposite directions; the antecedent and consequent are the same note.
 - (c) *Suspension/Retardation*: approached by unison (or sustain); departed by step; stronger metric position than antecedent.

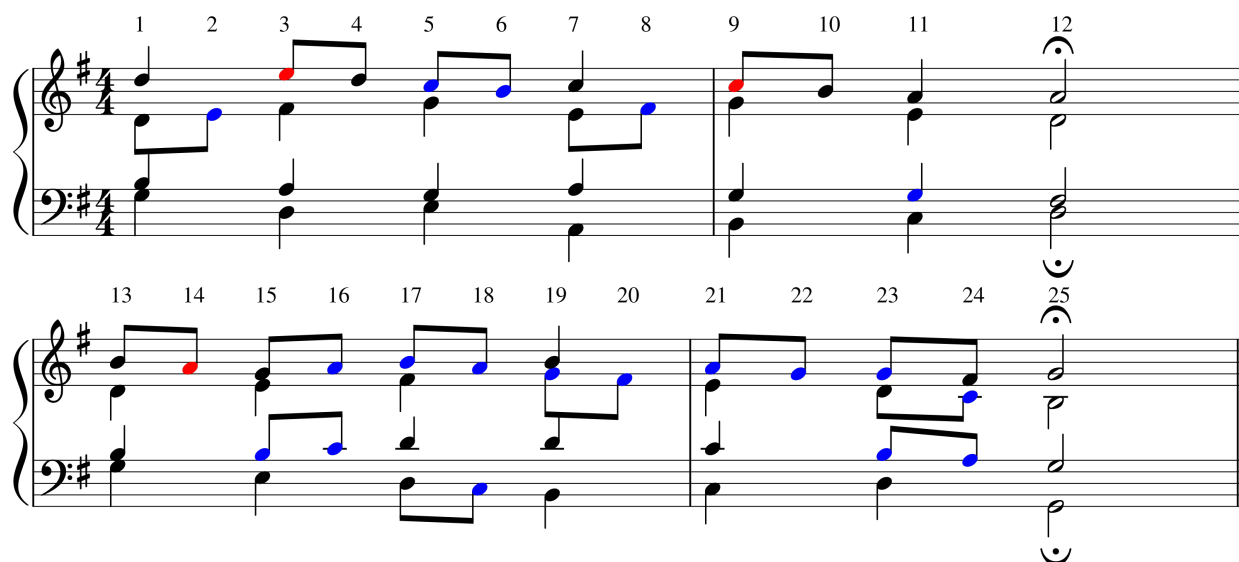


Figure 1. Illustration of “decorative” melodic idioms in a contrived example of four-part counterpoint. Slices (sonorities) are numbered above the staff. Notes colored red indicate dissonances. Notes colored blue indicate consonant notes which nonetheless articulate decorative melodic idioms, including passing tones (slices 2, 5, 8, 16, 18, 20, 21, 23, 24), neighbor tones (slices 6, 17, 18, 19), suspensions (slices 11, 23), a retardation (slice 15), and an anticipation (slice 22). Some of these interpretations are mutually exclusive, as a decorative tone cannot decorate another decorative tone. For instance, if the *C* in slice 5 is considered a passing tone, then the *B* in slice 6 must be a chord tone which resolves the passing tone.

- (d) *Appoggiatura*: approached by leap; departed by step in opposite direction; stronger metric position than antecedent.
- (e) *Escape tone*: approached by step; left by skip; weaker metric position than its antecedent.
- (f) *Pedal tone*: approached by unison (or sustain); left by unison (or sustain).
- (g) *Double passing*: two non-chord tones of the same duration, separated by step; approached and departed by step in the same direction; the first of the pair must occupy a weaker beat than its antecedent.

As in all dimensions of harmonic analysis, there is not universal agreement regarding the rules for non-chord tones. The rules set out here are an amalgam of the rules explicitly, or implicitly, described in typical music theory text books [15, 17], specialized (through some trial and error) for our chorale datasets.

3.1 Data parsing

Symbolic encodings of the Bach chorales were gathered from the KernScores repository (kern.ccarh.org), which is maintained by Stanford’s Center for Computer Assisted Research in the Humanities. The music of 370 four-part chorales, and one five-part chorale², is encoded in the humdrum `**kern` representation (www.humdrum.org) [10]. Symbolic encodings of 200 chorales by Pr torius were recently digitized by members

of McGill University SIMSSA project: Scores were initially scanned and interpreted by optical music recognition software before being corrected by a human annotator. This data was originally encoded in `musicXML` format, but was converted to `**kern` data for this project, so as to facilitate alignment with harmonic annotations. The Pr torius data includes 197 four-voice chorales and three five-voice chorales. In total, the dataset includes 571 chorales, consisting of 129,568 notes (+ 898 rests), which form 42,895 sonority slices.

`**kern` data was parsed using the Humdrum Toolkit [10], before being loaded into R [25] for additional parsing. The analysis workflow was also conducted in R. To make the analyses useful as comparisons across the two composers, (almost) the exact same parsing and analysis workflow are applied to each.

In addition to pitch and rhythm data, the Bach chorale data contains some phrasing information, in particular, fermatas. A phrase ending in a Bach chorale was identified whenever all four voices reach a fermata.³ The Pr torius chorale data contains phrasing information, encoded as rests in all voices, and both datasets contain metric information.⁴

3.2 Workflow

Our process has a two-stage workflow. The first-stage is to divide the music exhaustively into contiguous groups

³ Several chorales had notational inconsistencies, wherein fermatas were not encoded on the inner voices. These inconsistencies were fixed manually.

⁴ Though metric indications in Pr torius’ era are not exactly conceptually equivalent to modern time signatures.

² This five-part chorale was excluded from the dataset available on KernScores, but was encoded for the purposes of this study

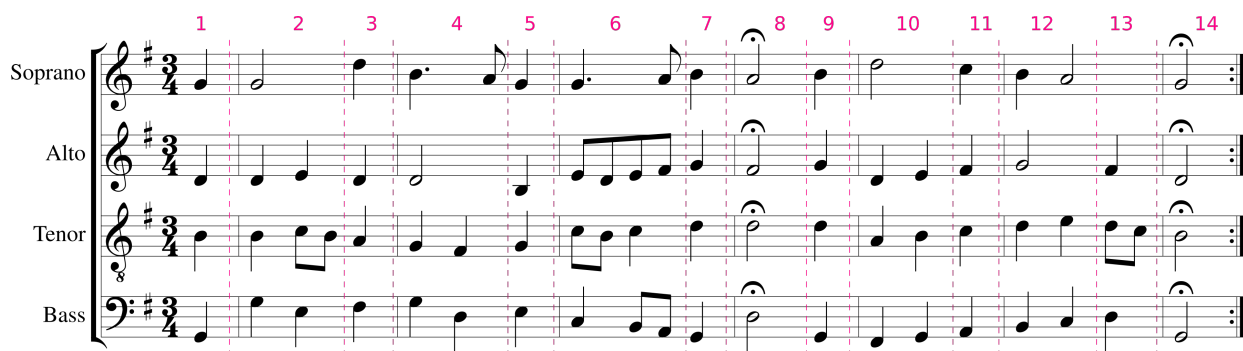


Figure 2. Illustration of contextual windows in Bach’s Chorale 1, *Aus meines Herzens Grunde*. Slices between dashed red lines are analyzed as one window.

of successive slices: “contextual windows.” The second-stage applies an analysis algorithm to each segment.

3.2.1 Stage 1

Many sonority slices can be analyzed in isolation. However, many more slices need context to be analyzed. Our approach is to parse the music into a single set of contiguous (non-overlapping) windows, identified using a simple, rule-based heuristic. A new contextual window begins anytime:

1. All voices attack on a strong beat.
2. All voices attack and one or more voices did not attack in the previous slice.
3. In an offbeat slice, more than two voices attack and one or more voices sustains into/past the next beat.
4. After a phrase boundary.

Figure 2 illustrates the contextual windows derived by this heuristic in the first seven measures of Bach’s first chorale. The aim of this heuristic is to err on the side of larger segments: unnecessarily large windows can be broken down into separate harmonies at a later stage, but windows that are too small will not provide enough context to identify all legal interpretations, and in some cases may result in windows that are not parsable.

3.2.2 Stage 2

Once analytical windows are identified, we apply the following permutational algorithm to the slices in each window.

1. Identify all ways in which the window can be divided exhaustively into sub-segments while obeying harmonic-rhythm constraints (Harmonic Rule 2).
2. For each possible segmentation, identify all pitches that can legally be non-chord tones (Melodic Rules 3–4)—we call these *potential* non-chord tones.
3. Compute every combination of potential non-chord tones, allowing that some interpretations are mutually exclusive (detailed explanation below).
4. For every legal combination of potential non-chord tones, remove these non-chord tones and group

the remaining chord tones into every possible sub-segment.

5. Discard interpretations which contain (any) illegal harmonies.
6. If any preferred harmonies are present, discard incomplete harmonies (Harmonic Rule 3).
7. If the same chord is identified in two successive slices, discard this interpretation (a different sub-segmentation is sure to have found the equivalent).
8. If a slice is identified as a dyad/unison, and the preceding or succeeding slice is a superset of that dyad/unison, the slice is subsumed into the superset.

Figure 3 illustrates the application of this algorithm to the sixth window in the chorale shown in Figure 2. The four slices in this window can be legally divided in six segmentations (Step 1), shown below the staff. Eleven of the twelve notes in the window are potential non-chord tones (labeled and enumerated in Figure 3). The algorithm tests various permutations of these potential non-chord tones (algorithm Steps 3–5) as so: First, assume potential non-chord #1 is a non-chord tone and all other notes are chord tones: under this assumption, segmentation 1 . . . forms the illegal sonority $\{A, B, C, D, E, F\# \}$; segmentation 1 . 2 . forms the illegal sonorities $\{B, C, D, E\}$ and $\{A, B, C, E, F\# \}$; segmentation 1 . . 2 forms the illegal sonority $\{B, C, D, E\}$ and the legal sonority $\{A, C, F\# \}$; etc. Repeat this procedure for every other potential non-chord tone, every pair of non-chord tones, every triplet of non-chord tones, etc., skipping combinations which are mutually exclusive—i.e., if #2 is an appoggiatura #4 must be a chord tone (Rule 1). Testing all non-chord tone permutations across all six segmentations reveals eleven non-redundant (Steps 6–8) interpretations with legal chords in all segments (Step 5).⁵ Of these eleven, we can “filter out” interpretations involving 7th chords, leaving the three triadic analyses shown in Figure 3.

⁵ Our actual algorithm incorporates a few additional optimizations to limit the number of permutations which must be tested. The most important involves pitch classes: within a given harmonic segment all instances of a single pitch class must be either non-chord tones or chord tones. For instance, it would be meaningless to treat #9 as a passing tone but treat #11 as a chord tone. Similarly, #7 (a *C*) can never actually be a passing tone, since the *C* in the bass is always a chord tone.

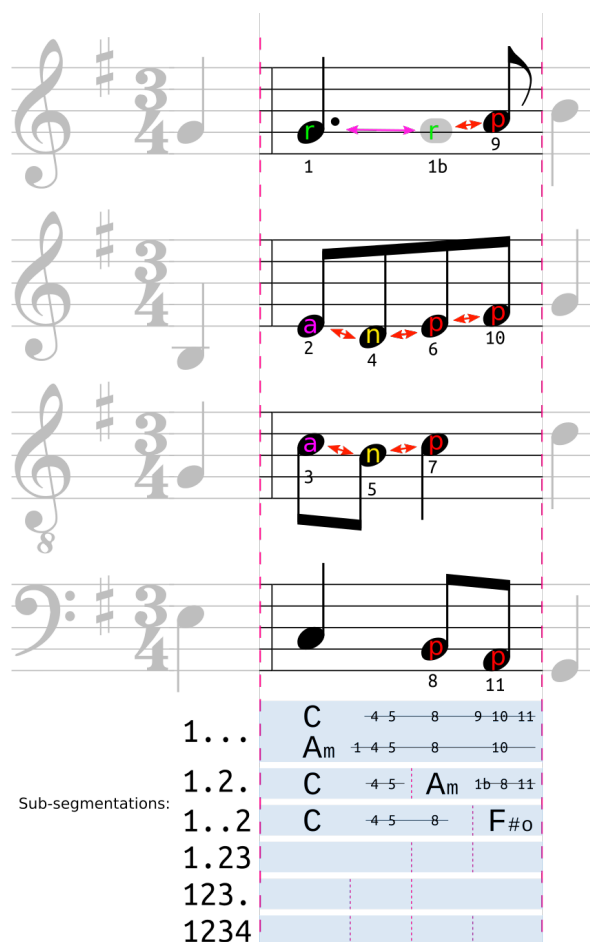


Figure 3. Illustration of the permutational analysis of a single contextual window (window 6 from Figure 2). Each note in the window is annotated as a potential non-chord tone, marked *p* for passing tone, *n* for neighbor tone, *r* for retardation, or *a* for appoggiatura—mutually exclusive potentials are annotated with arrows. The single unlabeled *C* must be a chord tone, as it does not match any contrapuntal dissonance model (Melodic Rules 4). Below the staff, the six possible rhythmic segmentations of the window are shown. The four possible purely-triadic interpretations of the window are shown; the notes which are interpreted as non-chord tones are identified (by number) beside each analysis.

3.3 Edge cases

Chorale music is valued pedagogically for its simplicity and consistency. Nonetheless, a handful of chorales contain unusual features which complicate the batch analysis of the corpora. Notable examples in the Bach chorales include: an unusual call and response between the soprano and the rest of the voices in Chorale 43; dissonant notes which resolve across phrase boundaries (i.e., through a fermata) in Chorales 127, 202, and 234; and suspensions which resolve indirectly in Chorales 5 and 199. A number of Prætorius chorales also contain subsections in which a subset of voices sing while the others rest, confounding our windowing heuristic. Solutions to these special cases, and a handful others, were hard-coded into the workflow.

4. API

The data is hosted at github.com/DDMAL/Flexible-harmonic-chorale-annotations.

The harmonic permutation data is stored in a *rData* file. Users may filter out specific harmonic analyses using an online GUI, and download them as a zipped collection of text files encoded in the Humdrum Syntax. Each file contains the ***kern* representation of a chorale aligned with one or more harmonic analyses in a ***harm* representation. Interpretations can be filtered by the following criteria:

- Type of harmonies.
- Number of harmonies (per beat/per window).
- Types of non-chord tones.
- Number of non-chord tones (per slice/per window).

For example, one could extract analyses which forbid augmented triads, appoggiaturas, and ♯ harmonic rhythms. Users may also download the raw data and associated R scripts for local use or customization.

5. CONCLUSION

The empirical and computational study of harmony is essential to furthering our understanding of musical structure and perception. However, this research must remain cognisant of the subtle complexities and controversies of harmonic theory if it is to be fruitful. We have presented a novel approach to automated harmonic analysis which is not limited to one specific set of theoretical assumptions, allowing for just such subtleties to be explored systematically. We have also described a new dataset generated via this method. We hope that this dataset will facilitate research into tonality and harmonic progression, especially changes in harmonic practice between the early 1600s and the mid 1700s. However, our grander purpose is to facilitate critical, data-driven, interrogation of harmonic theory in general.

6. REFERENCES

- [1] John Ashley Burgoyne, John Wild, and Ichiro Fujinaga. An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis. In A Klapuri and C. Leider, editors, *Proceedings of the 12th International Society for Music Information Retrieval (ISMIR) Conference*, pages 633–638, Miami, FL, 2011.
- [2] Trevor de Clercq. A Model for Scale-Degree Reinterpretation: Melodic Structure, Modulation, and Cadence Choice in the Chorale Harmonizations of J. S. Bach. *Empirical Musicology Review*, 10:188–206, 05 2015.
- [3] Trevor de Clercq and David Temperley. A Corpus Analysis of Rock Harmony. *Popular Music*, 30(01):47–70, January 2011.
- [4] Johanna Devaney, Claire Arthur, Nathaniel Condit-Schultz, and Kirsten Nisula. Theme and Variation Encodings with Roman Numerals: A New Data Set for Symbolic Music Analysis. In Meinard Müller and Frans Wiering, editors, *Proceedings of the 16th International Society for Music Information Retrieval (ISMIR) Conference*, pages 728–734, Malaga, Spain, 2015.
- [5] Christopher Doll. Definitions of ‘Chord’ in the Teaching of Tonal Harmony. *Dutch Journal of Music Theory*, 18(2):91–106, 2013.
- [6] Mark Granroth-Wilding and Mark Steedman. A Robust Parser-Interpreter for Jazz Chord Sequences. *Journal of New Music Research*, 43(4):355–374, 2014.
- [7] Gaëtan Hadjeres and François Pachet. DeepBach: a Steerable Model for Bach Chorales Generation. *CoRR*, 2016.
- [8] Thomas Hedges and Martin Rohrmeier. Exploring Rameau and Beyond: A Corpus Study of Root Progression Theories. In *Mathematics and Computation in Music: Third International Conference, MCM 2011, Paris, France, June 15-17, 2011. Proceedings*, pages 334–337, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [9] Tim Hoffman and William P Birmingham. A Constraint Satisfaction Approach to Tonal Harmonic Analysis. *Electrical Engineering and Computer Science Department. Technical Report CSE-TR-397-99. University of Michigan*, 2000.
- [10] David Huron. *Music Research Using Humdrum: A User’s Guide*. Stanford, California: Center for Computer Assisted Research in the Humanities, 1999.
- [11] Plácido R. Illescas, David Rizo, and José M. Iñesta. Harmonic, Melodic, and Functional Automatic Analysis. In *Proceedings of the International Computer Music Conference*, pages 165–168, 2007.
- [12] Nori Jacoby, Naftali Tishby, and Dmitri Tymoczko. An Information Theoretic Approach to Chord Categorization and Functional Harmony. *Journal of New Music Research*, 44(3):219–244, 2015.
- [13] Yaolong Ju, Nathaniel Condit-Schultz, and Ichiro Fujinaga. Non-chord Tone Identification Using Deep Neural Networks. In *Proceedings of the Fourth International Workshop on Digital Libraries for Musicology*, pages 13–16. ACM, 2017.
- [14] Hendrik Vincent Kooops, W. Bas de Hass, John Ashley Burgoyne, and Jeroen Bransen. Harmonic Subjectivity in Popular Music. Technical report, Department of Information and Computing Sciences, Utrecht University, 2017.
- [15] Stefan Kostka and Dorothy Payne. *Tonal Harmony: With an Introduction to Twentieth-Century Music*. McGraw Hill, 2004.
- [16] Pedro Kröger, Alexandre Passos, Marcos Sampaio, and Givaldo De Cidra. Rameau: a System for Automatic Harmonic Analysis. In *Proceedings of International Computer Music Conference*, pages 273–281, 2008.
- [17] Steven G. Laitz. *The Complete Musician: an Integrated Approach to Tonal Theory, Analysis, and Listening*. Oxford University Press, 3rd edition, 2012.
- [18] Kristen Masada and Razvan Bunescu. Chord Recognition in Symbolic Music Using Semi-Markov Conditional Random Fields. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pages 272–278, 2017.
- [19] Lesley Mearns. *The Computational Analysis of Harmony in Western Art Music*. PhD thesis, Queen Mary University of London, 2013.
- [20] Néstor Nápoles López. Automatic Harmonic Analysis of Classical String Quartets from Symbolic Score. Master’s thesis, Universitat Pompeu Fabra, 2017.
- [21] Brian Pardo and William P. Birmingham. The Chordal Analysis of Tonal Music. In *The University of Michigan, Department of Electrical Engineering and Computer Science Technical Report CSE-TR-439-01*, 2001.
- [22] Ian Quinn. Are Pitch-Class Profiles Really Key for Key? *Gesellschaft für Musiktheorie*, 7(2):151–163, 2010.
- [23] Ian Quinn and Panayotis Mavromatis. Voice-Leading Prototypes and Harmonic Function in Two Chorale Corpora. In Carlos Agon, Moreno Andreatta, Gérard Assayag, Emmanuel Amiot, Jean Bresson, and John Mandereau, editors, *Mathematics and Computation in Music*, pages 230–240, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

- [24] Ian Quinn and Christopher Wm. White. Expanding Notions of Harmonic Function Through a Corpus Analysis of the Bach Chorales. In *Annual Conference of the Society for Music Theory*, Charlotte, NC, 2013.
- [25] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.
- [26] Christopher Raphael and Joshua Stoddard. Functional Harmonic Analysis Using Probabilistic Models. *Computer Music Journal*, 28(3):45–52, 2004.
- [27] Martin Rohrmeier and Ian Cross. Statistical Properties of Tonal Harmony in Bach’s Chorales. In *Proceedings of the 10th International Conference on Music Perception and Cognition*, 2008.
- [28] Craig Stuart Sapp. Computational Chord-Root Identification in Symbolic Musical Data: Rationale, Methods, and Applications. *Computing in Musicology*, 15:99–119, 2007.
- [29] David Temperley and Daniel Sleator. Modeling Meter and Harmony: A Preference-rule Approach. *Computer Music Journal*, 23(1):10–27, 1999.
- [30] Christopher Wm. White. Changing Styles, Changing Corpora, Changing Models. *Music Perception*, 31(3):244–253, 2014.
- [31] Matthew Woolhouse. Probability and Style in the Chorales of J. S. Bach. *Empirical Musicology Review*, 10(3):207, 2015.

EVALUATING A COLLECTION OF SOUND-TRACING DATA OF MELODIC PHRASES

Tejaswinee Kelkar

RITMO, Dept. of Musicology

University of Oslo

tejaswinee.kelkar@imv.uio.no

Udit Roy

Independent Researcher

udit.roy@alumni.iiit.ac.in

Alexander Refsum Jensenius

RITMO, Dept. of Musicology

University of Oslo

a.r.jensenius@imv.uio.no

ABSTRACT

Melodic contour, the ‘shape’ of a melody, is a common way to visualize and remember a musical piece. The purpose of this paper is to explore the building blocks of a future ‘gesture-based’ melody retrieval system. We present a dataset containing 16 melodic phrases from four musical styles and with a large range of contour variability. This is accompanied by full-body motion capture data of 26 participants performing sound-tracing to the melodies. The dataset is analyzed using canonical correlation analysis (CCA), and its neural network variant (Deep CCA), to understand how melodic contours and sound tracings relate to each other. The analyses reveal non-linear relationships between sound and motion. The link between pitch and verticality does not appear strong enough for complex melodies. We also find that descending melodic contours have the least correlation with tracings.

1. INTRODUCTION

Can hand movement be used to retrieve melodies? In this paper we use data from a ‘sound-tracing’ experiment (Figure 1) containing motion capture data to describe music–motion cross-relationships, with the aim of developing a retrieval system. Details about the experiment and how motion metaphors come to play a role in the representations are presented in [19]. While our earlier analysis was focused on the use of the body and imagining metaphors for tracings [17, 18], in this paper, we will focus on musical characteristics and study music–motion correlations. The tracings present a unique opportunity for cross-modal retrieval, because a direct correspondence between tracing and melodic contour presents an inherent ‘ground-truth.’

Recent research in neuroscience and psychology has shown that *action* plays an important role in perception. In phonology and linguistics, the co-articulation of action and sound is also well understood. Theories from embodied music cognition [22] have been critical to this exploration of multimodal correspondences.

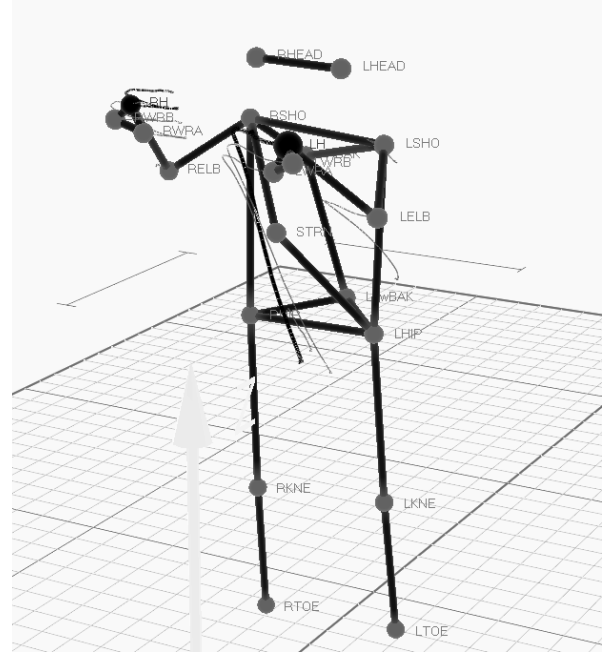


Figure 1. An example of post-processed motion capture data from a sound-tracing study of melodic phrases.

Contour perception is a coarse-level musical ability that we acquire early during childhood [30, 33, 34]. Research suggests that our memory for contour is enhanced when melodies are tonal, and when tonal accent points of melodies co-occur with strong beats [16], making melodic memory a salient feature in musical perception. More generally, it is easier for people to remember the general shape of melody rather than precise intervals [14], especially if they are not musical experts. Coarse representations of melodic contour, such as with drawing or moving hands in the air may be intuitive to capturing musical moments of short time scales [9, 25].

1.1 Research Questions

The inspiration for our work mainly comes from several projects on melodic content retrieval using intuitive and multi-modal representations of musical data. The oldest example of this is the 1975 project titled ‘Directory of Tunes and Musical Themes,’ where the author uses a simplified contour notation method, involving letters for de-



© Tejaswinee Kelkar, Udit Roy, Alexander Refsum Jensenius. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Tejaswinee Kelkar, Udit Roy, Alexander Refsum Jensenius. “Evaluating a collection of Sound-Tracing Data of Melodic Phrases”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

noting contour directions, to create a dictionary of musical themes where one may look up a tune they remember [29]. This model is adopted for melodic contour retrieval in Musipedia.com [15]. Another system is proposed in the recent project SoundTracer, in which a user's motion of their mobile phone is used to retrieve tunes from a music archive [21]. A critical difference between these approaches is how they handle mappings between contour information and musical information, especially differences between time-scales and time-representations. Most of these methods do not have ground-truth models of contours, and instead use one of several ways of mappings, each with its own assumptions.

Godøy et al. has argued for using motion-based, graphical, verbal, and other representations of motion data in music retrieval systems [10]. Liem et al. make a case for using multimodal user-centered strategies as a way to navigate the discrepancy between audio similarity and music similarity [23], with the former referring to more mathematical features, and the latter to more perceptual features. We proceed with this as the point of departure for describing our dataset and its characteristics, to approach the goal of making a system for classifying sound-tracings of melodic phrases with the following specific questions:

1. Are the mappings between melodic contour and motion linearly related?
2. Can we confirm previous findings regarding correlation between pitch and the vertical dimension?
3. What categories of melodic contour are most correlated for sound-tracing queries?

2. RELATED WORK

Understanding the close relationship between music and motion is vital to understanding subjective experiences of performers and listeners, [7, 11, 12]. Many empirical experiments aimed at investigating music-motion correspondences deal with stimulus data that is made to explicitly observe certain mappings, for example pitched and non-pitched sound, vertical dimension and pitch, or player expertise [5, 20, 27]. This means that the music examples themselves are sorted into types of sound (or types of motion). We are more interested in observing how a variety of these mapping relationships change in the content of melodic phrases. For this we use multiple labeling strategies as explained in section 3.4. Another contribution of this work is the use of musical styles from various parts of the world, including those that contain microtonal inflections.

2.1 Multi-modal retrieval

Multi-modal retrieval is the paradigm of information retrieval used to handle different types of data together. The objective is to learn a set of mapping functions that project the different modalities into a common metric space, to be able to retrieve relevant information in one modality

through a query in another. We see that this paradigm is used often in the retrieval of image from text and text from image. Canonical Correlation Analysis (CCA) is a common tool for investigating linear relationships of two sets of variables. In the review paper by Wang et al. for cross modal retrieval [35], several implementations and models are analyzed. CCA is also previously used to show music and brain imaging cross relationships [3].

A previous paper analyzing tracings to pitched and non pitched sounds also used CCA to understand music-motion relationships [25], where the authors describe inherent non-linearity in the mappings, despite finding intrinsic sound-action relationships. This work was extended in [26], in which CCA was used to interpret how different features correlate with each other. Pitch and vertical motion have linear relationships in this analysis, although it is important to note that the sound samples used for this study were short and synthetic.

The biggest reservations in analyzing music-motion data through CCA is that non-linearity cannot be represented, and the dependence of the method on time synchronization is high. The temporal evolution of motion and sound remains linear over time [6]. To get around this, kernel-based methods can be used to introduce non-linearity. Ohkushi et al., present a paper that uses Kernel-based CCA methods to analyze motion and music features together using video sequences from classical ballet, and optical flow based clustering. Bozkurt et al. present a CCA based system to analyze and generate speech and arm motion for prosody-driven synthesis of the 'beat-gesture' [4], which is used for emphasizing prosodically salient points in speech. We explore our dataset through CCA due to the previous successes of using this family of methods. We will analyze the same data using Deep CCA, a neural-network approximation of CCA, to understand better the non-linear mappings.

2.2 Canonical Correlation Analysis

CCA is a statistical method to find a linear combination of two variables $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_m)$ with n and m independent variables as vectors a and b such that their correlation $\rho = \text{corr}(aX, bY)$ of the transformed variables is maximized. Linear vectors a' and b' can be found such that $a', b' = \arg\max_{a, b} \text{corr}(a^T X, b^T Y)$. We can then find the second set of coefficients which maximize the correlation of the variables $X' = aX$ and $Y' = bY$ with the additional constraint to keep (X, X') and (Y, Y') uncorrelated. This process can be repeated till $d = \min(m, n)$ dimensions.

The CCA can be extended to include non-linearity by using a neural network to transform the X and Y variables as in the case of Deep CCA [2]. Given the network parameters θ_1 and θ_2 , the objective is to maximize the correlation $\text{corr}(f(X, \theta_1), f(Y, \theta_2))$. The network is trained by following the gradient of the correlation objective as estimated from the training data.

3. EXPERIMENT DESCRIPTION

3.1 Procedure

The participants were instructed to move their hands as if their movement was creating the melody. The use of the term ‘creating,’ instead of ‘representing,’ is purposeful, as shown in earlier studies [26,27], to be able to access sound-production as the tracing intent. The experiment duration was about 10 minutes. All melodies were played at a comfortable listening level through a Genelec 8020 speaker, placed 3m in front of the subjects. Each session consisted of an introduction, two example sequences, 32 trials and a conclusion. Each melody was played twice with a 2s pause in between. During the first presentation, the participants were asked to listen to the stimuli, while during the second presentation, they were asked to trace the melody. All the instructions and required guidelines were recorded and played back through the speaker. Their motions are tracked using 8 infra-red cameras from Qualisys (7 Oqus 300 and 1 Oqus 410). We then post-process the data in Qualisys Track Manager (QTM) first by identifying and labeling each marker for each participant. Thereafter, we create a dataset containing Left and Right hand coordinates for all participants.

Six participants in the study had to be excluded due to too many marker dropouts, giving us a final dataset containing 26 participants tracing 32 melodies: 794 tracings for 16 melodic categories.

3.2 Subjects

The 32 subjects (17 females, 15 males) had a mean age of 31 years ($SD = 9$ years). They were mainly university students and employees, both with and without musical training. Their musical experience was quantized using the OMSI (Ollen Musical Sophistication Index) questionnaire [28], and they were also asked about the familiarity with the musical genres, and their experience with dancing. The mean of the OMSI score was 694 ($SD = 292$), indicating that the general musical proficiency in this dataset was on the higher side. The average familiarity with Western classical music was 4.03 out of a possible 5 points, 3.25 for jazz music, 1.87 with Sami joik, and 1.71 with Hindustani music. None of the participants reported having heard any of the melodies played to them. All participants provided their written consent for inclusion before they participated in the study, and they were free to withdraw during the experiment. The study design was approved by the National ethics board (NSD).

3.3 Stimuli

In this study, we decided to use melodic phrases from vocal genres that have a tradition of singing without words. Vocal phrases without words were chosen so as to not introduce lexical meaning as a confounding variable. Leaving out instruments also avoids the problem of subjects having to choose between different musical layers in their sound-tracing. The final stimulus set consists of four different

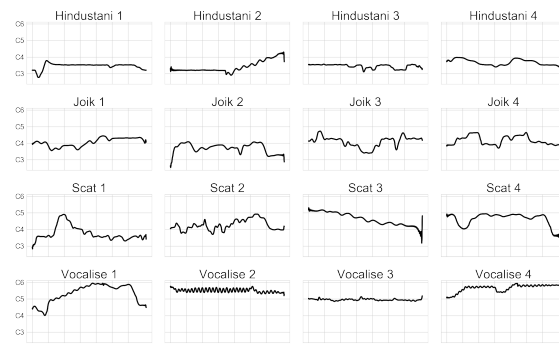


Figure 2. Pitch plots of all the 16 melodic phrases used as experiment stimuli, from each genre. The x axis represents time in seconds, and the y axis represents notes. The extracted pitches were re-synthesized to create a total of 32 melodic phrases used in the experiment.

musical genres and four stimuli for each genre. The musical genres selected are: (1) Hindustani music, (2) Sami joik, (3) jazz scat singing, (4) Western classical vocalise. The melodic fragments are phrases taken from real recordings, to retain melodies within their original musical context. As can be seen in the pitch plots in Figure 2, the melodies are of varying durations with an average of 4.5 s ($SD = 1.5$ s). The Hindustani and joik phrases are sung by male vocalists, whereas the scat and vocalise phrases are sung by female vocalists. This is represented in the pitch range of each phrase as seen in Figure 2.

Seeger	xx	xy	xyy	xyx
Schaeffer	Impulsive	Iterative	Sustained	
Varna	Ascending	Descending	Stationary	Varying
Hood	Arch	Bow	Tooth	Diagonal
Adams	Repetition	Recurrence		

Figure 3. Contour Typologies discussed previously in melodic contour analysis. This figure is representative, made by the authors.

Melodic contours are overwhelmingly written about in terms of pitch, and so we decided to create a ‘clean’ pitch-only representation of each melody. This was done by running the sound files through an autocorrelation algorithm to create phrases that accurately resemble the pitch content, but without the vocal, timbral and vowel content of the melodic stimulus. These 16 re-synthesized sounds were added to the stimulus set, thus obtaining a total of 32 sound stimuli.

	ID	Description
1	All	16 Melodies
2	IJSV	4 Genres
3	ADSC	Ascending, Descending, Steady or Combined
4	OrigVSyn	Original vs Synthesized
5	VibNonVib	Vibrato vs No Vibrato
6	MotifNonMotif	Motif Repetition Present vs Not

Table 1. Multiple labellings for melodic categories: we represent the 16 melodies using 5 different label sets. This helps us analyze which features are best related to which contour classes, genres, or melodic properties.

3.4 Contour Typology Descriptions

We base the selection of melodic excerpts on the descriptions of melodic contour classes as seen in Figure 3. The reference typologies are based on the work of Seeger [32], Hood [13], Schaeffer [8], Adams [1], and the Hindustani classical Varna system. Through these typologies, we hope to cover commonly understood contour shapes and make sure that the dataset contains as many of them as possible.

3.4.1 Multiple labeling

To represent the different contour types and categories that these melodies represent, we create multiple labels that explain the differences. This enables us to understand how the sound tracings actually map to the different possible categories, and makes it easier to see patterns from the data. We describe these labels as seen in Table 3.4.1. Multiple labels allow us to see what categories does the data describe, and which features or combination of features can help retrieve which labels. Some of these labels are categories, while some are one-versus-rest. Category labels include individual melodies, genres, and contour categories, while one-versus-rest correlations are computed for finding whether vibrato, motivic repetitions exist in the melody, and whether the melodic sample is re-synthesized or original.

4. DATASET CREATION

4.1 Preprocessing of Motion Data

We segment each phrase that is traced by the participants, label participant and melody numbers, and extract the data for left and right hand markers for this analysis, since the instructions asked people to trace using their hands. To analyze this data, we are more interested in contour features and shape information than time-scales. We therefore time-normalize our datasets so that every melodic sample and every motion tracing is the same length. This makes it easier to find correlations between music and motion data using different features.

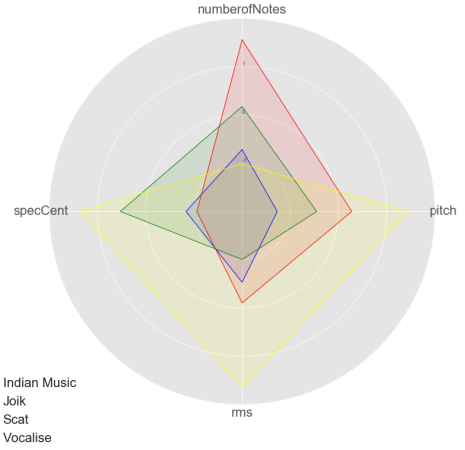


Figure 4. Feature distribution of melodies for each genre. We make sure that a wide range of variability in the features, as described in Table 2 is present in the dataset.

	Feature	Calculated by
1	Pitch	Autocorrelation function using PRAAT
2	Loudness	RMS value of the sound using Librosa
3	Brightness	Spectral Centroid using Librosa
4	Number of Notes	Number of notes per melody

Table 2. Melody features extracted for analysis, and details of how they are extracted.

5. ANALYSIS

5.1 Music

Since we are mainly interested in melodic correlations, the most important feature describing melodies is to extract pitch. For this, we use autocorrelation algorithm available in the PRAAT phonetic program. We use Librosa v0.5.1 [24] to compute the RMS energy (loudness), and the brightness using Spectral Centroid. We transcribe the melodies to get the number of notes per melody. The distribution of these features can be seen for each genre in the stimulus set in Figure 4. We have tried to be true to the musical styles used in this study, most of which do not have written notation as an inherent part of their pedagogy.

5.2 Motion

For tracings, we calculate 9 features that describe various characteristics of motion. We record only X and Z axes, as maximum motion is found along these directions. The derivatives of motion (velocity, acceleration, jerk) and quantity of motion (QoM) which is a cumulative velocity quantity are calculated. Distance between hands, cumulative distance, and symmetry features are calculated as indicators of contour-supporting features, as found in previous studies.

	Feature	Description
1	X-coordinate (X)	Axis corresponding to the direction straight ahead of the participant
2	Z-coordinate (Z)	Axis corresponding to the upwards direction
3	Velocity (V)	First derivative of vertical position
4	Acceleration (A)	Second derivative of vertical position
5	Quantity of Motion	Sum of absolute velocities for all markers
6	Distance between Hands	Sample-wise Euclidean distance between hand markers
7	Jerk	Third derivative of vertical position
8	Cumulative Distance Traveled	Euclidean distance traveled per sample per hand
9	Symmetry	Difference between the left and right hand in terms of vertical position and horizontal velocity

Table 3. Motion features used for analysis. 1-5 are for the dominant hand, while 6-9 are features for both hands.

5.3 Joint Analysis

In this section we present our analysis on our dataset with these two feature sets. We analyze the tracings for each melody as well as utilize the multiple label sets to discover interesting patterns in our dataset which are relevant for a retrieval application.

5.3.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is a method to align sequences of different lengths using substitution, addition and subtraction costs. It is a non-metric method giving us the distance between two sequences after alignment.

In recent research, vertical motion has been shown to correlate with pitch in the past for simple sounds. Some form of non-alignment is also observed between the motion and pitch signals. We perform the same analysis on our data: compute the correlation between pitch and motion in the Z axis before and after alignment with DTW for the 16 melodies and plot their mean and variance in Figure 5.

5.3.2 Longest Run-lengths

While observing the dataset, we find that longest ascending and descending sequences in the melodies are most often reliably represented in the motions, although variances in stationary notes, and ornaments is likely to be much higher. To exploit this feature in tracings, we use “Longest Run-lengths” as a measure. We find multiple subsequences following a pattern which can possess discriminative qualities. For our analysis, we use the ascending and descending patterns, thus finding the subsequences

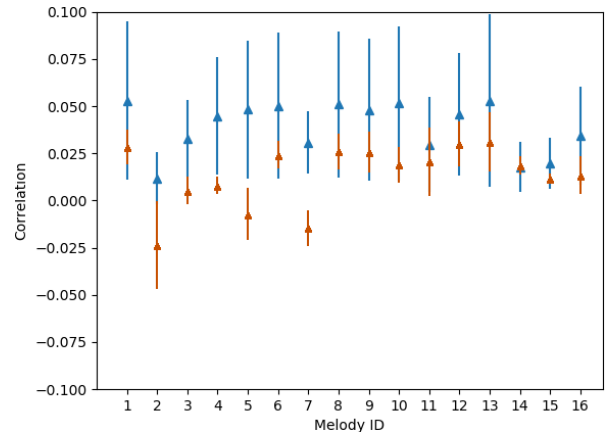


Figure 5. Correlations of pitch with raw data (red) vs after DTW-alignment (blue). Although a DTW alignment improves the correlation, we observe that correlation is still low suggesting that vertical motion and pitch height are not that strongly associated.

from the feature sequence which are purely ascending or descending. We then rank the subsequences and build a feature vector from the lengths of the top N results. This step is particularly advantageous when comparing features from motion and music sequences as it captures the overall presence of the pattern in the sequence remaining invariant to the mis-alignment or lag between the sequences from different modalities. As an example, if we select the Z-axis motion of the dominant hand and the melody pitch as our sequences and retrieve top 3 ascending subsequence lengths. To make the features robust, we do a low pass filtering of the sequence as a preprocessing step.

We analyze our dataset by computing the features for few combinations of motion and music features for ascending and descending patterns. Thereafter, we perform CCA and show the resulting correlation of first transformed dimension in Table 4. We utilize the various label categories generated for the melodies, and show the impact of the features on the labels from each category in Tables 4 and 5. We select the top four run lengths as our feature for each music-motion feature sequence. For Deep CCA analysis, we use a two layered network (same for both motion and music features) with 10 and 4 neurons. A final round of linear CCA is also performed on the network output.

6. RESULTS AND DISCUSSION

Figure 5 shows correlations with raw data and after DTW alignment between the vertical motion and pitch for each melody. Overall, the correlation improves after DTW alignment, suggesting phase lags and phase differences between the timing of melodic peaks and onsets, and those of motion. We see no significant differences between genres, although the improvement in correlations for the vocalized examples is the least pre and post DTW. This could be because of the continuous vibrato in these examples, causing people to use more ‘shaky’ representations which are most

Motion	Music	All		ADSC		IJSV	
Ascend Pattern		CCA	Deep CCA	CCA	Deep CCA	CCA	Deep CCA
Z	Pitch	0.19	0.23	0.25 0.16 0.09 0.05	0.24 0.17 0.12 0.13	0.16 -0.13 0.01 0.37	0.19 0.21 0.08 0.36
Z + V	Pitch	0.21	0.27	0.26 0.09 0.15 0.10	0.30 0.03 0.05 0.17	0.22 -0.13 -0.01 0.35	0.24 0.25 0.15 0.34
All	All	0.33	0.44	0.31 0.14 0.19 0.29	0.44 0.29 0.01 0.36	0.30 0.28 0.23 0.42	0.38 0.43 0.27 0.52
Descend Pattern							
Z	Pitch	0.18	0.21	0.16 -0.11 0.15 0.20	0.17 0.19 0.09 0.19	0.22 0.21 -0.04 0.23	0.22 0.18 0.08 0.28
Z + V	Pitch	0.21	0.31	0.23 0.03 0.14 0.22	0.28 0.28 0.30 0.32	0.26 0.23 0.10 0.24	0.42 0.18 0.34 0.17
All	All	0.35	0.44	0.39 0.12 0.20 0.25	0.38 0.02 0.37 0.37	0.35 0.25 0.12 0.36	0.40 0.22 0.14 0.52

Table 4. Correlations for all samples in the dataset and the two major categorizations of music labels, using ascend and descend patterns as explained in Section 5.3.2, and features from Tables 3 and 2

Motion	Music	MotifNonMotif		OrgSyn		VibNonVib	
Ascend Pattern		CCA	Deep CCA	CCA	Deep CCA	CCA	Deep CCA
Z	Pitch	0.05 0.23	0.13 0.26	0.19 0.19	0.22 0.25	0.33 0.07	0.33 0.13
Z + V	Pitch	0.10 0.24	0.17 0.31	0.19 0.22	0.24 0.31	0.33 0.09	0.32 0.20
All	All	0.29 0.34	0.36 0.47	0.30 0.35	0.42 0.45	0.38 0.29	0.49 0.40
Descend Pattern							
Z	Pitch	0.20 0.17	0.19 0.21	0.20 0.16	0.23 0.18	0.20 0.17	0.24 0.18
Z + V	Pitch	0.22 0.22	0.32 0.29	0.24 0.20	0.35 0.26	0.22 0.22	0.14 0.34
All	All	0.25 0.40	0.37 0.45	0.38 0.33	0.45 0.44	0.33 0.35	0.54 0.35

Table 5. Correlations for two-class categories, using ascend and descend patterns as explained in Section 5.3.2 with features from Tables 3 and 2

consistent between participants. The linear mappings of pitch and vertical motion are limited, making the dataset challenging. This also means that the associations between pitch and vertical motion, as described in previous studies, are not that clear for this stimulus set, especially as we use musical samples that are not controlled for being isochronous, nor equal tempered.

Thereafter, we conduct CCA and Deep CCA analysis as seen in Tables 4, 5. Overall, Deep CCA performs better than its linear counterpart. We find better correlation with all features from Table 3, as opposed to just using vertical motion and velocity. With ascending and descending longest run-lengths, we are able to achieve similar results for correlating all melodies with their respective tracings. However, descending contour classification does not have similar success. There is more general agreement on contour with some melodies than others, with purely descending melodies having particularly low correlation. There is some evidence that descending intervals are harder to identify than ascending intervals [31], and this could explain a low level of agreement in this study amongst people for descending melodies. Studying differences between ascending and descending contours requires further study.

While using genre-labels (IJSV) for correlation, we find that scat samples show the least correlation, and the least improvement. Speculatively, this could be related to the high number of spoken syllables in this style, even though the syllables are not words. Deep CCA also gives an overall correlation of 0.54 for recognizing melodies containing vibrato from the dataset. This is an indication that sonic

textures are well represented in such a dataset.

With all melody and all motion features, we find an overall correlation of 0.44 with Deep CCA, for both the longest ascend and longest descend features. This supports the view that non-linearity is inherent to tracings.

7. CONCLUSIONS AND FUTURE WORK

Interest in cross-modal systems is growing in the context of multi-modal analysis. Previous studies in this area include shorter time scales or synthetically generated isochronous music samples. The strength of this particular study is in using musical excerpts as are performed, and that the performed tracings are not iconic or symbolic, but spontaneous. This makes the dataset a step closer to understanding contour perception in melodies. We hope that the dataset will prove useful for pattern mining, as it presents novel multimodal possibilities for the community and could be used for user-centric retrieval interfaces.

In the future, we wish to create a system to synthesize melody–motion pairs based on training a network to this dataset, and conducting a user evaluation study, where users evaluate system generated music–motion pairs in a forced-choice paradigm.

8. ACKNOWLEDGMENTS

Partially supported by the Research Council of Norway through its Centres of Excellence scheme (262762 & 250698), and the Nordic Sound and Music Computing Network funded by the Nordic Research Council.

9. REFERENCES

- [1] Charles R Adams. Melodic contour typology. *Ethnomusicology*, pages 179–215, 1976.
- [2] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International Conference on Machine Learning*, pages 1247–1255, 2013.
- [3] Nick Gang Blair Kaneshiro Jonathan Berger and Jacek P Dmochowski. Decoding neurally relevant musical features using canonical correlation analysis. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, Souzhou, China*, 2017.
- [4] Elif Bozkurt, Yücel Yemez, and Engin Erzin. Multi-modal analysis of speech and arm motion for prosody-driven synthesis of beat gestures. *Speech Communication*, 85:29–42, 2016.
- [5] Baptiste Caramiaux, Frédéric Bevilacqua, and Norbert Schnell. Towards a gesture-sound cross-modal analysis. In *International Gesture Workshop*, pages 158–170. Springer, 2009.
- [6] Baptiste Caramiaux and Atau Tanaka. Machine learning of musical gestures. In *NIME*, pages 513–518, 2013.
- [7] Martin Clayton and Laura Leante. Embodiment in music performance. 2013.
- [8] Rolf Inge Godøy. Images of sonic objects. *Organised Sound*, 15(1):54–62, 2010.
- [9] Rolf Inge Godøy, Egil Haga, and Alexander Refsum Jensenius. Exploring music-related gestures by sound-tracing: A preliminary study. 2006.
- [10] Rolf Inge Godøy and Alexander Refsum Jensenius. Body movement in music information retrieval. In *10th International Society for Music Information Retrieval Conference*, 2009.
- [11] Anthony Gritten and Elaine King. *Music and gesture*. Ashgate Publishing, Ltd., 2006.
- [12] Anthony Gritten and Elaine King. *New perspectives on music and gesture*. Ashgate Publishing, Ltd., 2011.
- [13] Mantle Hood. *The ethnomusicologist*, volume 140. Kent State Univ Pr, 1982.
- [14] David Huron. The melodic arch in western folksongs. *Computing in Musicology*, 10:3–23, 1996.
- [15] K Irwin. Musipedia: The open music encyclopedia. *Reference Reviews*, 22(4):45–46, 2008.
- [16] Mari Riess Jones and Peter Q Pfordresher. Tracking musical patterns using joint accent structure. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 51(4):271, 1997.
- [17] Tejaswinee Kelkar and Alexander Refsum Jensenius. Exploring melody and motion features in sound-tracings. In *Proceedings of the SMC Conferences*, pages 98–103. Aalto University, 2017.
- [18] Tejaswinee Kelkar and Alexander Refsum Jensenius. Representation strategies in two-handed melodic sound-tracing. In *Proceedings of the 4th International Conference on Movement Computing*, page 11. ACM, 2017.
- [19] Tejaswinee Kelkar and Alexander Refsum Jensenius. Analyzing free-hand sound-tracings of melodic phrases. *Applied Sciences*, 8(1):135, 2018.
- [20] M Kussner. Creating shapes: musicians and non-musicians visual representations of sound. In *Proceedings of 4th Int. Conf. of Students of Systematic Musicology, U. Seifert and J. Wewers, Eds. epOs-Music, Osnabrück (Forthcoming)*, 2012.
- [21] Olivier Lartillot. Soundtracer, 2018.
- [22] Marc Leman. *Embodied music cognition and mediation technology*. Mit Press, 2008.
- [23] Cynthia Liem, Meinard Müller, Douglas Eck, George Tzanetakis, and Alan Hanjalic. The need for music information retrieval with user-centered and multimodal strategies. In *Proceedings of the 1st international ACM workshop on Music information retrieval with user-centered and multimodal strategies*, pages 1–6. ACM, 2011.
- [24] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. 2015.
- [25] Kristian Nymoen, Baptiste Caramiaux, Mariusz Kozak, and Jim Torresen. Analyzing sound tracings: A multimodal approach to music information retrieval. In *Proceedings of the 1st International ACM Workshop on Music Information Retrieval with User-centered and Multimodal Strategies*, MIRUM '11, pages 39–44, New York, NY, USA, 2011. ACM.
- [26] Kristian Nymoen, Rolf Inge Godøy, Alexander Refsum Jensenius, and Jim Torresen. Analyzing correspondence between sound objects and body motion. *ACM Trans. Appl. Percept.*, 10(2):9:1–9:22, June 2013.
- [27] Kristian Nymoen, Jim Torresen, Rolf Godøy, and Alexander Refsum Jensenius. A statistical approach to analyzing sound tracings. *Speech, sound and music processing: Embracing research in India*, pages 120–145, 2012.
- [28] Joy E Ollen. *A criterion-related validity test of selected indicators of musical sophistication using expert ratings*. PhD thesis, The Ohio State University, 2006.
- [29] Denys Parsons. *The directory of tunes and musical themes*. Cambridge, Eng.: S. Brown, 1975.

- [30] Aniruddh D Patel. *Music, language, and the brain*. Oxford university press, 2010.
- [31] Art Samplaski. Interval and interval class similarity: Results of a confusion study. *Psychomusicology: A Journal of Research in Music Cognition*, 19(1):59, 2005.
- [32] Charles Seeger. On the moods of a music-logic. *Journal of the American Musicological Society*, 13(1/3):224–261, 1960.
- [33] Sandra E Trehub, Judith Becker, and Iain Morley. Cross-cultural perspectives on music and musicality. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 370(1664):20140096, 2015.
- [34] Sandra E Trehub, Dale Bull, and Leigh A Thorpe. Infants’ perception of melodies: The role of melodic contour. *Child development*, pages 821–830, 1984.
- [35] Kaiye Wang, Qiyue Yin, Wei Wang, Shu Wu, and Liang Wang. A comprehensive survey on cross-modal retrieval. *arXiv preprint arXiv:1607.06215*, 2016.

MAIN MELODY EXTRACTION WITH SOURCE-FILTER NMF AND CRNN

Dogac Basaran¹

Slim Essid²

Geoffroy Peeters¹

¹ CNRS, Ircam Lab, Sorbonne Université, Ministère de la Culture, F-75004 Paris, France

² LTCI, Télécom ParisTech, Université Paris Saclay, Paris, France

dogac.basaran@ircam.fr

ABSTRACT

Estimating the main melody of a polyphonic audio recording remains a challenging task. We approach the task from a classification perspective and adopt a convolutional recurrent neural network (CRNN) architecture that relies on a particular form of pretraining by source-filter nonnegative matrix factorisation (NMF). The source-filter NMF decomposition is chosen for its ability to capture the pitch and timbre content of the leading voice/instrument, providing a better initial pitch salience than standard time-frequency representations. Starting from such a musically motivated representation, we propose to further enhance the NMF-based salience representations with CNN layers, then to model the temporal structure by an RNN network and to estimate the dominant melody with a final classification layer. The results show that such a system achieves state-of-the-art performance on the MedleyDB dataset without any augmentation methods or large training sets.

1. INTRODUCTION

Automatic dominant melody estimation (AME) is a popular and rather challenging task in Music Information Retrieval (MIR). In general, AME can be defined as the estimation of fundamental frequencies that represent the pitch values of the dominant melody [24]. The source of the dominant melody could be a leading singing voice or an instrument. The difficulty is that there is usually a polyphonic accompaniment to the lead vocal/instrument, and that this accompaniment follows the melody rhythmically and harmonically, in the sense that chord progressions will naturally contain the dominant F_0 and/or its harmonics. As a consequence, it is not trivial to obtain a representation that discriminates the main melody from the background music. Hence, one of the main research directions in AME remains finding a salience representation that enhances the

fundamental frequency of the dominant melody against the possibly polyphonic background.

One of the most popular and rather simple salience representations is the Harmonic Sum Spectrum (HSS) [18] that consists of mapping the energy among harmonically related F_0 s. This has been used effectively in a popular melody extraction algorithm, jbcorsor-called *Melodia* [23]. Durrieu et. al. [11, 12] proposed a salience function where the dominant melody (singing voice or instrument) is modeled with a Source-Filter Nonnegative Matrix Factorization (SF-NMF). This method was later combined with HSS in [7] in order to obtain an enhanced salience representation. There also exist other methods that utilize a simple time-frequency representation, e.g., the Short Time Fourier Transform (STFT) or Constant Q-Transform (CQT), as a low-level representation of salience [13, 25].

Recently, Bittner et. al. [6] proposed a Convolutional Neural Network (CNN) system to learn salience representations based on harmonic CQT. The rationale for this approach is to learn harmonic relationships implicitly and to obtain a salience representation similar to (or better than) HSS.

Salience-based melody estimation methods usually use pitch tracking methods on top of salience representations to exploit the temporal relationships between dominant F_0 s. In [12], a Hidden Markov model (HMM) was adopted where the states represent the bins of the source activations, i.e. F_0 s. Then a threshold-based voicing estimation (melody/non-melody estimation) was applied. Another very popular pitch tracking method was proposed by Salamon et. al. [23] where the algorithm creates and characterizes pitch contours on top of HSS. Characteristics of these contours have proven very effective in voicing estimation [7, 23].

Recently, Deep Neural Networks (DNNs) have become very popular in MIR applications such as sound event detection [2, 4] and chord estimation [20]. The ability of DNNs to approximate any function with linear weights and non-linear activations, given enough data, makes such systems attractive for MIR tasks. That said, comparatively few attempts have been made to estimate dominant melody using neural networks. In [19, 22], bidirectional Long Short-Term Memory (LSTM) [15], a special kind of Recurrent Neural Network (RNN), are used for singing voice separation. Such networks are mostly used in modeling the



© Dogac Basaran, Slim Essid, Geoffroy Peeters. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Dogac Basaran, Slim Essid, Geoffroy Peeters. “Main Melody Extraction with source-filter NMF and CRNN”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

temporal information in time sequences. Recently, in [3], a hierarchical CNN structure similar to a stacked denoising autoencoder (SDA) [26] is used to learn a mapping between an STFT representation and a transcription similar to a piano roll. A tutorial on deep learning techniques for MIR tasks can be found in [9].

Although most of these DNNs perform end-to-end training, it has proven effective to use a more structured input data, such as harmonic-CQT [6]. Recently, [4] achieved state-of-the-art results in sound classification by using NMF activations as input as a form of pretraining.

Contributions. Inspired by these works, we propose a Convolutional-Recurrent Neural Network (CRNN) model whose pretraining is based on the SF-NMF model proposed in [12]. We show that with NMF-based pretraining, we can achieve state-of-the-art results without requiring large training datasets or data augmentation methods, and using relatively simpler networks in terms of training parameters. Our results clearly demonstrate the usefulness of a good input salience representation to the network, suggesting that performance would climb even higher if the SF-NMF model were improved. Our results are obtained on MedleyDB [5], which is a challenging dataset due to inclusion of singing voice and instrument melodies in a diverse set of music genres.

The rest of the paper is organized as follows: the proposed CRNN system and pretraining with SF-NMF are detailed in Section 2. Section 3 discusses the dominant melody estimation results obtained on the MedleyDB dataset, and also gives an analysis of SF-NMF-based salience and the comparison between different CRNN variants. Finally, some conclusions and future directions are given in Section 4.

2. SYSTEM OVERVIEW

The block diagram of the CRNN system we propose is given in Figure 1. In the first stage (Pretraining), we estimate an initial salience representation using the SF-NMF model. Then this salience is fed into a CNN (CNN stage), where the salience representation is further enhanced by learning local features. The CNN output activations are then fed into an RNN to exploit the long-term relationships between fundamental frequencies (RNN stage). Then in the final Classification stage, we classify the representations as melody/non-melody and give an estimate for F_0 at each time-frame where each class represents a semitone fundamental frequency. Note that the same procedure is applied in both the training and testing of the system.

In the design of the CRNN system, we are inspired by a similar CRNN proposed in [20] for chord recognition, where the network is interpreted as an encoder-decoder scheme. In the CRNN structure we propose, the CNN and RNN stages can also be treated together as an encoding stage (input sequence to mid-level salience representation) where the output is an enhanced salience representation that captures both spatial and temporal features. Then the classification stage acts as a decoding stage (mid-level representation to output sequence) where the salience is

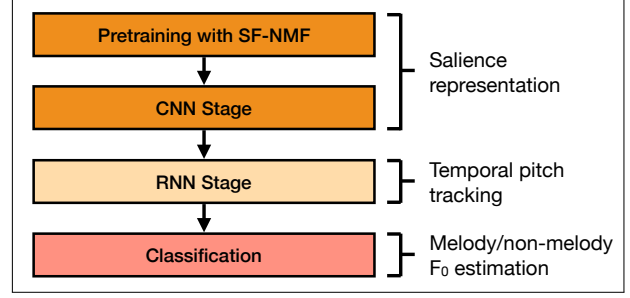


Figure 1: Block diagram of the proposed CRNN system with pretraining

mapped into a frame-based note representation.

2.1 Pretraining with SF-NMF

In [12], the dominant melody (voice/instrument) is modeled using a source-filter model. Assuming the mixing of the dominant melody and the accompaniment (background) is instantaneous, the source, filter and accompaniment parts are modeled with the SF-NMF model as follows:

$$\mathbf{V} \approx \hat{\mathbf{V}} = \mathbf{V}^{F_0} \odot \mathbf{V}^\Phi + \mathbf{V}^B$$

$$= \mathbf{W}^{F_0} \mathbf{H}^{F_0} \odot \mathbf{W}^\Phi \mathbf{H}^\Phi + \mathbf{W}^B \mathbf{H}^B \quad (1)$$

$$= \mathbf{W}^{F_0} \mathbf{H}^{F_0} \odot \mathbf{W}^\Gamma \mathbf{H}^\Gamma \mathbf{H}^\Phi + \mathbf{W}^B \mathbf{H}^B \quad (2)$$

where \mathbf{V} represents the power spectrogram of the signal, i.e., $\mathbf{V} = |\mathbf{X}|^2$ (where \mathbf{X} is the STFT of the audio signal to be analyzed); F_0 , Φ and B represents the source, filter and background respectively; \mathbf{W} and \mathbf{H} represent the basis and activation matrices; and \odot denotes the Hadamard product. The filter basis \mathbf{W}^Φ is further modeled with yet another NMF representation, as in [11]: $\mathbf{W}^\Phi = \mathbf{W}^\Gamma \mathbf{H}^\Gamma$.

In this model, the source, $\mathbf{V}^{F_0} = \mathbf{W}^{F_0} \mathbf{H}^{F_0}$, is assumed to have a harmonic structure. To ensure such a structure, the basis \mathbf{W}^{F_0} is pre-constructed (not estimated) such that each column represents the harmonic structure for one F_0 . Represented F_0 s start from a minimum frequency, i.e., $F_0 = 55\text{Hz}$, and they are logarithmically spaced, i.e., the ratio between consecutive F_0 values would be $2^{(1/60)}$ for a resolution of 5 bins per semitone. Such a construction enforces the corresponding row in the activation matrix \mathbf{H}^{F_0} to represent the activation of that specific F_0 , similar to a saliency representation. That is the rationale behind using \mathbf{H}^{F_0} as a saliency representation as in [7, 11, 12].

The main assumption with the filter, \mathbf{V}^Φ , is to have a smooth structure. One way to ensure such smoothness is to construct a basis \mathbf{W}^Φ from smooth filters in advance, similar to enforcing harmonic structure in the source \mathbf{V}^{F_0} . However it is not possible to directly construct \mathbf{W}^Φ with smooth basis filter structures since it depends on the dominant melody. In [11], it is proposed to represent \mathbf{W}^Φ with another NMF model, $\mathbf{W}^\Gamma \mathbf{H}^\Gamma$, where the columns of \mathbf{W}^Γ are constructed (not estimated) as simple and smooth band pass filters that are linearly spaced and overlapping. This structure forces \mathbf{W}^Φ to be smooth, thus ensuring that \mathbf{V}^Φ will be smooth as expected.

The accompaniment/background, $\mathbf{V}^B = \mathbf{W}^B \mathbf{H}^B$, is also represented with a standard NMF model where there are no constraints on the basis such as smoothness or being harmonic. In summary, the source basis \mathbf{W}^{F_0} and smooth filter basis \mathbf{W}^I are pre-constructed and the rest of the parameters \mathbf{H}^{F_0} , \mathbf{H}^I , \mathbf{H}^Φ , \mathbf{W}^B and \mathbf{H}^B are estimated using the standard alternating scheme and heuristic multiplicative updates.

In this work, for the SF-NMF model, we follow the parametrization given in [7] where the minimum and maximum frequencies represented in \mathbf{H}^{F_0} are chosen as $55Hz$ and $1760Hz$ respectively. We choose the resolution of the F_0 s as 5 bins per semitone which results in 60 bins per octave (bpo) and 301 bins in total per frame.

Note that due to the logarithmic spacing of the F_0 s where the consecutive frequencies have a ratio of $2^{1/60}$, one can tune the represented F_0 s with proper choice of the minimum frequency $F_{0,min}$. As an example, if $F_{0,min} = 55Hz$, the notes will be tuned to $A4 = 440Hz$ whereas if $F_{0,min} = 55.25Hz$, they will be tuned to $A4 = 442Hz$. This choice of tuning might depend on the target dataset. Here, we choose the tuning $A4 = 440Hz$ assuming that such tuning is more widely used. It is important to mention that this construction of F_0 s in \mathbf{W}^{F_0} cannot be generalized to all music genres, e.g., traditional Turkish music with makams. Hence the methods based on SF-NMF, as well as the proposed scheme, are limited in that sense.

Although we aim to classify the fundamental frequencies at semitone resolution, we initially choose a higher resolution for the F_0 s in \mathbf{W}^{F_0} . In practice, it is highly probable that a dominant voice or instrument will be slightly out-of-tune, and hence will not fit any of the represented F_0 s. In such cases, a high resolution representation of F_0 s might better describe these out-of-tune notes.

2.2 CNN stage

In order to enhance the \mathbf{H}^{F_0} -salience, we propose two different CNN architectures, which we denote as CNN1 and CNN2. In CNN architecture 1 (CNN1), we first decrease the F_0 resolution to semitones, then we train CNN layers to learn local structures, i.e., the confusions between semitones. In the second approach (CNN2), we follow the network proposed in [6]. Here, the network learns the features in the original resolution and within a semitone interval with one additional layer that learns the octave patterns.

Note that since each CNN architecture only applies 2D linear filters and non-linear activations, the input structure is not lost through the layers of the network. This provides an advantage of interpretable hidden layer activations and leads to a new form of salience as output where each row still represents the activation of a fundamental frequency.

In both architectures, rectified linear units (ReLus) are used as non-linear activations and are applied to each CNN layer output. Batch normalization is applied before each intermediate CNN layer input, as it has proven effective in the convergence of the network by reducing the internal covariance shift [16]. The columns of \mathbf{H}^{F_0} are normalized with l_1 norm before being fed into the CNN network. Such

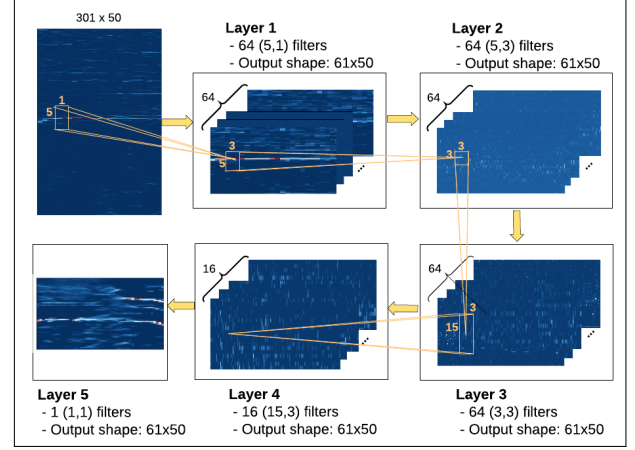


Figure 2: CNN Architecture 1 (CNN1).

a normalization is possible since the task at hand is the estimation of the melody; that is, only the position of the fundamental frequency is needed, not the exact energy.

2.2.1 CNN Architecture 1 (CNN1)

There are 5 layers in the CNN1 architecture. The first layer gathers the energy around each semitone by applying focused filters centered around each semitone frequency. In this layer, there are 64 (5x1) filters each with a stride (5,1). This way, not only is the energy focused on the semitones, but also the frequency resolution is decreased to the semitone scale from 5 bins per semitone (time resolution remains the same). The rationale behind the first layer is two-fold: First, the number of parameters is severely decreased by lowering the frequency resolution, i.e., it takes 5 times less filter parameters in order to learn features. Second, out-of-tune notes would already be represented in the vicinity of the corresponding semitone in the \mathbf{H}^{F_0} representation. Focused filters on semitones would gather the energy on the semitone that is a way of retuning the melody on the represented semitone fundamental frequencies.

In the following layers, zero padding is applied to convolutions to keep the dimensions unchanged. The second layer has 64 (5 x 3) filters that cover ± 2 semitone interval and roughly 30ms in time. Then the third layer has 64 (3 x 3) filters that cover ± 1 semitone and 30ms in time. The fourth layer has 16 (15 x 3) filters to learn note confusions in one octave. Filters cover ± 7 semitone interval and again 30ms in time. Then enhanced salience representation is obtained as the output of the final CNN layer that has only one (1x1) filter as in [6] but with a rectified linear unit instead of a sigmoid. The overall structure of CNN architecture 1 is shown in Figure 2.

2.2.2 CNN Architecture 2 (CNN2)

CNN2 is based on the network proposed in [6]. In this network, the resolution of the input remains the same throughout the layers of the CNN, i.e., no pooling is applied. Note that the input to CNN2 is \mathbf{H}^{F_0} ; therefore, the first layer of the network contains only a single channel instead of six.

As mentioned before, the overall system targets semitone resolution for the output fundamental frequencies. This requires a reduction in resolution somewhere in the system. In this architecture, we left the dimensionality reduction to the final classification layer.

2.3 RNN stage

Recurrent neural networks are mostly used in MIR and audio analysis tasks to model the dynamics of the observations, typically for chord recognition [20] and speech recognition [14]. Here, we use a single bidirectional Gated Recurrent Unit (BiGRU) layer to capture temporal relationships between F_0 s. A GRU is a special kind of RNN [8] where the units are able to model long-term temporal relationships whilst using a gate structure. It has the advantage of not suffering from the vanishing gradient problem of standard RNN and has proven to be easier to train compared to the LSTM alternative.

The number of units in a BiGRU layer should be chosen higher or equal to the output dimension of the preceding CNN network. In the BiGRU structure, actually two GRU layers are trained with the same input but in reverse directions to model the F_0 relationships from both directions. Later, the two layers are merged to have a single output.

2.4 Classification

The final layer of the system is a classifier where one class represents the non-melody and the rest of the 61 classes represent semitone fundamental frequencies between A1 and A6 (inclusive). The multiclass classification output is obtained with a single dense layer and softmax activations.

The overall system is trained minimizing the cross entropy loss between the softmax activations and true probabilities. A frame is classified as a non-melody frame only if the probability of non-melody class is higher than the rest. Regardless of this decision, F_0 is estimated for each frame by simply picking the most probable F_0 class among the 61 note classes. Note that even if the non-melody class has the highest probability, the second-highest probability gives a good estimation of the pitch.

An example output of the classification layer that is obtained from a CNN1 + RNN + Classification architecture is shown in Figure 3. In this example, \mathbf{H}^{F_0} input (top-left) gives a very good initial salience. Then the CNN1 output activations (top-right) further enhance the dominant part against the harmonic background. It is observed that the dominant F_0 classes mostly have the highest probabilities against the rest of the class probabilities (bottom-left).

3. EXPERIMENTS

In this section, we evaluate the proposed NMF-based CRNN system using the MedleyDB dataset [5]. For the annotations, we use the "Melody2" definition in MedleyDB that is the F_0 of the dominant melody at each time step, drawn from multiple sources. With this definition of melody, it is possible to have separate instruments or

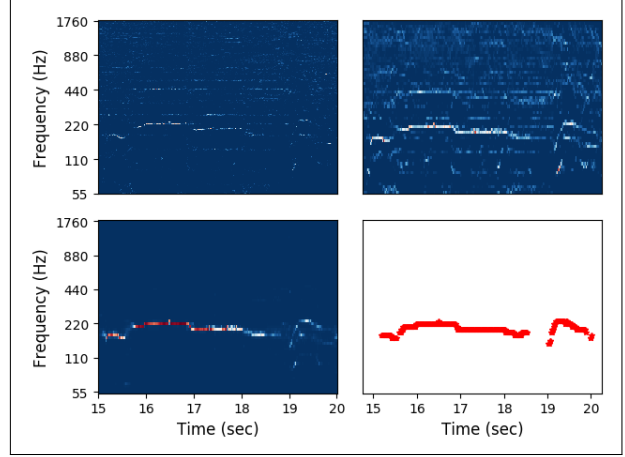


Figure 3: (Top-left) \mathbf{H}^{F_0} representation of a small audio excerpt as input to CRNN, (Top-right) CNN1 activations, (Bottom-left) Classifier activations of CRNN, (Bottom right) Ground-truth annotations.

voices as the source of dominant melody throughout a single song. Among 108 annotated songs in the dataset, 48 songs have predominant instrumental melody, 30 songs have predominant vocal melody and 30 songs have both predominant instrument and vocal melodies.

We randomly split the MedleyDB set into train, validation and test sets such that the tracks from the same artist do not belong to different sets following the artist conditional random splitting as in [6, 7]. There are 27 full-length tracks in the test set, 67 full-length tracks in the training set and 14 full-length tracks in the validation set. Note that we used the same test split with [6] in the MedleyDB in the rest of the experiments to be able compare the results.

We use the five standard evaluation metrics given in [24], namely: Raw Pitch Accuracy (RPA), Raw Chroma Accuracy (RCA), Overall Accuracy (OA), Voicing False Alarm (VFA) and Voicing Recall (VR). All the codes are written in Python and available online¹. CQT implementation is based on the *librosa* python package [21].

3.1 Network training

We trained three different networks with the following combinations of the architectures given in Section 2:

CRNN-1: CNN1 + 1 layer BiGRU (128 Units) + Classification layer;

CRNN-2: CNN2 + 1 layer BiGRU (160 Units) + Classification layer;

C-NN: CNN2 + Classification layer.

We further denote the network variants by prepending a label indicating the input to the network: "SF" for \mathbf{H}^{F_0} input and "CQT" for CQT input. Note that the CQT parameters are chosen such that the representation of a signal via \mathbf{H}^{F_0} or CQT would have the same dimensions².

¹ github.com/dogacbasaran/ismir2018_dominant_melody_estimation

² CQT parameters: Minimum $F_0=55\text{Hz}$, # of octaves = 5, bpo = 60

	CRNN-1	CRNN-2	Baseline
# of Param.	307,199	854,319	406,253

Table 1: The number of trainable parameters for CRNN-1, CRNN-2 and the baseline CNN network [6]

In the proposed CRNN structure, the purpose of the CNN stage is to learn local features, whereas the purpose of the RNN stage is to account for long term temporal relationships. This requires selecting relatively small patch lengths for the CNN layers but longer patch lengths for the RNN layer. For this purpose, we used different patch lengths for the CNN and RNN parts while jointly training them.

In all the models, the CNN layers are trained on either 0.29-second (25-frame) or 0.58-second (50-frame) patches and the RNN layer is trained on 5.8-second (500-frame) patches. The training is performed using mini-batches of 16 patches per batch. We use the ADAM optimizer [17], and reduce the learning rate if there is no improvement in validation loss after 20 epochs. The early stopping strategy is used if the validation loss is not decreased after 20 epochs. The maximum possible number of epochs is set to 200. All models were implemented with Keras 2.0 [10] and Tensorflow 1.0 [1] and tested using NVIDIA-Tesla K80 GPUs. The number of parameters for each network model is given in Table 1.

Note that, in the training, we do not benefit from any data augmentation method or from other larger datasets.

3.2 Results

We compare the outputs of all three models to a CNN-based melody tracking system [6], considered as a baseline, which proved to significantly outperform the previous state-of-the-art methods in [7, 23]. The evaluation results of [6] are available online.³ By choosing the same test split from the MedleyDB, we are able to compare these published results to ours without any re-evaluation. The evaluation results for all network variants (SF-CRNN-1, SF-CRNN-2, CQT-CRNN-2, SF-C-NN) and for the baseline are given in Figure 4. We use McNemar’s test on the classification results and provide p-values as a measure of significance whenever relevant⁴.

CQT vs. H^{F_0} as salience

We explore the usefulness of pretrained input by comparing the evaluation results of the CRNN-2 model when the input is CQT or H^{F_0} —i.e., comparing CQT-CRNN-2 and SF-CRNN-2. The results show that CRNN-2 model performs significantly better in OA ($p=0.0015$) and RCA ($p=0.0003$) scores when the input to the network is H^{F_0} . On average, results for SF-CRNN-2 are 6, 9 and 7 percentage points higher for OA, RPA and RCA, respectively.

The reason the CRNN-2 model performs better with pretrained input is that H^{F_0} provides a better initial

	H^{F_0}	CQT
RPA	0.538 ± 0.141	0.210 ± 0.16
RCA	0.648 ± 0.127	0.411 ± 0.15

Table 2: The comparison of RPA and RCA scores for H^{F_0} feature and CQT feature by simple peak-picking method.

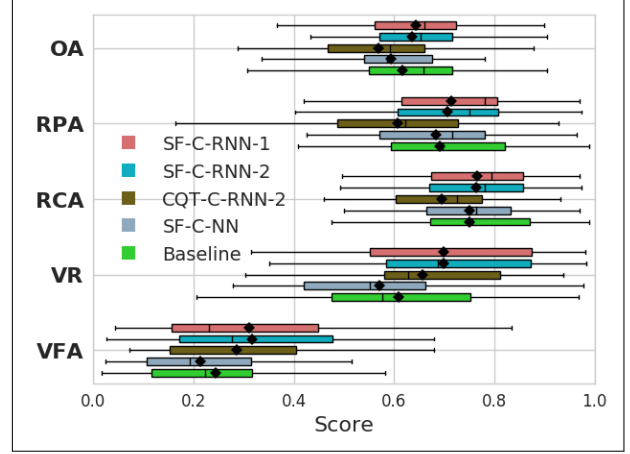


Figure 4: Evaluation metrics for SF-CRNN-1, SF-CRNN-2, CQT-CRNN-2, SF-C-NN and the baseline [6].

salience representation than the CQT. Ideally, a salience representation of melody should be discriminative for each target fundamental frequency against the polyphonic background music. We can analyze both H^{F_0} and CQT representations to see how well they fit this definition of “ideal” salience by performing a simple peak-picking strategy as in [6]. Specifically, the frequency with maximum amplitude/salience for each time frame point is chosen as the estimate of the fundamental frequency. We can compute the RPA and RCA scores using those estimates to see their performances as salience. The results obtained on the full MedleyDB dataset are given in Table 2. It can be seen that H^{F_0} performs nearly twice as well as the CQT representation in both RPA and RCA scores, showing that H^{F_0} provides a better initial salience to the CRNN networks.

SF-CRNN-2 model vs. Baseline CNN Network

The SF-CRNN-2 model uses the CNN-2 architecture in the CNN stage, the same CNN as the baseline. When we compare the evaluation results given in Figure 4, we observe that the SF-CRNN-2 model outperforms the baseline in the RPA ($p = 0.0015$) and VR ($p=0.052$) scores. The model has slightly higher OA and RCA scores on average than the baseline. On the other hand, SF-CRNN-2 has a higher number of network parameters (854,319) than the baseline CNN (406,253). This is due to the additional RNN layer that exists in SF-CRNN-2.

Comparison between variants SF-CRNN-1, SF-CRNN-2 and SF-C-NN

On average, SF-CRNN-1 performs slightly better than all other models in all metrics aside from VFA. Comparing SF-CRNN-1 and SF-CRNN-2, we observe that a similar or

³ github.com/rabitt/ismir2017-deepsalience

⁴ McNemar test is based on *statsmodel* package in python.

	OA	RPA	RCA	VR	VFA
SF-CRNN-1	0.444	0.595	0.677	0.556	0.423
Baseline	0.580	0.756	0.725	0.590	0.219

Table 3: Evaluation results for the track "MatthewEntwistle_TheFlaxenField" where the worst OA performance occurs against the baseline [6].

higher performance can be achieved by the low resolution CNN1 architecture and with far fewer training parameters (see Table 1). VR rates for SF-CRNN-1 and SF-CRNN-2 are significantly higher than the SF-C-NN; however, VFA rates are higher as well. This behavior could be due to the activations of the RNN layer that should force some sort of temporal smoothing on the salience representation.

On the other hand, the significantly better OA, RPA and RCA scores of SF-CRNN-2 relative to SF-C-NN suggest that the temporal tracking with RNN effectively improves the performance of the melody estimation.

Comparing the best performing network variant SF-CRNN-1 to the baseline, we observe that it outperforms the baseline on the OA ($p=0.052$), RPA ($p=0.0003$) and VR ($p=0.0015$) scores, and achieve those results with a less complex network in terms of network parameters (see Table 1). A track-level comparison by computing the overall accuracy differences for each track shows that SF-CRNN-1 performs better on 19 tracks out of 27.

The worst OA of SF-CRNN-1 occurs against the baseline with the "MatthewEntwistle_TheFlaxenField" track where the dominant melody consists only of instruments including Piano. The evaluation results for this track are given in Table 3. It is observed that both SF-CRNN-1 and baseline have relatively high VFA; however, the effect of this is minimal since the track mostly contains voiced frames. On the other hand, the OA score would be highly affected by the combination of high RPA and VR scores. For this track, although the baseline and SF-CRNN-1 have comparable VR rates, the RPA score of the baseline is better, which explains the difference in OA performance.

Singing voice vs. Instrument

Among the test set in MedleyDB, 16 tracks contain only instrumental dominant melody, 3 tracks contain only dominant singing voice melody and 8 tracks contain both⁵. Evaluation results in Table 4 show that SF-CRNN-1 performs better for singing voice melodies than instrument melodies. SF-CRNN-1 outperforms the baseline in overall accuracy for singing voice melodies and instrument melodies.

4. CONCLUSIONS AND FUTURE WORK

In this work, we have introduced a novel audio-based dominant melody estimation architecture using source-filter NMF as pretraining for a new variant of deep network for

⁵ The ratio of the dominant singing voice melody frames and the dominant instrumental melody frames among all voiced frames is 0.238 and 0.762, respectively.

	SF-CRNN-1		Baseline	
	S.V.	Ins.	S.V.	Ins.
OA	0.638	0.466	0.598	0.424
RPA	0.791	0.647	0.784	0.619
RCA	0.804	0.726	0.823	0.717

Table 4: OA, RPA and RCA scores for singing voice (S.V.) main melody and Instrument (Ins.) main melody for SF-CRNN-1 and baseline.

this task, namely a CNN-BiGRU scheme. We have shown that the proposed system achieves state-of-the-art performance on standard evaluation metrics, even significantly improving on it while maintaining a lower system complexity.

Analysis of \mathbf{H}^{F_0} as a salience representation shows that it provides a good initial salience in general with high RPA and RCA, even when performing melody estimation using frame-based salience peak-picking. The evaluation results clearly show the usefulness of SF-NMF-based pretraining in many aspects. We observe that when provided with a good initial salience input to the CRNN structure, the system performs considerably better without requiring any augmentation or additional training data. This encourages the idea of improving the pretraining part to obtain even more discriminative salience representations which will surely increase the melody estimation performance. For such improvements, SF-NMF is a good candidate since many other variants with various constraints such as smoothness or sparsity exist in the literature.

We observe that in the proposed CRNN structure, the CNN stage helps to improve the quality of the salience representation against \mathbf{H}^{F_0} . In addition, exploiting temporal information with the RNN significantly improves OA, RPA, RCA and VR. These two stages act similarly to an encoder scheme and the classification layer acts as the decoder. Therefore one can interpret the proposed CRNN as an encoder-decoder network where the encoder is used to obtain an enhanced salience representation and the decoder produces a frame-based transcription.

From a melody classification viewpoint, the MedleyDB dataset is quite challenging due to its diverse range of instrumentation and music genres. Also, there is an imbalance between the note classes and the non-melody class in the dataset. The CRNN network has proven effective in handling such imbalance when pretrained with an SF-NMF model.

A clear future direction to pursue is training the SF-NMF and CRNN jointly, learning the \mathbf{H}^{F_0} representation while minimizing the classification error.

5. ACKNOWLEDGEMENT

This project is partly funded by the *DigThatLick* project. We'd like to thank Rachel Bittner, Dr. Umut Simsekli and Dr. Jordan Smith for their valuable technical support.

6. REFERENCES

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [2] Sharath Adavanne, Konstantinos Drossos, Emre Çakir, and Tuomas Virtanen. Stacked convolutional and recurrent neural networks for bird audio detection. In *Signal Processing Conference (EUSIPCO), 2017 25th European*, pages 1729–1733. IEEE, 2017.
- [3] Stefan Balke, Christian Dittmar, Jakob Abeßer, and Meinard Müller. Data-driven solo voice enhancement for jazz music retrieval. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 196–200. IEEE, 2017.
- [4] Victor Bisot, Romain Serizel, Slim Essid, and Gaël Richard. Leveraging deep neural networks with non-negative representations for improved environmental sound classification. In *IEEE International Workshop on Machine Learning for Signal Processing MLSP*, 2017.
- [5] Rachel Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Bello. Medleydb: A multitrack dataset for annotation-intensive mir research.
- [6] R.M. Bittner, B. McFee, J. Salamon, P. Li, and J.P. Bello. Deep salience representations for f_0 estimation in polyphonic music. In *18th International Society for Music Information Retrieval Conference, ISMIR*, 2017.
- [7] J.J. Bosch, R.M. Bittner, J. Salamon, and E. Gómez. A comparison of melody extraction methods based on source-filter modelling. In *17th International Society for Music Information Retrieval Conference, ISMIR*, 2016.
- [8] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.
- [9] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark B. Sandler. A tutorial on deep learning for music information retrieval. *CoRR*, abs/1709.04396, 2017.
- [10] François Chollet. keras. <https://github.com/fchollet/keras>, 2015.
- [11] J. L. Durrieu, B. David, and G. Richard. A musically motivated mid-level representation for pitch estimation and musical audio source separation. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1180–1191, Oct 2011.
- [12] J. L. Durrieu, G. Richard, B. David, and C. Févotte. Source/filter model for unsupervised main melody extraction from polyphonic audio signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):564–575, March 2010.
- [13] B. Fuentes, A. Liutkus, R. Badeau, and G. Richard. Probabilistic model for main melody extraction using constant-q transform. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5357–5360, March 2012.
- [14] A. Graves, A. r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013.
- [15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.
- [16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, ICML’15*, pages 448–456. JMLR.org, 2015.
- [17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [18] Anssi Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *ISMIR*, pages 216–221, 2006.
- [19] S. Leglaive, R. Hennequin, and R. Badeau. Singing voice detection with deep recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–125, April 2015.
- [20] B. McFee and J.P. Bello. Structured training for large-vocabulary chord recognition. In *18th International Society for Music Information Retrieval Conference, ISMIR*, 2017.
- [21] Brian McFee, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Dana Lee, Oriol Nieto, Eric Batteberg, Dan Ellis, Ryuichi Yamamoto, Josh Moore, Rachel Bittner, Keunwoo Choi, Pius Friesch, Fabian Robert Stöter, Vincent Lostanlen, Siddhartha Kumar, Simon Waloschek, Seth, Rimvydas Naktinis, Douglas

Repetto, Curtis "Fjord" Hawthorne, CJ Carr, Waldir Pimenta, Petr Viktorin, Paul Brossier, João Felipe Santos, Jackie Wu, Erik, and Adrian Holovaty. *librosa/librosa*: 0.6.1, May 2018.

- [22] François Rigaud and Mathieu Radenen. Singing voice melody transcription using deep neural networks. In *ISMIR*, 2016.
- [23] J. Salamon and E. Gomez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, Aug 2012.
- [24] J. Salamon, E. Gomez, D. P. W. Ellis, and G. Richard. Melody extraction from polyphonic music signals: Approaches, applications, and challenges. *IEEE Signal Processing Magazine*, 31(2):118–134, March 2014.
- [25] Emmanuel Vincent, Nancy Bertin, and Roland Badeau. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):528–537, 2010.
- [26] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

FUNCTIONAL HARMONY RECOGNITION OF SYMBOLIC MUSIC DATA WITH MULTI-TASK RECURRENT NEURAL NETWORKS

Tsung-Ping Chen and Li Su

Institute of Information Science, Academia Sinica, Taiwan

{tearfulcanon, lisu}@iis.sinica.edu.tw

ABSTRACT

Previous works on chord recognition mainly focus on chord symbols but overlook other essential features that matter in musical harmony. To tackle the functional harmony recognition problem, we compile a new professionally annotated dataset of symbolic music encompassing not only chord symbols, but also various interrelated chord functions such as key modulation, chord inversion, secondary chords, and chord quality. We further present a novel holistic system in functional harmony recognition; a multi-task learning (MTL) architecture is implemented with the recurrent neural network (RNN) to jointly model chord functions in an end-to-end scenario. Experimental results highlight the capability of the proposed recognition system, and a promising improvement of the system by employing multi-task learning instead of single-task learning. This is one attempt to challenge the end-to-end chord recognition task from the perspective of functional harmony so as to uncover the grand structure ruling the flow of musical sound. The dataset and the source code of the proposed system is announced at <https://github.com/Tsung-Ping/functional-harmony>.

1. INTRODUCTION

Harmony and tonality represent the essence of Western tonal music. A complete analysis of the *functional harmony* in a musical piece needs one to utilize several interrelated concepts, such as chord progression, diatonic function, chord inversion, key modulation, to name but a few. These concepts are of fundamental importance in music theory, as they provide a systematic guide for one to understand how a phrase starts and how it ends, how one chord is related to another, how a chord is related to the key of the music, and more generally, how music works.

Computational approaches to analyzing musical harmony have gained wide attention in the past decades. Many works related to this topic, such as chord recognition [2, 6, 12, 18, 21, 23, 35], key detection [3, 9, 17, 27], and

chord sequence modeling and generation [5, 10, 14, 28, 31, 32], as the *sub-problems* of the complete *functional harmony recognition* problem, have been extensively studied. Among these sub-problems, chord recognition is arguably the most widely-investigated one.

Chord recognition focuses on the identification of *chord symbol*, i.e., symbols which indicate the root note, the chord quality (e.g., Major), and occasionally an extra interval number (e.g., seventh) of a chord.¹ Such a notation system provides direct instructions on chord construction, and therefore becomes prevalent in jazz and pop music. However, this notation system is insufficient for a more holistic analysis as it provides no information about *chord functions*.² For example, the *secondary chord*³ that plays an important role in the analysis of the hierarchical structure in a chord sequence is rarely discussed in the literature. Little efforts at such data annotation are due to it requires musicology expertise. As a result, there is no systematic studies on a more holistic recognition system based on all the above-mentioned concepts of functional harmony analysis, to the best of our knowledge. Although this topic has been extensively studied in the field of music information retrieval (MIR), the computers' ability of harmonic analysis is still quite limited.

In this paper, we discuss the *functional harmony recognition* problem. To tackle this problem, we first build a new dataset comprising five different chord functions, namely the key, primary degree, secondary degree, quality, and inversion. Since there is no unique and exact definition on functional harmony analysis of music, we alternatively consider the functional harmony recognition problem as the recognition of the above-mentioned five aspects, in order to facilitate the discussion in an engineering sense. We formulate this problem with the perspective of multi-task learning (MTL), and implement the system using the recurrent neural networks (RNN) with long short term memory (LSTM) units, a network structure that has been found useful in the audio chord recognition problem [6]. Experiments on the dataset show that the chord functions can be better resolved within the multi-task learning scenario

¹ For example, a chord played with notes C-E-G-B is notated as CM7.

² In the strict sense, the term *chord function* refers to the *diatonic function*, namely the Roman numeral annotation and the functions like tonic (T), dominant (D) and sub-dominant (S). In this paper we opt to choose a rather loose definition by regarding key, degree, and inversion also as some generalized 'functions' of a chord.

³ In this paper, the term *secondary chord* refers to the chord that does not serve the key. The borrowed chords, altered chords and the secondary dominant belong to this category.



compared to a single RNN structure, marking a step toward a more advanced computational music analysis framework.

2. RELATED WORK

2.1 Chord recognition and key detection

The chord recognition problem has been widely investigated on both the audio and symbolic data. In recent years, various machine learning techniques have been applied in this problem. In audio data processing, RNN-based methods such as the LSTM-based networks have been adopted due to its potential to model the long-term dependency of a time series [6, 12, 30]. Besides, [26] proposes a word2vec neural network to model the *harmony tension*, which also represents another perspective of chord function modeling. In symbolic data processing, early studies based on hand-crafted rules have considered the chord recognition of Roman numeral notations (i.e., chord symbol and tonality) [13]. [19] considered deep neural networks in chord recognition. Recent approaches based on machine learning, with evaluation performance include: [15] applies deep learning to identify non-chord tones in symbolic music data, and [21] uses a semi-Markov conditional random field (CRF) model for symbolic-level chord recognition.

Most of the studies on the key detection problem investigate the global key or home key detection [9, 17]. [17] proposes a global key finding algorithm with a convolutional neural network (CNN). The studies of key modulation detection are less seen, while there are still some related works such as local key detection [27].

2.2 Multi-task learning (MTL)

The MTL technique is proposed to fit one shared network to multiple related sets of labels, i.e., to learn multiple tasks at a time [20, 29]. If a primary task itself is difficult or is short of training data, its performance can be improved by introducing some auxiliary tasks by assuming these tasks share similar network structure.

MTL has exhibited great potential in MIR [11] since different attributes of music are often highly related. For example, in [34], the neural network is shared by the chord recognition task as well as the root note recognition task, and doing this can help to improve the accuracy of chord recognition. Similar ideas can also be seen in other models such as the multi-chain hidden Markov model (HMM) [22] and the dynamic Bayesian network [24]. Therefore, it suggests that the functional harmony problem itself is a multi-task learning problem, as determining one type of chord function usually needs the information of another.

2.3 Datasets for functional harmony recognition

Accurate annotation chord functions is hard to build in the audio domain, but rather feasible in the symbolic domain. There are a few datasets including annotation of some, if not all, chord functions: for example, the KSN dataset provides the annotation of chord and key modulation (i.e., the Roman numeral annotation) [16], the Theme And Variation Encodings with Roman Numerals (TAVERN) dataset

has Roman number chord annotation [8], and the Yale Classical Archive Corpus (YCAC) dataset has local tonic label and chord [33].

3. DATA AND LABELS

We propose the Beethoven Piano Sonata with Function Harmony (BPS-FH) dataset, which contains the symbolic musical data and functional harmony annotations of the 1st movements of 23 of Beethoven's Piano Sonatas.⁴ BPS-FH dataset provides a more consistent corpus in terms of musical form and genre with concise annotations for the analysis of harmony. As an ongoing work, the annotation will be extended to all the 32 piano sonatas.

3.1 Annotation process: harmonic analysis⁵

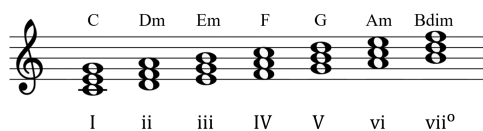
The BPS-FH dataset is annotated by an expert musicologist with a basic *harmonic analysis* process step-by-step. As opposed to the chord symbol annotation, the traditional harmonic analysis in music theory and musicology adopts a relative representation for chords to emphasize the interaction between chords in a given context. To perform harmonic analysis, there are several implicit processes:

- **Key identification:** the first step of harmonic analysis is to identify the *local key* according to context. Note that in many classical musical pieces, there is no exact analysis on the local key, for key modulation usually occurs, making it hard to find the local key in a certain excerpt.⁶ When the ambiguity occurs, finding a later cadence which is in a key-steady context, and then analyzing chords backwards might give a solution.
- **Segmentation:** since music itself is not represented originally as a sequence of chords, it is important to identify reasonable segments for labeling chords. A convincing segmentation should take the temporal rhythm and the harmonic rhythm (i.e., the rate at which the chords change) into consideration.
- **Harmonic reduction:** after determining the segments, each segment is reduced to a chord symbol (including chord root and chord quality) according to the tones within it. Harmonic reduction is a non-trivial and complicated process; there are many confusing factors, such as the non-chord tones, or the absence of harmonic tones in the segment.
- **Inversion recognition:** the inversion of a chord is determined by which of the notes is the bottom note, or bass note, of the chord. Typically, the lowest note in

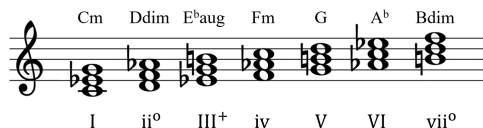
⁴ The 23 pieces are: No. 1, 3, 5, 6, 8, 11, 12, 13, 14, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 31, and 32. And all the repetitions in the sonatas are unfold.

⁵ In this paper, harmonic analysis refers to Roman numeral analysis.

⁶ In music, modulation is the act of changing from one key (tonic, or tonal center) to another. Generally speaking, the key of a musical piece refers to the global key which identifies the global tonic note and the final point of rest for the piece, while a modulation conducts the piece temporarily to another key, that is, a local key, which replaces the global tonic with a temporary tonic in a local area.



(a) Diatonic function in C major.



(b) Diatonic function in C minor (harmonic minor scale).

Figure 1: Illustration of diatonic functions in relation to the diatonic chords of the given keys. Note that in minor key, the superscript ⁺ is added to the *mediant* because it is an augmented chord.

a segment would be considered as the bass of the reduced chord. However, the lowest note is not always regarded as the bass notes; the pedal point is one of such examples.

- **Labeling diatonic functions:** after determining the key and the chord symbol, a *function* is assigned to the chord. In a major key, the following Roman numerals are used to represent the functions of diatonic chords: I (tonic), ii (super-tonic), iii (mediant), IV (sub-dominant), V (dominant), vi (sub-mediante), and vii° (leading). The capital numerals denote major chords, the lowercase numerals denote minor chords, and the superscript ° denotes diminished chords. Figure 1 shows the details of diatonic functions in both major and minor keys.

Figure 3b exhibits a brief example of harmonic analysis for the excerpt in Figure 3a. It is worth mentioning some possible confusions when analyzing harmony on this example: at measure 83, there are two non-chord tones, G at the 1st beat, and E^b at the second half of the 2nd beat, both of which might be confusing for harmonic reduction. Especially, the existence of the non-chord tone G prevents the note E (the last note of measure 82) from directly resolving to F, and blurs the boundary between F-minor key and E^b-major key. Hence, the key modulation might occur at measure 83 as labeled, but might also occur at measure 84 or even 85. It should be acknowledged that harmonic analysis is inherently subjective, and the confounding effect of subjectivity may affect the performance of a chord recognition system in many ways [25]. Details about the harmonic analysis techniques and labeling paradigms can be found in [1] and [4].

3.2 Annotations in the BPS-FH Dataset

A fundamental harmonic analysis provides the information of key, degree, quality and inversion. Therefore, the BPS-FH dataset has the corresponding annotations as follows:

- **Key:** the key to which a chord belongs in a local area. Since key modulation is essential in piano sonata, we trace the change of key, that is, we specify the *local key*, or temporary tonic, so as to show that how a key deviates from the global one during the course of the movement.
- **Primary degree and secondary degree:** degree refers to the position of a chord's root on the diatonic scale of a key.⁷ There are seven possible degrees on a diatonic scale, that is, 1, 2, ..., 7. We use a pair of degrees, *primary degree* and *secondary degree*, for both diatonic chords and secondary chords. Primary degree indicates the position of the temporary tonic on the scale, while secondary degree denotes the position of the chord's root based on the temporary tonic; the couple of degrees is represented as secondary degree/primary degree. In the case of diatonic chord, the primary degree is always 1. That is, the temporary tonic is the same as that of the current key. As for the secondary chord, both the primary degree and the secondary degree can be any possible degree. For example, the diatonic chord V is represented as 5/1, while the secondary chord V/IV is represented as 5/4.
- **Quality:** chord quality is defined by the intervals within a chord. For instance, a major triad has a major third and a perfect fifth above its root. 10 types of chord quality are identified in the dataset, which are major triad (M), minor triad (m), augmented triad (a), diminished triad (d), major seventh (M7), minor seventh (m7), dominant seventh (D7), diminished seventh (d7), half-diminished seventh (h7), and augmented sixth (a6).
- **Inversion:** inversion of a chord describes which of the tones in a chord is the bass note. For example, the C-major triad has three candidates, C, E and G, as its bass, and thus has three possible inversions (root position is regarded as one inversion in the context). For triads and seventh chords, there are totally four possible inversions: the 0th inversion (root position), 1st inversion ($\frac{6}{3}$ or $\frac{6}{3}$ for triad, and $\frac{6}{5}$ for seventh chord), 2nd inversion ($\frac{6}{4}$ for triad, and $\frac{4}{3}$ for seventh chord), and 3rd inversion ($\frac{4}{2}$ for seventh chord). Note that only seventh chords have 3rd inversion.

In summary, the BPS-FH dataset contains 86,950 note events, 29 different keys, 531 key modulations, and 7,394 chord labels.⁸

⁷ For example, the chord C major triad has the degree 1 in C major key, while has the degree 4 in G major key.

⁸ Among all the chords, 3,438 are inverted; 839 are secondary chords; 2,951 are major triads; 1,356 are minor triads; 25 are augmented triads; 286 are diminished triads; 30 are major seventh chords; 86 are minor seventh chords; 2,037 are dominant seventh chords; 453 are diminished seventh chords; 104 are half diminished seventh chords; 66 are augmented sixth chords.

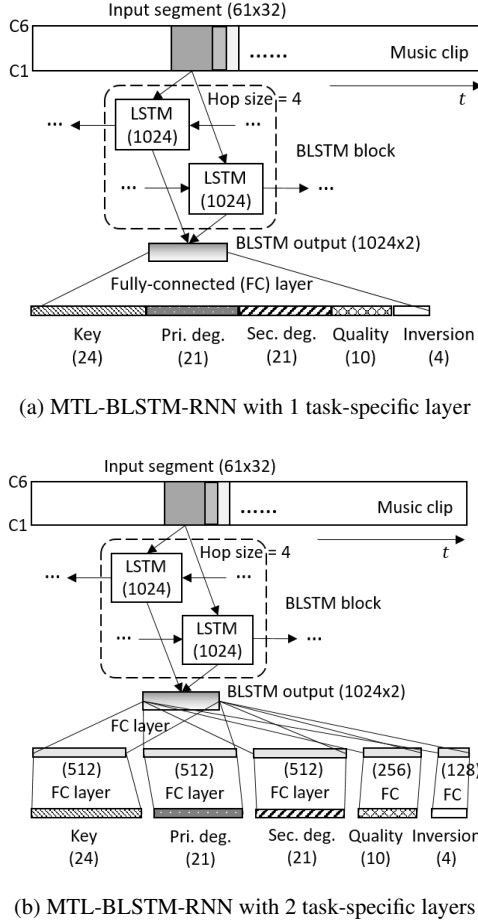


Figure 2: Illustration of the MTL-based functional harmony recognition system, with a BLSTM-RNN model taking a data stream as input.

3.3 Data representation

The input data is represented in the format of a 61-key piano-roll, with the pitch range from C1 to C6 (middle C = C4); the duration of each note is measured in crotchet beats. For time resolution, we define a 32th note as the minimal time step. All the note events out of this pitch range are transposed to fit in, while the durations of the note events whose lengths are shorter than the minimal time resolution are set to be the same as the time resolution. A piano-roll at a time instance is called a *frame*.

As shown in Figure 2, the input of the LSTM cell is a segment of data with 32 frames. That is, for a musical piece with 4/4 meter, the length of a segment is 4 beats (or equivalently 1 bar). And a musical *clip* containing 64 segments is fed to the neural networks. The *hop size* for the neural networks is 4 frames (or half a beat.)

4. MODEL

We employ recurrent neural networks (RNN) with bidirectional long-short-term memory (BLSTM) units (denoted as BLSTM-RNN hereafter) to model sequences of functional harmony, by using the above-mentioned data representa-

Label	Dim	Content
Key	24	24 major and minor keys
Pri. deg.	21	7 Roman numerals by 3 (neutral, \sharp , \flat)
Sec. deg.	21	7 Roman numerals by 3 (neutral, \sharp , \flat)
Quality	10	M, m, a, d, M7, m7, D7, d7, h7, a6
Inversion	4	0th, 1st, 2nd, 3rd

Table 1: Chord function labels in the BPS-FH dataset, including key, primary degree (pri. deg.), secondary degree (sec. deg.), chord quality, and chord inversion.

Set	Piece No.
Training	1, 3, 5, 11, 16, 19 20, 22, 25, 26, 32
Validation	6, 13, 14, 21, 23, 31
Testing	8, 12, 18, 24, 27, 28

Table 2: The pieces in training, validation, and testing sets.

tion as input. Such kind of model has been widely used in audio chord symbol recognition problems [6, 7, 12], and has been found capable in learning long-term information such as music structure. Specifically, we consider the following two types of networks:

- MTL-BLSTM-RNN with 1 task-specific layer: as shown in Figure 2a, we adopt a simple BLSTM architecture with 1024 hidden units for multi-task learning. The outputs of the forward and the backward cells are concatenated and form a 1024-by-2 matrix. This matrix is flattened and is connected to the output layer through a fully-connected layer. The output layer is a 80-D vector containing the classes for the five tasks listed in Table 1. Each class is one-hot encoding, and the Softmax function is used for the output vector.
- MTL-BLSTM-RNN with 2 task-specific layers: as shown in Figure 2b, the architecture is the same as the above, but with an additional task-specific layer before the output layer, in order to further increase the model capacity.

Moreover, to verify the advantage of MTL, we also consider the single-task learning (STL) as a baseline approach, where the same BLSTM-RNN is used. As a result, there are five networks in the STL-BLSTM-RNN model, each for one chord function recognition task respectively, and are trained individually in the experiment.

5. EXPERIMENT

5.1 Experimental settings

In the training stage, we divide the 23 pieces in the dataset into three parts, namely the training set, the validation set, and the testing set. Each part contains overlapped clips which are the input instances of the BLSTM networks. Each clip contains 64 segments, and the overlap between two consecutive clips is 32 segments. To balance the data

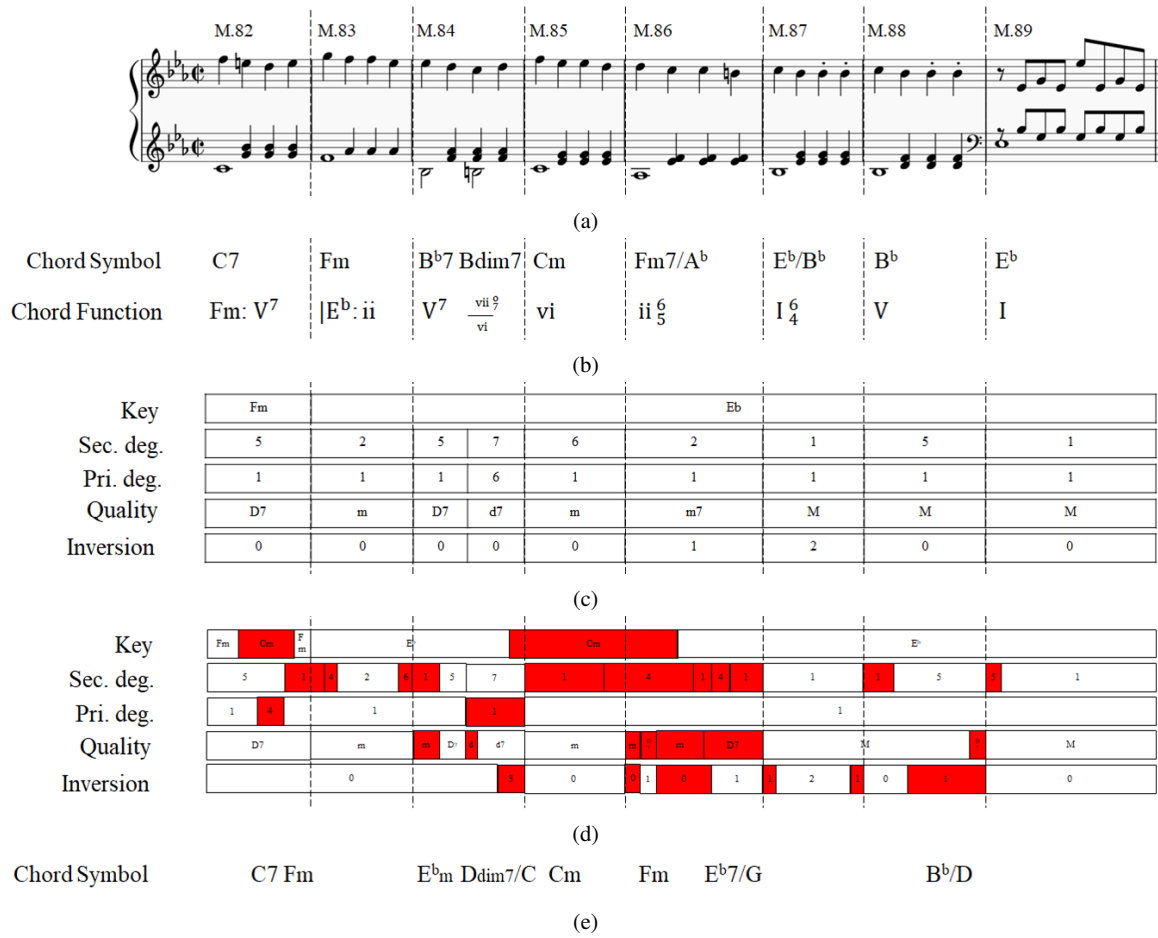


Figure 3: (a) An excerpt from the 1st movement of Beethoven’s Piano Sonata No. 8, MM. 82-89. (b) The harmonic analysis of this excerpt represented in both chord symbol and chord function. Note that the slash used in chord symbol stands for an inversion, and the note behind the slash denotes the bass of the chord. In the analysis, this expert starts from F minor, modulates to E^b major at measure 83, and finally ends with an authentic cadence. (c) 5 types of annotations representing the functions in (b). (d) The testing result of chord function recognition of the excerpt. Wrong predictions are marked in red. (e) The translation of the result in (d) to chord symbol. For the sake of concision, only the wrong predictions lasting at least one quarter note are translated.

distribution among all possible keys, We perform data augmentation by transposing all the clips into 12 keys. As a result, there are 7,320 clips for training, 3,672 clips for validation, and 3,636 clips for testing. Table 2 shows the musical pieces used in each set. In the experiment, we compare the following two tasks:

- Chord symbol recognition: with the symbolic data of music as inputs, the model outputs chord symbol predictions in a segment-wise manner. We used 25 chord classes for the output layer, that is, 24 classes for 12 major triads and 12 minor triads, and an ‘other’ class for chords not belonging to either major triads or minor triads.
- Chord function recognition: similar as the chord symbol recognition, but the outputs of the model are chord functions containing five components.

Both the MTL and STL schemes are tested on the chord

function recognition task, while the chord symbol recognition is tested with STL. For the chord function recognition task with MTL scheme, the outputs of the five chord functions are translated to chord symbol to evaluate the performance in terms of chord symbol recognition. And for the chord function recognition task with STL scheme, five different networks are trained individually for the evaluation of chord function recognition.

All networks are implemented with TensorFlow, and are trained using stochastic gradient descent with the Adam optimization method. For training objective, we compute categorical cross-entropy between targets labels and network outputs, and include a L2 regularization term. Moreover, to prevent over-fitting and to speed up training convergence, recurrent batch normalization is applied, and the dropout rate at the input and the output of the LSTM cell is set to be 0.5.

Task	Model	Key	Degree	Secondary	Quality	Inversion	Overall	Translation
Chord Symbol	STL-BLSTM-RNN	—	—	—	—	—	72.71	—
Chord Function	STL-BLSTM-RNN	67.06	48.31	9.38	61.87	57.95	23.57	56.05
	MTL-BLSTM-RNN with 1 task-specific layer	68.48	50.49	10.96	62.31	60.04	25.53	56.91
	MTL-BLSTM-RNN with 2 task-specific layers	66.65	51.79	3.97	60.59	59.10	25.69	56.25

Table 3: Accuracy (in %) of functional harmony recognition and comparison between multi-task BLSTM and single-task BLSTM. In the table, *Degree* stands for the accuracy of correctly predicting both the primary and secondary degrees of all chords; while *Secondary* indicates the accuracy of correctly predicting the degrees of secondary chords.

5.2 Evaluation metrics

We compute the segment-level *accuracy*, the ratio between the number of correct detection and the number of total segments in the testing set, for each category. Only one accuracy value is computed in the case of chord symbol recognition, while six types of accuracies are computed in the case of chord function recognition, namely the accuracies of key, degree, secondary chord, quality, inversion, and finally, the overall accuracy. Note that the accuracy of secondary chord is computed when a secondary chord does exist. The overall accuracy counts the segments in which the five chord function detections are all correct. An extra translation accuracy is computed to examine the performance of chord function recognition in terms of chord symbol recognition.

5.3 Results

Table 3 shows the results of chord symbol recognition and chord function recognition. In the task of chord symbol recognition, the STL-BLSTM-RNN-based model gives an accuracy of 72.71%. In comparison to other existing works which also estimate chord symbols on classical music datasets such as [12,21], this result is acceptable while also reveals the room for improvement in recognizing chords in western classical music.

In comparison with the chord symbol recognition task, performing the chord function recognition task is much more challenging. Specifically, the best overall accuracy among all chord function recognition tasks is only 25.69%, which is far from that of chord symbol recognition. This is partly because there are as many as 10 chord qualities for the model to predict, and partly because tonal harmony itself is complicated and equivocal. On the other hand, MTL-BLSTM-RNN model with 1 task-specific layer outperforms the single-task one for all chord functions. This indicates that employing multi-task learning results in a promising improvement. Among all chord functions, the improvements of predicting degree and inversion are the most significant, with 2.18% and 2.09% increases in accuracy respectively. This consequence may result from the fact that identifying the degree and identifying the inversion of a chord are relatively difficult in classical music, and thus benefit more from multi-task learning. Moreover, the accuracies of secondary chord are very low for all experiment settings; adding one more task-specific layer even degrades its performance. This displays the difficulty of learning the chord representation consisting of semantic

information. Finally, we translate the predictions of chord function recognition tasks into chord symbol to examine the performance in terms of chord symbol recognition. It comes as no surprise that the all the translation accuracies are lower than that of chord symbol recognition. This again marks the challenge of chord function recognition, as it needs to consider not only the elements constructing a chord symbol, but also more high-level semantic information such as local key and degree.

An example of the chord function recognition result is shown in Figure 3d. Because the prediction is segment-wise, there are numbers of discontinuities in the predicted sequences. This issue can be addressed by further incorporating temporal smoothing models such as the CRF [21] in the future. A close examination of this result shows that although the model gives ‘wrong’ predictions, part of the predictions does match the ground truth on the level of chord symbol. For instance, as demonstrated in Figure 3d & 3e, there are whole-bar error predictions in key and secondary degree at measure 85; however, these detections become correct if we translate them into chord symbol: they are both C minor triads, albeit in different keys. In fact, further analysis points out that the prediction of the modulation to C minor at measure 85 is also meaningful: there does exist a potential modulation for there is a *tonicization* of vi constructed by the previous chord vii²/vi at the second half of the measure 84. From this point of view, the model does provide more insight into the analysis of tonal structure in this excerpt, as an expert analyzer can do.

6. CONCLUSION AND FUTURE WORK

We have given a systematic investigation on the problem of functional harmony recognition of symbolic data based on deep learning techniques. Experiments on the proposed Beethoven Piano Sonata with Functional Harmony dataset indicate that functional harmony recognition is a task much more challenging than the chord symbol recognition, and a multi-task learning framework provides a promising solution better than a single-task one. Detailed analysis results not only give insightful interpretation, and also pose further challenging problems on recognizing key modulation, secondary degree, etc., all with its semantic level higher than chord symbols. This work marks a preliminary step towards a holistic approach of modeling functional harmony, and also provide the potential for one to analyze interpretable and meaningful music patterns from music, or to explore some alternative interpretation of music in the study of computational music analysis.

7. REFERENCES

- [1] Edward Aldwell and Carl Schachter. *Harmony and Voice Leading*. Belmont: Thomsom Schirmer, 3 edition, 2003.
- [2] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In *Proceedings of the 14th International Conference on Music Information Retrieval (ISMIR)*, pages 335–340, 2013.
- [3] Wei Chai and Barry Vercoe. Detection of key change in classical piano music. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, pages 468–473, 2005.
- [4] Nicholas Cook. *A guide to musical analysis*. London: J. M. Dent & Sons, 1987.
- [5] W. Bas de Haas, José Pedro Rodrigues Magalhães, Frans Wiering, and Remco C. Veltkamp. Automatic functional harmonic analysis. *Computer Music Journal*, 37(4):37–53, 2013.
- [6] Junqi Deng and Yu-Kwong Kwok. Large vocabulary automatic chord estimation using deep neural nets: Design framework, system variations and limitations. *arXiv preprint arXiv:1709.07153*, 2017.
- [7] Junqi Deng and Yu-Kwong Kwok. Large vocabulary automatic chord estimation using bidirectional long short-term memory recurrent neural network with even chance training. *Journal of New Music Research*, 47(1):53–67, 2018.
- [8] Johanna Devaney, Claire Arthur, Nathaniel Condit-Schultz, and Kirsten Nisula. Theme and variation encodings with roman numerals (TAVERN): A new data set for symbolic music analysis. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.
- [9] Emilia Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra, 2006.
- [10] Mark Thomas Granroth-Wilding. *Harmonic Analysis of Music Using Combinatory Categorical Grammar*. PhD thesis, The University of Edinburgh, 2013.
- [11] Philippe Hamel, Matthew E. P. Davies, Kazuyoshi Yoshii, and Masataka Goto. Transfer learning in mir: Sharing learned latent representations for music audio classification and similarity. In *Proceedings of the 14th International Conference on Music Information Retrieval (ISMIR)*, pages 9–14, 2013.
- [12] Takeshi Hori, Kazuyuki Nakamura, and Shigeki Sagayama. Music chord recognition from audio data using bidirectional encoder-decoder LSTMs. In *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2017, pages 1312–1315. IEEE, 2017.
- [13] Plácido R Illescas, David Rizo, and José Manuel Iñesta Quereda. Harmonic, melodic, and functional automatic analysis. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 165–168, 2007.
- [14] Nori Jacoby, Naftali Tishby, and Dmitri Tymoczko. An information theoretic approach to chord categorization and functional harmony. *Journal of New Music Research*, 44(3):219–244, 2015.
- [15] Yaolong Ju, Nathaniel Condit-Schultz, Claire Arthur, and Ichiro Fujinaga. Non-chord tone identification using deep neural networks. In *Proceedings of the 4th International Workshop on Digital Libraries for Musicology*, pages 13–16. ACM, 2017.
- [16] Hitomi Kaneko, Daisuke Kawakami, and Shigeki Sagayama. Functional harmony annotation database for statistical music analysis. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR): Late Breaking session*, 2010.
- [17] Filip Korzeniowski and Gerhard Widmer. End-to-end musical key estimation using a convolutional neural network. In *Proceedings of the 25th European Signal Processing Conference (EUSIPCO)*, pages 966–970, 2017.
- [18] Filip Korzeniowski and Gerhard Widmer. On the futility of learning complex frame-level language models for chord recognition. *arXiv preprint arXiv:1702.00178*, 2017.
- [19] Pedro Kröger, Alexandre Passos, Marcos Sampaio, and Givaldo de Cidra. Rameau: A system for automatic harmonic analysis. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 273–281, 2008.
- [20] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921, 2015.
- [21] Kristen Masada and Razvan Bunescu. Chord recognition in symbolic music using semi-markov conditional random fields. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pages 23–27, 2017.
- [22] Matthias Mauch. *Automatic chord transcription from audio using computational models of musical context*. PhD thesis, Queen Mary University of London, 2010.
- [23] Brian McFee and Juan Pablo Bello. Structured training for large-vocabulary chord recognition. In *Proceedings of the 18th International Conference on Music Information Retrieval (ISMIR)*, pages 188–194, 2017.

- [24] Yizhao Ni, Matt McVicar, Raúl Santos-Rodriguez, and Tijl De Bie. An end-to-end machine learning system for harmonic analysis of music. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1771–1783, 2012.
- [25] Yizhao Ni, Matt McVicar, Raúl Santos-Rodriguez, and Tijl De Bie. Understanding effects of subjectivity in measuring chord estimation accuracy. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(12):2607–2615, 2013.
- [26] Ali Nikrang, David R. W. Sears, and Gerhard Widmer. Automatic estimation of harmonic tension by distributed representation of chords. *arXiv preprint arXiv:1707.00972*, 2017.
- [27] Hélène Papadopoulos and Geoffroy Peeters. Local key estimation based on harmonic and metric structures. In *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx)*, pages 1–8, 2009.
- [28] Christopher Raphael and Joshua Stoddard. Functional harmonic analysis using probabilistic models. *Computer Music Journal*, 28(3):45–52, 2004.
- [29] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.
- [30] Siddharth Sigtia, Nicolas Boulanger-Lewandowski, and Simon Dixon. Audio chord recognition with a hybrid recurrent neural network. In *Proceedings of the 16th International Conference on Music Information Retrieval (ISMIR)*, pages 127–133, 2015.
- [31] David Temperley and Daniel Sleator. Modeling meter and harmony: A preference-rule approach. *Computer Music Journal*, 23(1):10–27, 1999.
- [32] Christopher William White. A corpus-sensitive algorithm for automated tonal analysis. In *Mathematics and Computation in Music*, pages 115–121. Springer, 2015.
- [33] Christopher William White and Ian Quinn. The yale-classical archives corpus. *Empirical Musicology Review*, 11(1), 2016.
- [34] Mu-Heng Yang, Li Su, and Yi-Hsuan Yang. Highlighting root notes in chord recognition using cepstral features and multi-task learning. In *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1–8. IEEE, 2016.
- [35] Xinquan Zhou and Alexander Lerch. Chord detection using deep learning. In *Proceedings of the 16th International Conference on Music Information Retrieval (ISMIR)*, pages 52–58, 2015.

A SINGLE-STEP APPROACH TO MUSICAL TEMPO ESTIMATION USING A CONVOLUTIONAL NEURAL NETWORK

Hendrik Schreiber

tagtraum industries incorporated

hs@tagtraum.com

Meinard Müller

International Audio Laboratories Erlangen

meinard.mueller@audiolabs-erlangen.de

ABSTRACT

We present a single-step musical tempo estimation system based solely on a convolutional neural network (CNN). Contrary to existing systems, which typically first identify onsets or beats and then derive a tempo, our system estimates the tempo directly from a conventional mel-spectrogram in a single step. This is achieved by framing tempo estimation as a multi-class classification problem using a network architecture that is inspired by conventional approaches. The system’s CNN has been trained with the union of three datasets covering a large variety of genres and tempi using problem-specific data augmentation techniques. Two of the three ground-truths are novel and will be released for research purposes. As input the system requires only 11.9s of audio and is therefore suitable for local as well as global tempo estimation. When used as a global estimator, it performs as well as or better than other state-of-the-art algorithms. Especially the exact estimation of tempo without tempo octave confusion is significantly improved. As local estimator it can be used to identify and visualize tempo drift in musical performances.

1. INTRODUCTION

Undoubtedly, the *tempo* of a musical piece is one of its main characteristics. Its estimation is often defined as measuring the frequency with which humans “tap” along to the beat. This is notably different from *beat tracking*, which aims at determining individual beat positions. If the tempo of a musical piece stays constant throughout the whole performance, it is called *global tempo*. It can be represented by a single number usually specified in *beats per minute* (BPM). Global tempi often occur in genres like Rock, Pop, and Dance music. The method proposed in this paper was primarily developed for estimating the tempo of short excerpts, but can also be applied to global tempo estimation.

Many different approaches to tempo estimation have been taken in the past. Gouyon et al. [11] provided a comparative evaluation of the systems that participated in the ISMIR 2004 contest, the first large-scale evaluation of

tempo induction algorithms. Five years later, Zapata and Gómez gave an updated overview [39]. To our knowledge, the most recent comprehensive evaluations are presented in [2, 25, 31]. For a textbook-style introduction see [22].

Early tempo estimation methods often combined signal processing with heuristics. Scheirer [28] for example used bandpass filters, followed by parallel comb filters, followed by peak picking. Klapuri et al. [17] replaced the conventional bandpass approach with STFTs, producing 36 band spectra. By differentiating and then half-wave rectifying the power in each band, they created band-specific onset strength signals (OSS), which were then combined into four accent signals and fed into comb filters in order to detect periodicities. Instead of processing an OSS with comb filters, several other methods have been proposed. Among them autocorrelation [1, 22], clustering of inter-onset intervals (IOI) [5, 33], and the discrete Fourier transform (DFT) [22, 23].

Recent approaches put emphasis on finding not just a periodicity, but on finding one corresponding to the perceived tempo, trying to avoid common errors by a factor of 2 or 3, so-called *octave errors* [11, 31]. The methods used range from genre classification (e.g., obtained by a genre classification component) [14, 32], secondary tempo estimation [30], and the discrete cosine transform of IOI histograms [7], to machine learning approaches like Gaussian mixture models (GMM) [24], support vector machines (SVM) [9, 25], k-nearest neighbor classification (k-NNC) [37, 38], neural networks [6], and random forests [31].

Another area of active research aims at creating a better OSS through the use of neural networks. Elowsson [6] uses harmonic/percussive source separation and two different feedforward neural networks to classify a frame as beat or non-beat. Böck et al. [2] use a bidirectional long short-term memory (BLSTM) recurrent neural network (RNN) to map spectral magnitude frames and their first order differences to beat activation values. These are then processed further with comb filters. For their dancing robot application, Gkiokas et al. [10] use a convolutional neural network (CNN) to derive a beat activation function, which is then used for beat tracking and tempo estimation.

What all these methods have in common is the *multi-step* approach of decomposing the signal into sub-bands, deriving some kind of OSS, detecting periodicities, and then trying to pick the best one. As Humphrey et al. [15] point out, this can be described as a deep architecture con-



© Hendrik Schreiber, Meinard Müller. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: Hendrik Schreiber, Meinard Müller. “A single-step approach to musical tempo estimation using a convolutional neural network”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

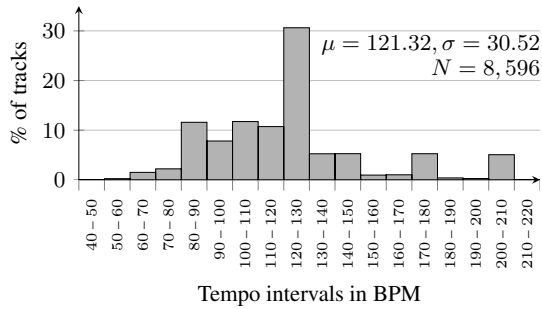


Figure 1: Tempo distribution for the Train dataset consisting of LMD Tempo, MTG Tempo, and EBall.

sisting of multiple components (“layers”) that has evolved naturally. But to the best of our knowledge, nobody has replaced the traditional multi-component architecture with a single deep neural network (DNN) yet. In this paper we describe a CNN-based approach that estimates the local tempo of a short musical piece or excerpt based on mel-scaled spectrograms in a single step, i.e., without explicitly creating mid-level features like an OSS or a beat activation function that need to be processed further by another, separate system component. Using averaging, we can combine multiple local tempi into a global tempo.

The remainder of this paper is structured as follows: Section 2 introduces our training datasets. Then Section 3 describes the signal representation, network architecture, network training, and how we combine multiple local estimates into a global estimate. In Section 4 we evaluate our global tempo estimation approach quantitatively by benchmarking against known datasets and state-of-the-art algorithms. Then we discuss local tempo estimation qualitatively using samples from different genres and eras. Finally, in Section 5 we present our conclusions.

2. TRAINING DATASETS

Our goal is to create a general purpose system that does not suffer from strong genre-bias. Therefore we avoid cross-validation on small datasets and instead created a large, multi-genre training dataset, consisting of three smaller datasets: One derived from a subset of the Lakh MIDI dataset (LMD) [27], a subset of the GiantSteps MTG key dataset (MTG Key) [8]¹, and a subset of the Extended Ballroom [20] dataset. Two of the derived ground-truths have been newly created for this paper.

2.1 LMD Tempo

LMD is a dataset containing MIDI files that have been matched to 30s audio excerpts. While some of the MIDI files contain tempo information, none of the audio files are annotated, and there is no guarantee that associated MIDI and audio files have the same tempo. Our idea is to create a sub-dataset, called LMD Tempo, that can be used for training supervised tempo induction algorithms. To this

end, we estimated the tempo of the matched audio previews using the algorithm from [31]. Then the associated MIDI files were parsed for tempo change messages. If the value of more than half the tempo messages for a given preview were within 2% of the estimated tempo, we assumed the estimated tempo of the audio excerpts to be correct and added it to LMD Tempo. This resulted in 3,611 audio tracks. We were able to match more than 76% of the tracks to the Million Song Dataset (MSD) genre annotations from [29]. Of the matched tracks 29% were labeled rock, 27% pop, 5% r&b, 5% dance, 5% country, 4% latin, and 3% electronic. Less than 2% of the tracks were labeled jazz, soundtrack, world and others. Thus it is fair to characterize LMD Tempo as a good cross-section of popular music.

2.2 MTG Tempo

The MTG Key dataset was created by Faraldo [8] as a ground-truth for key estimation of electronic dance music (edm), a genre that is very much underrepresented in LMD Tempo. Each two-minute track in MTG Key is annotated with one or more keys and a confidence value $c \in \{0, 1, 2\}$ for the key annotation. We annotated those tracks that have an unambiguous key and a confidence of $c = 2$ with a manually tapped tempo, which makes it one of the very few datasets that is suitable for key *and* tempo estimation. The resulting dataset size is 1,159 tracks. In the following we will refer to this new ground-truth as MTG Tempo.

2.3 Extended Ballroom

The original Ballroom dataset [11] is still used as test dataset today, which is why we exclude it from training. Better suited is the recently released and much larger Extended Ballroom dataset. Because it contains some songs also occurring in Ballroom, we use the complement $\text{Extended Ballroom} \setminus \text{Ballroom}$. We refer to the resulting dataset as EBall. It contains 3,826 tracks with 30s length each. EBall contributes tracks from genres that are underrepresented or simply absent from both MTG Tempo and LMD Tempo.

2.4 Combined Training Dataset

Combined, LMD Tempo, MTG Tempo, and EBall have a size of 8,596 tracks with tempi ranging from 44 to 216 BPM (Figure 1). In the following we will call it Train. The *sweet octave* (i.e., the tempo interval $[\tau, 2\tau)$ that contains the most tracks [31]) for Train is 77 – 154 BPM, covering 84.4% of the items. The shortest interval that covers 99% of the items is 65 – 204 BPM. Even though many different tempi are represented, Train is not tempo-balanced. More than 30% of its tracks have tempi in the $[120, 130)$ interval. Its mean is $\mu = 121.32$ and the standard deviation $\sigma = 30.52$. And while covering many different genres, Train is not genre-balanced, either. Genres like jazz and world only have relatively few representatives. But despite these shortcomings,

¹ <https://github.com/GiantSteps/GiantSteps-mtg-key-dataset>

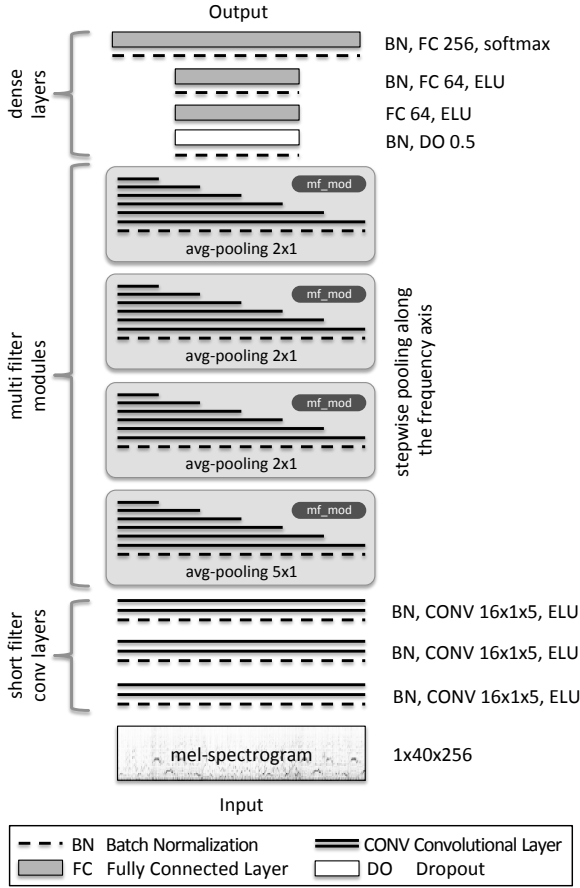


Figure 2: Schematic overview of the network architecture. Three convolutional layers are followed by four *mf_mod* modules, which in turn are followed by four dense layers.

Train is a very rich, multi-faceted dataset and completely independent from the test datasets we are going to use for evaluation in Section 4.1.

3. METHOD

Our proposed method for estimating a local tempo consists of a single step. Using a suitable representation we classify the signal with a CNN, which produces a BPM value. We extend the system for global tempo estimation by averaging the softmax activation function over different parts of a full track.

3.1 Signal Representation

Although we believe that it is possible to build a system like ours with raw audio as input [4, 19], we choose to represent the signal as mel-scaled magnitude spectrogram to reduce the amount of data that needs to be processed by the CNN. The mel-scale as opposed to a linear scale was chosen for its relation to human perception and instrument frequency ranges.

To create the spectrogram, we convert the signal to mono, downsample to 11,025 Hz and use half-overlapping windows of 1,024 samples. This is equivalent to a

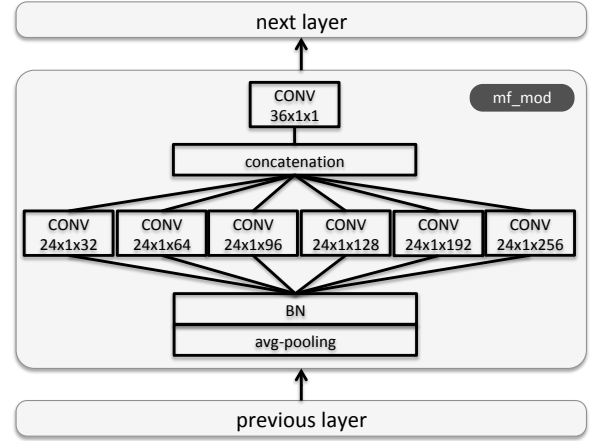


Figure 3: Each multi-filter module *mf_mod* consists of a pooling layer, batch normalization, six different convolutional layers, a concatenation layer and a bottleneck layer. The activation function for all convolutional layers is ELU.

frame rate of 21.5 Hz, which (according to the Nyquist-Shannon sampling theorem) suffices to represent tempi up to 646 BPM—well above the tempi we usually find in music. Each window is transformed into a 40 band mel-scaled magnitude spectrum covering 20 – 5,000 Hz by applying a Hamming window, the DFT, and a suitable filterbank. Since musical tempo is not an instantaneous quantity, we require a spectrogram of a musically sufficient length. As such we choose 256 frames, equivalent to ≈ 11.9 s.

3.2 Network Architecture

Even though tempo estimation appears to be a regression problem, we are approaching it as a classification problem for two reasons. First, a probability distribution over multiple classes allows us to judge how reliable a given estimate is. Additionally, such a distribution is naturally capable of representing tempo ambiguities [21], allowing for the estimation of a second best tempo. Second, in informal experiments we found that a classification-based approach led to more stable results compared to a regression-based approach. So instead of attempting to estimate a BPM value as decimal number, we are choosing one of 256 tempo classes, covering the integer tempo values from 30 to 285 BPM.

The proposed network architecture (Figure 2) is inspired by the traditional approach of first creating an OSS, which is then analyzed for periodicities. In our approach, we first process the input with three convolutional layers with 16 (1×5) filters each. All filters are oriented along the time axis using padding and a stride of 1. Using these fairly short filters, we hope to match onsets in the signal.

These three layers are followed by four almost identical multi-filter modules (*mf_mod*, Figure 3) each consisting of an average pooling layer ($m \times 1$), parallel convolutional layers with different filter lengths ranging from (1×32) to (1×256), a concatenation layer and a (1×1) bottleneck layer for dimensionality reduction. With each of these

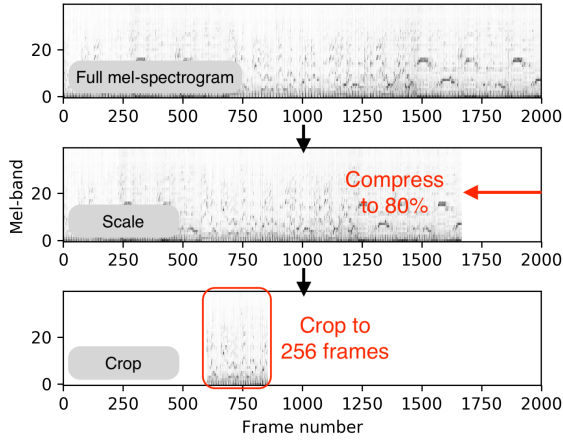


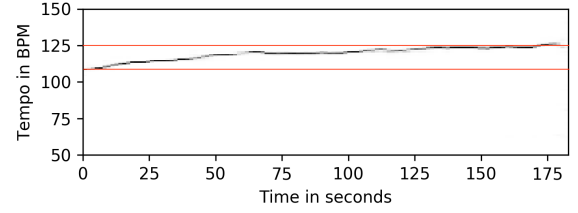
Figure 4: Scale-&-crop data augmentation. During training, the mel-spectrogram is first stretched or compressed along the time axis, which requires an adjustment of the ground-truth label, and then cropped to 256 frames at a randomly chosen offset.

modules we are trying to achieve two goals: 1) Pooling along the frequency axis to summarize mel-bands, and 2) matching the signal with a variety of filters that are capable of detecting long temporal dependencies. Using parallel convolutional layers with different filter lengths has been inspired by [26, 35]. In a traditional system, this could be regarded as some sort of comb filterbank

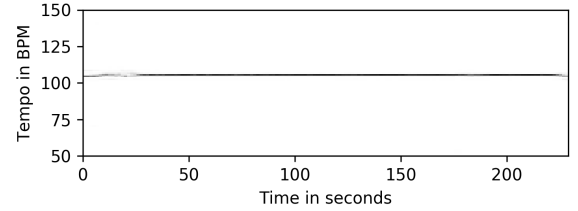
To classify the features delivered by the convolutional layers, we add two fully connected layers (64 units each) followed by an output layer with 256 units. The output layer uses softmax as activation function, while all other layers use ELU [3]. Each convolutional or fully connected layer is preceded by batch normalization [16]. The first fully connected layer is additionally preceded by a dropout layer with $p = 0.5$ to counter overfitting. As loss function we use categorical cross-entropy. Overall, the network has 2,921,042 trainable parameters.

3.3 Network Training

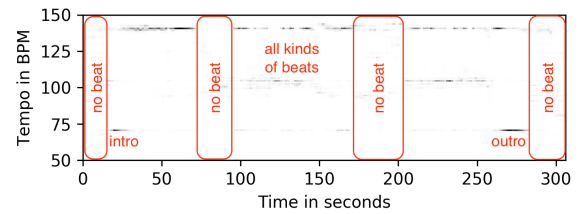
We use 90% of Train for training and 10% for validation. To counter the tempo class imbalance and, at the same time, augment the dataset during training, for each epoch, we use a scale-&-crop-approach borrowed from image recognition systems (see e.g., [34]). Contrary to regular images, the two dimensions of spectrograms have very different meaning, which is why we cannot simply scale-&-crop indiscriminately. Instead, we have to be careful to either not change the labeled meaning of a sample or change its label suitably (Figure 4). In our case this means that we have to preserve the properties of the frequency axis, but may manipulate the time axis. Concretely, we scale the time axis of the samples’ mel-spectrograms with a randomly chosen factor $\in \{0.8, 0.84, 0.88, \dots, 1.16, 1.2\}$ using spline interpolation and adjust the ground-truth tempo labels accordingly. This substantially increases the number



(a) “Honky Tonk Women” by The Rolling Stones



(b) “Rolling in the Deep” by Adele



(c) “Typhoon” by Foreign Beggars/Chasing Shadows

Figure 5: Tempo class probabilities for tracks from different genres and eras. (a) The tempo drift of the performance is clearly visible: the track starts with 108 BPM and ends with 125 BPM. (b) Very stable tempo of a modern pop music production. (c) Dubstep track with several no beat passages, a very active middle section, and half tempo intro and outro.

of different samples we can present to the network. Since the full mel-spectrogram for a sample is longer than the network input layer (e.g., covering 60 s vs. 11.9 s), we crop each scaled sample at a randomly chosen time axis offset to fit the input layer. This again drastically increases the number of different samples we can offer to the network. After scaling and cropping, the values of the resulting sub-spectrogram are rescaled to $[0, 1]$. In order to ensure comparability, time-axis augmentations are skipped during validation.

We define *Accuracy0* as the fraction of estimates that are correct when rounding decimal ground-truth labels to the nearest integer. To avoid overfitting, we train until *Accuracy0* for the validation set has not improved for 20 epochs using Adam (with a learning rate of 0.001, $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-8$) as optimizer, and then keep the model that achieved the highest validation *Accuracy0* (early stopping).

Dataset	schr	böck	new	Dataset	schr	böck	new	Dataset	schr	böck	new
ACM Mirum	38.3	29.4-	40.6	ACM Mirum	72.3-	74.0-	79.5	ACM Mirum	97.3	97.7	97.4
ISMIR04	37.7	27.2-	34.1	ISMIR04	63.4	55.0	60.6	ISMIR04	92.2	95.0	92.2
Ballroom	46.8-	33.8-	67.9	Ballroom	64.6-	84.0-	92.0	Ballroom	97.0	98.7	98.4
Hainsworth	43.7	33.8	43.2	Hainsworth	65.8-	80.6	77.0	Hainsworth	85.6	89.2+	84.2
GTzan	38.8	32.2-	36.9	GTzan	71.0	69.7	69.4	GTzan	93.3	95.0+	92.6
SMC	14.3	17.1	12.4	SMC	31.8	44.7+	33.6	SMC	55.3	67.3+	50.2
GiantSteps	53.5-	37.2-	59.8	GiantSteps	63.1-	58.9-	73.0	GiantSteps	88.7	86.4-	89.3
Combined	40.9-	31.2-	44.8	Combined	66.5-	69.5-	74.2	Combined	92.2	93.6+	92.1
DS Average	39.0	30.1	42.1	DS Average	61.7	66.7	69.3	DS Average	87.1	89.9	86.4

(a) *Accuracy0*(b) *Accuracy1*(c) *Accuracy2*

Table 1: Accuracies in percent. The ‘+’ and ‘-’ signs indicate a statistically significant difference between either *schr* or *böck*, and *new*. Bold numbers mark the best-performing algorithm(s) for a dataset. DS Average is the mean of the algorithms’ results for each dataset.

3.4 Global Tempo Estimation

Since the input layer is usually shorter than the mel-spectrogram of a whole track, it estimates merely a local tempo. To estimate the global tempo for a track, we calculate multiple output activations using a sliding window with half-overlap, i.e., a hop size of 128 frames ≈ 5.96 s. The activations are averaged class-wise and then—just like in the local approach—the tempo class with the greatest activation is picked as the result.

4. EVALUATION

For evaluation, we trained three models and chose the one with the highest *Accuracy0* measured against the validation set as our final model. As metrics we used *Accuracy0* as well as *Accuracy1* and *Accuracy2*, which are typically used for evaluating tempo estimation systems. *Accuracy1* is defined as the fraction of estimates identical to reference values while allowing a 4% tolerance. *Accuracy2* is the percentage of correct estimates allowing for octave errors 2 and 3 again using a 4% tolerance.

4.1 Global Tempo Benchmarking

It has become customary to benchmark tempo estimation methods with results reported for a small set of datasets: ACM Mirum [24], Ballroom [11], GTzan [36], Hainsworth [12], ISMIR04 [11], GiantSteps Tempo [18], and SMC [13]. The latter was specifically designed to be difficult for beat trackers. Where applicable, we used the corrected annotations from [25]. A detailed description of the datasets is given in [31]. We refer to the union of these seven datasets as *Combined*. Unweighted averages of results for all seven datasets will be referred to as *DS Average*. We benchmarked our approach *new* with the algorithms by Böck et al. (*böck*) [2]² and Schreiber (*schr*) [31]. Table 1 shows the results.

Overall, *new* achieves the highest results when tested against *Combined* with the strict metrics *Accuracy0* (44.8%) and *Accuracy1* (74.2%). Both accuracy values are slightly lower when summarized as *DS Average*.

² *madmom-0.15.1*, default options, available at <https://github.com/CPJKU/madmom>

When testing with octave-error tolerance, i.e., *Accuracy2*, *böck* reaches 93.6% for *Combined*, versus 92.2% reached by *schr*, and 92.1% reached by *new*. In essence, *new* is better than *böck* at estimating the tempo octave correctly, while *böck*—and to a lesser degree *schr*—achieve a slightly higher accuracy when ignoring the metrical level. This may be due to the fact that both *böck* and *schr* use a traditional periodicity analysis (DFT and comb filters, respectively) that tends to be prone to octave errors, while *new* does not use a comparable isolated component.

When inspecting the dataset-specific results, we find that *new*’s *Accuracy1* is particularly high for Ballroom (92.0%), GiantSteps (73.0%), and ACM Mirum (79.5%). In fact, they are significantly higher than *böck*’s (+8.0 pp/+14.1 pp/+5.5 pp) or *schr*’s (+27.4 pp/+9.9 pp/+7.2 pp) results. Both the Ballroom and GiantSteps values can be explained through our training dataset. They clearly correspond to EBall and MTG Tempo, therefore high values are not surprising. We believe the same is true for ACM Mirum and LMD Tempo. To us these results indicate that a genre-complete training set may lead to better results for the other datasets as well. This hypothesis is supported by the fact that GTzan contains genres like reggae, classical, blues, and jazz, and Hainsworth contains the genres choral, classical, folk, and jazz—none of which are well represented in Train. For both datasets *new* performs worse than *böck* or *schr*. A similar connection may exist for *böck* and GiantSteps—as far as we know, *böck* has not been trained on *edm*.

4.2 Local Tempo Visualization

To illustrate the system’s performance for continuous local tempo estimation, we analyzed several tracks from different genres using overlapping windows with a relatively small hop size of 32 frames, i.e., ≈ 1.5 seconds. For clarity, we cropped the images at 50 and 150 BPM. Figure 5a beautifully reveals the tempo drift in The Rolling Stone’s 1969 performance of “Honky Tonk Women”, starting out at 108 BPM and ending in 125 BPM. In contrast, Adele’s recent studio production “Rolling in the Deep” (Figure 5b) stays very stable at 105 BPM. A more complicated picture is presented by the dubstep track “Typhoon” by Foreign

Beggars/Chasing Shadows (Figure 5c). After several seconds of weather noises, the intro starts with 70 BPM. The main part's tempo is clearly 140 BPM interrupted by two sections with no beat. The outro again feels like 70 BPM followed by a fade out.

5. CONCLUSIONS

We have presented a single-step tempo estimation system consisting of a convolutional neural network (CNN). With a conventional mel-spectrogram as input, the system is capable of estimating the musical tempo using multi-class classification. The network's architecture consolidates traditional multi-step approaches into a single CNN, avoiding explicit mid-level features such as onset strength signals (OSS) or beat activation functions. Consequently and contrary to many other systems, our approach does not rely on handcrafted features or ad-hoc heuristics, but is completely data-driven. The system was trained with samples from the union of several large datasets, two of which were newly created. To aid training, we applied problem-specific data augmentation techniques. For global tempo estimation, we have shown that our single network, data-driven approach performs as well as or better than other more complicated state-of-the-art systems, especially w.r.t. *Accuracy1*. Furthermore, by visualizing examples for local tempo estimations, we have demonstrated qualitatively how the system can aid music analysis, e.g., to identify tempo drift.

We believe that the system can be improved even further by training with a more balanced dataset that contains tracks for all tested genres. Notably missing from the current training set are jazz, classical, or reggae tracks. Another area of potential improvement is the network architecture. Shorter filters, dilated convolutions, residual connections, and a suitable replacement for the fully connected layers might be used to reduce the number of parameters and thus the number of operations needed for training and estimation.

Additional Material

Datasets are available at http://www.tagtraum.com/tempo_estimation.html. Code to estimate tempi and create tempograms is available at <https://github.com/hendriks73/tempo-cnn>.

Acknowledgments

The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits IIS. Meinard Müller is supported by the German Research Foundation (DFG MU 2686/11-1).

6. REFERENCES

- [1] Miguel Alonso, Bertrand David, and Gaël Richard. Tempo and beat estimation of musical signals. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, 2004.
- [2] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 625–631, Málaga, Spain, 2015.
- [3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, Feb. 2015.
- [4] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 7014–7018, Florence, Italy, 2014. IEEE.
- [5] Simon Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30:39–58, 2001.
- [6] Anders Elowsson. Beat tracking with a cepstroid invariant neural network. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 351–357, New York, NY, USA, 2016.
- [7] Anders Elowsson and Anders Friberg. Modeling the perception of tempo. *The Journal of the Acoustical Society of America*, 137(6):3163–3177, 2015.
- [8] Ángel Faraldo, Sergi Jordà, and Perfecto Herrera. A multi-profile method for key estimation in EDM. In *Proceedings of the AES International Conference on Semantic Audio*, Erlangen, Germany, June 2017. Audio Engineering Society.
- [9] Aggelos Gkiokas, Vassilios Katsouros, and George Carayannis. Reducing tempo octave errors by periodicity vector coding and svm learning. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 301–306, Porto, Portugal, 2012.
- [10] Aggelos Gkiokas and Vassilis Katsouros. Convolutional neural networks for real-time beat tracking: A dancing robot application. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 286–293, Suzhou, China, October 2017.
- [11] Fabien Gouyon, Anssi P. Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, 2006.
- [12] Stephen Webley Hainsworth. *Techniques for the Automated Analysis of Musical Audio*. PhD thesis, University of Cambridge, UK, September 2004.

- [13] Andre Holzapfel, Matthew E.P. Davies, José R. Zapata, João Lobato Oliveira, and Fabien Gouyon. Selective sampling for beat tracking evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(9):2539–2548, 2012.
- [14] Florian Hörschläger, Richard Vogl, Sebastian Böck, and Peter Knees. Addressing tempo estimation octave errors in electronic music by incorporating style information extracted from wikipedia. In *Proceedings of the Sound and Music Computing Conference (SMC)*, Maynooth, Ireland, 2015.
- [15] Eric J. Humphrey, Juan Pablo Bello, and Yann LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 403–408, Porto, Portugal, 2012.
- [16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [17] Anssi P. Klapuri, Antti J. Eronen, and Jaakko Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):342–355, 2006.
- [18] Peter Knees, Ángel Faraldo, Perfecto Herrera, Richard Vogl, Sebastian Böck, Florian Hörschläger, and Michael Le Goff. Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 364–370, Málaga, Spain, October 2015.
- [19] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 220–226, Espoo, Finland, July 2017.
- [20] Ugo Marchand and Geoffroy Peeters. The extended ballroom dataset. In *Late Breaking Demo of the International Conference on Music Information Retrieval (ISMIR)*, New York, NY, USA, 2016.
- [21] Martin F. McKinney and Dirk Moelants. Deviations from the resonance theory of tempo induction. In *Proceedings of the Conference on Interdisciplinary Musicology*, Graz, Austria, 2004.
- [22] Meinard Müller. *Fundamentals of Music Processing – Audio, Analysis, Algorithms, Applications*. Springer Verlag, 2015.
- [23] Geoffroy Peeters. Template-based estimation of time-varying tempo. *EURASIP Journal on Advances in Signal Processing*, 2007(1):158–158, 2007.
- [24] Geoffroy Peeters and Joachim Flocon-Cholet. Perceptual tempo estimation using GMM-regression. In *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies (MIRUM)*, pages 45–50, New York, NY, USA, 2012. ACM.
- [25] Graham Percival and George Tzanetakis. Streamlined tempo estimation based on autocorrelation and cross-correlation with pulses. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(12):1765–1776, 2014.
- [26] Jordi Pons and Xavier Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2472–2476, New Orleans, USA, March 2017. IEEE.
- [27] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016.
- [28] Eric D. Scheirer. Tempo and beat analysis of acoustical musical signals. *Journal of the Acoustical Society of America*, 103(1):588–601, 1998.
- [29] Hendrik Schreiber. Improving genre annotations for the million song dataset. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 241–247, Málaga, Spain, 2015.
- [30] Hendrik Schreiber and Meinard Müller. Exploiting global features for tempo octave correction. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 639–643, Florence, Italy, 2014.
- [31] Hendrik Schreiber and Meinard Müller. A post-processing procedure for improving music tempo estimates using supervised learning. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 235–242, Suzhou, China, October 2017.
- [32] Björn Schuller, Florian Eyben, and Gerhard Rigoll. Tango or waltz?: Putting ballroom dance style into tempo detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2008:12, 2008.
- [33] Jarno Seppänen. Tatum grid analysis of musical signals. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 131–134, 2001.
- [34] Patrice Y. Simard, David Steinkraus, John C. Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of*

the 7th International Conference on Document Analysis and Recognition (ICDAR), volume 3, pages 958–962, August 2003.

- [35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, Boston, MA, USA, June 2015.
- [36] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.
- [37] Fu-Hai Frank Wu. Musical tempo octave error reducing based on the statistics of tempogram. In *23th Mediterranean Conference on Control and Automation (MED)*, pages 993–998, Torremolinos, Spain, 2015. IEEE.
- [38] Fu-Hai Frank Wu and Jyh-Shing Roger Jang. A supervised learning method for tempo estimation of musical audio. In *22nd Mediterranean Conference of Control and Automation (MED)*, pages 599–604, Palermo, Italy, 2014. IEEE.
- [39] Jose R. Zapata and Emilia Gómez. Comparative evaluation and combination of audio tempo estimation approaches. In *42nd AES Conference on Semantic Audio*, Ilmenau, Germany, 2011.

averaging the likelihoods produced by each CNN in the ensemble. Then, the authors turn the soft state assignments of the CNN ensemble into hard assignments (*downbeat* vs *no-downbeat*) using an HMM. This approach showed the potential of CNNs for downbeat tracking and the complementarity of the different musically inspired features.

In parallel, Böck et al. [4], presented a system that jointly tracks beats and downbeats using Bi-Directional Long-Short Term Memory networks (Bi-LSTMs). The authors used three different magnitude spectrograms and their first order differences as input representations, in order to help the networks capture features with sufficient resolution in both time and frequency. The input representations were fed into a cascade of three fully connected Bi-LSTMs, obtaining activation functions for beat and downbeat as output. Subsequently, a highly constrained DBN was used for inferring the metrical structure.

In turn, Krebs et al. [16] proposed a downbeat tracking system that uses two beat-synchronous features to represent the percussive and harmonic content of the audio signal. Those feature representations, based on spectral flux and chroma, are then fed into two independent Bidirectional Gated Recurrent Units (Bi-GRUs) [8]. Finally, the downbeat likelihood is obtained by merging the likelihoods produced by each Bi-GRU. The final inference for downbeat candidates relies on a constrained DBN.

More recently, combinations of CNNs and Recurrent Neural Networks (RNNs) such as GRUs or LSTMs have received increasing attention. For instance, Convolutional-Recurrent Neural Network architectures (CRNNs) have been proposed in other MIR tasks such as chord recognition [20] or drum transcription [23], and they are the state of the art in other audio processing domains such as sound event detection [2, 6].

1.2 Our contributions

In this paper we offer a systematic investigation of important system design choices, namely the impact of the input observations' temporal granularity, the output encoding, and the post-processing stage. Also, we investigate the potential of CRNNs for improving feature learning for the task of downbeat tracking. To perform our experiments, we modify a state-of-the-art RNN-system [16], and study the effect of the different envisaged variations, keeping the training setup and input features fixed. Our experimental results show that the post-processing stage improves the performance in all cases, whereas the addition of a dense-structured output encoding does not help in the training of downbeat tracking systems. The proposed CRNN architecture performs competitively with the state-of-the-art RNN system, being even able to improve the reference system's performance with a proper choice of input's temporal grid. We also observe that though beat tracking errors tend to propagate to the output decisions, the CRNN system is able to recover from these errors better than the baseline RNN when taking the input observations over a tatum grid (as opposed to beat grid).

2. ANALYSIS OF COMMON VARIATIONS

In this section we briefly describe the baseline system, the motivation of each studied variation (or design choice) and the experiments related to it. In particular, we study the effects and interactions of 4 design choices: the input's temporal granularity, the output encoding, the effect of post-processing and the network architecture.

2.1 Recurrent neural network baseline

To perform our analysis, we implemented the state-of-the-art downbeat tracking system presented by Krebs et al. [16]. The architecture of this system consists of two concatenated Bi-GRUs of 25 units each, where each hidden state vector $h(t)$ at time t is mapped by a dense layer to a state prediction $p(t)$ using a sigmoid activation. A dropout layer is used in training to avoid over-fitting. Two separate networks are trained using different input features and the obtained likelihoods are averaged. The low-level input representations comprise two beat-synchronous feature sets, representing the *harmonic* and *percussive* content of the audio signal. The set of features describing percussive content, which we will refer to as PCF (Percussive Content Feature), is based on a multi-band spectral flux, computed using the short time Fourier transform with a Hann window, using a hop-size of 10ms and a window length of 2048 samples, with a sampling rate of 44100 Hz. The obtained spectrogram is filtered with a logarithmic filter bank with 6 bands per octave, covering the range from 30 to 17 000 Hz. The harmonic content's representation is the CLP (Chroma-Log-Pitch) [21] with a frame rate of 100 frames per second. The temporal resolution of the features is 4 subdivisions of the beat for the PCF, and 2 subdivisions for the CLP features. For computational efficiency, the authors in [16] assembled in matrices column-wise this resolution increment so the CLP feature set is of dimension 12×2 and the PCF is 45×4 , which we maintained in this work. The beats for the beat-synchronous feature mapping are obtained using the beat tracker presented in [3], with the DBN introduced in [17].¹

In our experiments, we have observed that including batch normalization (BN) layers [14] consistently improves performance. We included two BN layers, one after the input layer, and the other between the Bi-GRUs.

The optimization of the model parameters is carried out by minimizing the binary-cross-entropy between the estimated and reference values.

2.2 Temporal granularity: beat vs tatums

The temporal granularity of the input observations (or temporal grid) relates to important aspects of the design of downbeat tracking systems. It determines the length of the context taken into account around musical events, which controls design decisions in the network architecture, such as filter sizes in a CNN, or the length of training sequences in an RNN.

¹ In particular we used the DBNBeatTracker algorithm of the madmom package version 0.16 [5].

Among the different downbeat systems, several granularities have been used. In particular, the latest state-of-the-art systems use either musically motivated temporal grids (such as tatums or beats) or fixed length frames. Systems that use beat- or tatum-synchronous input depend on reliable beat/tatum estimation upstream, so they are inherently more complex, and prone to error propagation [11, 16]. On the other hand, frame-based systems are not subject to these problems, but the input dimensionality is much higher due to the increased observation rate [4], which causes difficulties when training the models.

In this paper, we focus on musically motivated temporal analysis grids, because they reduce the computational complexity of the systems considerably. We study the variations in performance using beat and tatum grids.

We compute the tatums by interpolating the beats, with a resolution of 4 tatums per beat interval.² To study the impact of the temporal grid, we train the networks keeping the input features, architecture, training data, and post-processing fixed, while changing only the inputs' temporal granularity. We adapt the sequence length used for training the networks in order to consider the same musical context in all cases (as specified further below). We compare the interaction of the choice of temporal grid with those of the output encoding, the RNN or CRNN architectures and the post-processing stage.

2.3 Output encoding: structured vs unstructured

Among the downbeat tracking systems mentioned in Section 1.1, the common choice is to use an one-hot vector encoding to indicate the presence or absence of a downbeat at a particular position of the excerpt at training time. For instance, if using temporal analysis grid that is aligned on beats, a sequence of beats is usually encoded as $s = [1, 0, 0, 0, 1, 0, 0, 0]$, indicating the presence of a downbeat at the first and 5th beat positions. We refer to this as *unstructured* encoding. Here, we also investigate whether a densely *structured encoding* may help the neural networks perform a better downbeat tracking.

2.3.1 Structured encoding definition

We define the structured encoding as a set of classes that are active within the entire inter-beat interval. This is the set $\mathcal{C} = \{1, \dots, 13\}$, where each class indicates the position of the beat inside a bar. We consider a maximum bar length of 12 beats, and an extra class X for labeling an observation in the absence of beats and downbeats, for a total of 13 classes ($K = 13$). For instance, to label a musical piece with time signature 4/4, we use the subset of labels $\{1, 2, 3, 4\}$, and we label consecutive time units corresponding to the same beat interval with the same class. Figure 1 illustrates the difference between the proposed and the unstructured encoding. In this structured class lexicon, the downbeats are represented by the label 1.

We train the networks incorporating both the unstructured and the structured encoding. In this configuration,



Figure 1. Audio excerpt in 4/4 labeled with the *structured* encoding (top figure) and the *unstructured* encoding (bottom figure). The temporal granularity showed is tatum (beat quarter-notes). In the structured encoding each tatum receives a label corresponding to its metrical position.

we use one dense layer to decode each class lexicon, and we evaluate the performance of the system using the unstructured output. The dense layers are connected so the information of the beginning of the bar is provided by the unstructured dense to the structured one as an extra feature. We test the effect of the encoding on the different temporal granularities. It is important to note that the unstructured coding has a clear interaction with the temporal grid in terms of the number of 1- vs 0- symbols in the training data, while the structured coding is consistent (i.e., the amount of class instances remains proportional) under any temporal granularity.

2.4 Post-processing: DBN vs thresholding

The importance of the post-processing stage has been addressed in previous works [11, 16]. In this paper, we assess the relative importance of this stage depending on the temporal granularity and the network architecture. To that end, we use the DBN presented in [16]. This DBN models beats (or tatums) as states, forcing the state sequence to always transverse the bar from left to right (i.e., transitions from beat 2 to beat 1 are not allowed), and imposing that time signature changes are unlikely. We consider bar lengths of 3 and 4 beats (12 and 16 tatums). We invite the interested reader to refer to [16] for further information.

2.5 Architecture: RNNs vs CRNNs

We base our CRNN architecture design on previous state-of-the-art choices. Particularly, our CNN design is based on the best CNN of the ensemble in [11], which we combine with a single Bi-GRU layer [8]. The bi-directional version of GRUs integrates the information across both temporal directions, providing temporal smoothing. CNNs are capable of extracting high level features that are invariant to both spectral and temporal dimensions, whether RNNs model longer term dependencies accurately.

The architecture that we propose can be seen as an encoder-decoder system [7], where the encoder maps the input to an intermediate time series representation that is then mapped to the output space by the decoder. An interesting advantage of this kind of scheme is that several combinations of encoder-decoder can be explored easily as long as they share the intermediate representation.

²This estimation is on the 16th note level, which we assumed as a good compromise to perform downbeat tracking.

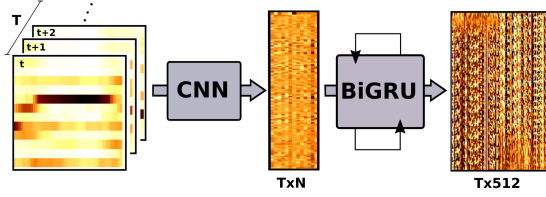


Figure 2. Encoder architecture: the input representation is either a beat/tatum-synchronous chromagram or multi-band spectral flux. Each time unit is fed into the encoder with a context window. The CNN outputs a sequence of dimension $T \times N$ which is fed to the Bi-GRU. Finally, the encoder output dimension is $T \times 512$.

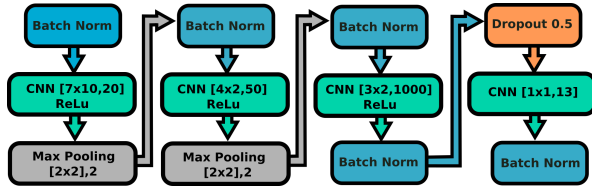


Figure 3. Summary of the CNN architecture.

2.5.1 Encoder architecture

The encoder architecture is depicted in Figure 2. It consists of a convolutional-recurrent network. Each temporal unit (either beat or tatum) is fed into the CNN considering a fixed-length context window of approximately one bar (following [11]). The CNN processes each window independently, and outputs a sequence of size $T \times N$ (T being the length of the input sequence and N the output dimension of the CNN) that is fed to the Bi-GRU. In this scheme, the CNN processes the signal locally whereas the recurrent network provides temporal consistency.

The CNN architecture is based on the harmonic-CNN presented in [11]. This network consists of a cascade of convolutional and max-pooling layers, with dropout used during training to avoid over-fitting, to a total of eight layers. We add batch normalization layers to avoid too large or small values within the network that could hurt the encoder. Additionally, we modify the filters' size to adapt to the feature shapes described in Section 2.1. Figure 3 shows the filter parameters in the case of the tatum grid.

The last layer of the CNN differs from the reference implementation in the number of units, which we set to 13 instead of 2 to fit features of bigger dimension to the Bi-GRU. We also remove the softmax activation of the last layer because the class discrimination is not carried out by the CNN. A summary of the CNN architecture is presented in Figure 3. Figure 3 represents the CNN's 2D filter sizes and the number of units, which is $[m \times n, u]$, with m and n operating in the spectral and temporal dimensions respectively, and u the number of units. The activation (if used) is indicated before the CNN description. Max pooling layers are notated as $[m' \times n', s]$, s indicating frequency and time dimension and stride. The interested reader is referred to [11] for the motivation of network architecture.

The local features computed by the CNN are fed into a Bi-GRU, which consists of two independent GRUs, one running in each temporal direction. Their hidden state vectors are concatenated to produce the bi-directional hidden state vector. We set the dimensionality of each GRU to 256, resulting in a total of 512.

2.5.2 Decoder architecture

Our decoder architecture is a fully connected dense layer that maps each hidden state vector to the prediction state using a sigmoid activation, resulting in a downbeat likelihood at each time unit. The optimization of the model parameters is carried out by minimizing the binary-cross-entropy among the estimated and actual values.

3. EXPERIMENTS

3.1 Experimental setup

Model implementation: The models were implemented with Keras 2.0.6 and TensorFlow 1.2.0 [1, 9]. We use the ADAM optimizer [15] with default parameters. We stop training after 10 epochs without changes on the validation set, up to a maximum of 100 epochs. The low-level representations were extracted using the madmom library and mapped to either the beat or tatum grid (see Section 2.1).

Model variations: We study the following variations:

- Temporal granularity:* using low-level features synchronized to two temporal granularities (tatum and beat);
- Output encoding:* with and without the addition of the structured encoding during training;
- Post-processing:* using either a threshold or a DBN;
- Architecture:* we test RNNs and CRNNs.

This results in sixteen different configurations, which we will refer to as R or C to indicate the architecture (RNN vs CRNN); S or U to indicate the encoding (structured vs unstructured); B or T to indicate temporal granularity (beat vs tatum); and t and d to indicate the post-processing method (threshold vs DBN). All models are trained using patches of 15 beats or 60 tatums depending on the temporal grid used. We use mini-batches of 64 patches per batch and a total of 100 batches per epoch.

Datasets: We investigate the performance of these configurations on 8 datasets of Western music, in particular: *Klapuri* which consists of 4h 54m of various genres songs. *R. Williams* which consists of 4h 31m of Pop songs. *Rock* which consists of 12h 53m of Pop and Rock songs. *RWC Pop* which consists of 6h 47m of Pop music. *Beatles* which consists of 8h 01m of Beatles songs. *Ballroom* which consists of 6h 04m of Ballroom dances. *Hainsworth* which contains 3h 19m of various genres. *RWC Jazz*, which consists of 3h 44min of Jazz music.

Evaluation methods: We perform leave-one-dataset-out evaluation and report the F-measure scores as in [11, 16]. 25% of the training data is used for validation. The

RWC Jazz dataset is only used to illustrate the performance of the systems in a challenging scenario where the beat estimation is less accurate and the music genre differs considerably from the training data, it is not used for training.³ Candidates for downbeats are obtained in two different manners. The first one is by thresholding the output activations with a threshold chosen to give the best F-measure result on the validation set. The second manner is to post-process the networks' outputs by adapting the DBN used in [16]. In this way we report the gain of using the DBN in each case. We use the DBN to model time signatures 3/4 and 4/4 following [16], and modifying it accordingly to the temporal grid (i.e., allowing bar lengths of {3,4} beats or {12, 16} tatum).

Methods are evaluated independently on each dataset listed above for comparison to prior work. We also include an evaluation over the union of all datasets (denoted *ALL*). To determine statistically significant differences, we conduct a Friedman test on the *ALL*-set results, followed by post-hoc Conover tests for pairwise differences using Bonferroni-Holm correction for multiple testing [13].

All configurations are trained with the same input low-level representations, the same musical context, the same training parameters and post-processing method. This allows us to draw conclusions about the performance of the models in different conditions and to compare the architectures modularly.

3.2 Results and discussion

We use as baseline two state-of-the-art downbeat tracking systems [11, 16], which reported 78% and 78.6% mean F-measure across all datasets.⁴ The performance of the models presented here across datasets is better than the baselines for all the cases when using the DBN as post-processing stage. The better results are obtained with RUBd (reference implementation, see Section 2.1) and CUTd up to 82.4% and 82.8% respectively. A possible explanation for this improvement is the difference in the beat tracking performance, which is 3.3% better than the one reported in [16]. This is likely to explain the 4.7% improvement in the RUBd model which is our reference implementation. To make a fair comparison, we use the RUBd model as a baseline, with the reasonable assumption that it behaves as the state-of-the-art. Figure 4 illustrates the performance of the different model variations across datasets. A detailed analysis is presented in the following.

The Friedman test on the *ALL* set rejected the null hypothesis ($p < 1e-10$). Post-hoc analysis determined that all pair-wise comparisons were significantly different ($p < 1e-3$), with the following exceptions: RUTt/RSTt, RUTd/RSTd, CUTt/CSBt, CUTd/CSBd, CUBd/CSTd, and RSBd/CUBd.

Effect of post-processing: As shown in Figure 4, d vs t model variants, the DBN post-processing helps in all

cases, being particularly important with the tatum granularity and with the RNN models. The gap in performance between the models with and without post-processing is notable in the case of the Ballroom database, where in some cases is up to 10% F-measure. The DBN increases the performance from RUTt to RUTd by 6.6% in the case of tatum grid across all databases, and 4.1% in the case of the beat grid (RUBt to RUBd). A similar trend is observed with the structured models (RS). The increase in the CRNN models performance is smaller, being 3.8% from CUTt to CUTd and 2.7% for CUBt to CUBd. The results obtained with the thresholding (t models) are more consistent over temporal granularities for the CRNN models, which suggests that the likelihood estimation of that model is more accurate and consistent over time.

Effect of the temporal granularity: The temporal grid has an important effect on the performance of the RNN models, as illustrated in Figure 4 (T vs B variations). When using a tatum grid, the thresholding results (e.g. RUTt) are lower in most of the cases, showing that the RNN variations have more difficulty to model the temporal dependencies in that grid. The post-processing stage with the DBN becomes more important in the case of the tatum grid, helping the RNN models up to an extra 2.5% in mean F-measure over all datasets compared to the beat grid case.

By contrast, the CRNN models appear to be robust to the temporal granularity change. In particular, for the case of the thresholding results, the performance of the models is similar for the beat and tatum granularities (e.g. CUBt vs CUTt), which implies the estimated likelihoods perform comparably. The increase in resolution seems to help the CRNN models in most cases, showing a small increase from beat to tatum grid with the DBN (e.g., CUBd vs CUTd). This indicates that the CRNN architecture is likely being able to take advantage of a finer temporal grid.

The impact of the temporal granularity in the RNN and CRNN models are in line with the decisions of the authors in [11, 16], who in the first case decided to use tatums (with CNNs), and in the second case decided to use beats (with Bi-GRUs).

Effect of the structural encoding: Regarding the structural encoding, the experiments show that it has no impact on the performance across databases (e.g. RU vs RS and CU vs CS in Figure 4). We observed some examples where the performance decreases, and we noticed two systematic problems: first, in some cases the estimated likelihoods become sharper and more structured when using the encoding, and when the estimation of the networks is not accurate the likelihoods consistently maintain the bad estimation across the whole duration of the audio signal. This indicates that the encoding is structuring the likelihood estimation, but that is not desirable in some cases, especially if it prevents the post-processing stages from compensating for these errors. Second, we observed several cases where the attack of the downbeat is not accurately estimated with the addition of the structured encoding. A possible expla-

³ We kept RWC Jazz out of the training set to be comparable to [16].

⁴ For datasets that are not evaluated in [11], we report results in [16].

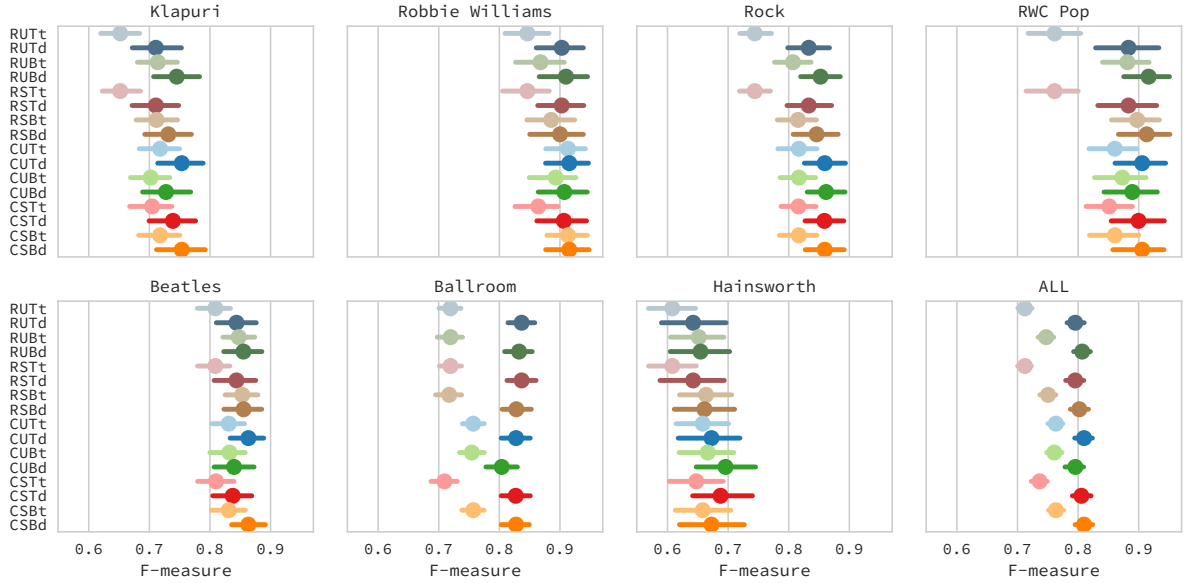


Figure 4. For each dataset, the estimated mean F-measure for each model under comparison. Error bars correspond to 95% confidence intervals under bootstrap sampling ($n = 1000$). *ALL* corresponds to the union of all test collections.

nation is the lack of information about the onset of the no-downbeat intervals in the structured encoding, which could be preventing the system to correctly model beats and downbeats internally. This could change in a scenario with joint beat and downbeat tracking, where the sparse encoding also contains the information of the location of beats. The addition of data augmentation could also contribute to help the system to learn the encoding properly.

CRNN vs RNN — difficult scenario: Finally, to see the performance of the systems in a difficult scenario, we performed an experiment on the RWC Jazz dataset, whose results are given in Figure 5. The DBN post-processing is used in all cases. The CRNN models are more robust to unseen data, since the jazz genre is different from the genres of the training data. The CRNN models have better performance and less dispersion in the results. Analogously to Figure 4, the RNN models show slightly better mean performance in beat grid and the CRNN models in tatum grid.

4. CONCLUSIONS AND FUTURE WORK

In this work we presented a systematic study of common decisions in the design of downbeat tracking systems based on deep neural networks. We explored the impact of temporal granularity, output encoding, and post-processing stage in two different architectures. The first architecture is a state-of-the-art RNN, and the second is a CRNN introduced in this paper. Experimental results show that the choice of the inputs’ temporal granularity has a significant impact on performance, and that the best configuration depends on the architecture. The post-processing stage improves performance in all cases, with less impact in the case of the CRNN models whose likelihood esti-

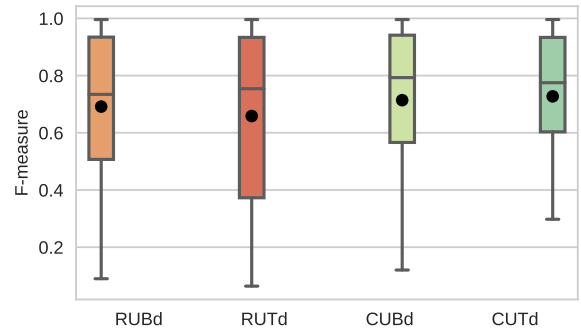


Figure 5. F-measure scores for the RWC Jazz dataset. Boxes show median value and quartiles, whiskers the rest of the distribution. Black dots denote mean values. All results are obtained using the DBN post-processing.

mations are most accurate. We conclude that the addition of a densely structured output encoding does not help in the training of downbeat tracking systems. Nevertheless, the interaction of the structured encoding with multi-task training (beat and downbeat tracking) and data augmentation are interesting perspectives for future studies, and will be addressed in future work. The proposed CRNN architecture performs as the state-of-the-art, proving robustness in a challenging scenario.

Acknowledgments. The authors would like to thank Simon Durand and Florian Krebs for sharing the code of their downbeat tracking architectures with us. B.M. is supported by the Moore-Sloan Data Science Environment at NYU.

5. REFERENCES

- [1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
- [2] S. Adavanne, P. Pertil, and T. Virtanen. Sound event detection using spatial features and convolutional recurrent neural network. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.
- [3] S. Böck, F. Krebs, and G. Widmer. A multi-model approach to beat tracking considering heterogeneous music styles. In *15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.
- [4] S. Böck, F. Krebs, and G. Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.
- [5] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new python audio and music signal processing library. In *Proceedings of the 24th ACM International Conference on Multimedia (ACMMM)*, 2016.
- [6] E. Çakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303, June 2017.
- [7] K. Cho, A. Courville, and Y. Bengio. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886, Nov 2015.
- [8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [9] F. Chollet. Keras. <https://github.com/fchollet/keras>, 2015.
- [10] S. Durand, J. P. Bello, B. David, and G. Richard. Downbeat tracking with multiple features and deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.
- [11] S. Durand, J. P. Bello, B. David, and G. Richard. Robust downbeat tracking using an ensemble of convolutional networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):76–89, Jan 2017.
- [12] S. Durand and S. Essid. Downbeat detection with conditional random fields and deep learned features. In *17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.
- [13] S. Garcia and F. Herrera. An extension on “statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, Dec 2008.
- [14] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *32nd International Conference on Machine Learning (ICML)*, 2015.
- [15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [16] F. Krebs, S. Böck, M. Dorfer, and G. Widmer. Downbeat tracking using beat synchronous features with recurrent neural networks. In *17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.
- [17] F. Krebs, S. Böck, and G. Widmer. An efficient state space model for joint tempo and meter tracking. In *16th International Society for Music Information Retrieval Conference (ISMIR)*, 2011.
- [18] N. C. Maddage. Automatic structure detection for popular music. *IEEE MultiMedia*, 13(1):65–77, Jan 2006.
- [19] M. Mauch and S. Dixon. Simultaneous estimation of chords and musical context from audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1280–1289, aug 2010.
- [20] B. McFee and J.P. Bello. Structured training for large-vocabulary chord recognition. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- [21] M. Müller and S. Ewert. Chroma toolbox: Matlab implementations for extracting variants of chroma-based audio features. In *12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011.
- [22] J. Paulus and A. Klapuri. Measuring the similarity of rhythmic patterns. In Michael Fingerhut, editor, *Proc. of the Third International Conference on Music Information Retrieval (ISMIR)*, 2002.
- [23] R. Vogl, M. Dorfer, G. Widmer, and P. Knees. Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

METER DETECTION AND ALIGNMENT OF MIDI PERFORMANCE

Andrew McLeod

University of Edinburgh

A.McLeod-5@sms.ed.ac.uk

Mark Steedman

University of Edinburgh

steedman@inf.ed.ac.uk

ABSTRACT

Metrical alignment is an integral part of any complete automatic music transcription (AMT) system. In this paper, we present an HMM for both detecting the metrical structure of given live performance MIDI data, and aligning that structure with the underlying notes. The model takes as input only a list of the notes present in a performance, and labels bars, beats, and sub beats in time. We also present an incremental algorithm which can perform inference on the model efficiently using a modified Viterbi search. We propose a new metric designed for the task, and using it, we show that our model achieves state-of-the-art performance on a corpus of metronomically aligned MIDI data, as well as a second corpus of live performance MIDI data. The code for the model described in this paper is available at <https://www.github.com/apmcleod/met-align>.

1. INTRODUCTION

Meter detection is the organisation of the beats of a given musical performance into a sequence of trees at the bar level, in which each node represents a single note value (although the actual durations of a node at a given level will vary with the tempo). In common-practice Western music (the subject of our work), the children of each node in the tree divide its duration into some number of equal-value notes such that every node at a given depth has equal value. The metrical structure of a single $\frac{4}{4}$ bar, down to the quaver level, is shown in Figure 1. Each level of a metrical tree corresponds with a pulse level in the underlying music: bar, beat, and sub beat, from top to bottom. The nodes should align in time with corresponding pulses in the performed music. There are theoretically more divisions further down the tree all the way to the tatum level (the fastest pulse present in a piece of music, often the 32nd note), but as these three levels are enough to unambiguously identify the time signature of a piece, we do not consider any lower.

The task is an integral component of automatic music transcription (AMT), particularly when trying to identify the time signature of a given performance. The time signature may change between bars (though this is not particularly common). However, such changes in structure

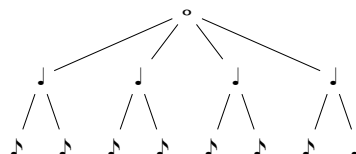


Figure 1. The metrical structure of a $\frac{4}{4}$ bar.

are not currently handled by our model, and are left for future work. The proposed model can be applied to any piece where the metrical tree structure under each node at a given level of the tree is identical. In this work, we evaluate our model only on the simple and compound meter types $\frac{2}{X}$, $\frac{3}{X}$, $\frac{4}{X}$, $\frac{6}{X}$, $\frac{9}{X}$, and $\frac{12}{X}$ (where X may take any value), and leave more uncommon and irregular meters for future work. Those interested in asymmetric meter detection should refer to [9].

Existing work on full metrical alignment of live performance MIDI data is sparse. There is a good amount of existing work on meter detection (but not alignment) from metronomic data (e.g., [2, 14]), including some which labels the meter type (i.e., duple or compound) of a given piece of music, but does not align a full metrical structure with the notes of the piece (except for synthetic rhythms, as in [8]). There is existing work which performs full metrical alignment of MIDI data, but not from live performance [4]. In the acoustic domain, beat tracking and downbeat detection are relatively common areas of research, but stop short of a full meter detection and alignment (e.g. [1, 7]).

The related problems of rhythm quantisation and note value detection have also seen some attention, but neither are directly relevant to our task. For example, [17] quantises performed rhythms to a grid, but the set of possible onset locations for notes is known a priori (and changes based on the time signature of the underlying piece). [3] tracks beats and tempo, but does not go so far as to align a full metrical grid with bars and sub beats. [15] assigns a note value to each note, but does not explicitly align the notes with any underlying beat or meter.

[22] performs full metrical structure detection and alignment probabilistically from live performance data by jointly modelling tempo, meter, and rhythm; however, the evaluation was very brief, only testing the model on 3 bars of a single Beatles piano performance, and the idea was not used further on MIDI data to our knowledge. [19] proposes a Bayesian model for the meter detection and alignment of monophonic MIDI performance data which mod-



© Andrew McLeod, Mark Steedman. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: Andrew McLeod, Mark Steedman. "Meter Detection and Alignment of MIDI Performance", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

els the probability of a note onset occurring given the current level of the metrical tree at any time with Bayes' rule. This is combined with a simple Bayesian model of tempo changes, giving a model which can detect the full metrical structure of a performance. [20] extends this model to work on polyphonic data, combining it into a joint model with a Bayesian voice separator and a Bayesian model of harmony. This joint model performs well on full metrical structure detection and alignment on a corpus of piano excerpts, and we compare against it in this work.

2. PROPOSED MODEL

Our proposed model tracks pulses at the tatum level of a musical performance based on two musicological principles: (1) the rate of these tatums should be relatively constant without large discontinuities; and (2) notes should lie relatively close to these tatums. The model is an HMM where the observed data is the notes of a given piece, grouped into sets.

2.1 State Space

Each state S in our model represents a single bar, containing (1) a list of the tatums from that bar and (2) a metrical hierarchy, describing which of those tatums are beats and sub beats. The list of tatums is represented by $S.t$, where $S.t_i$ is the i th tatum in the bar, and $S.t_{|S.t|}$ is the downbeat of the following bar. The tatums are in increasing time order, where $\mathcal{T}(S.t_i)$ represents the time of tatum $S.t_i$. A state's metrical hierarchy has some number of tatums per sub beat, sub beats per beat, and beats per bar, as well as an anacrusis length, measured by the number of tatums which fall before the first downbeat of a given piece. In our model, we restrict the number of tatums per sub beat to be 4, although in theory, any number could be used. We also restrict the anacrusis length to be some integer multiple of the number of tatums per sub beat, a simplifying assumption that ensures the first note of each piece will fall on a sub beat. The set of possible sub beat per beat and beat per bar pairs (i.e., time signatures) are taken all of those found in our training set ($\frac{2}{4}$, $\frac{3}{4}$, $\frac{4}{4}$, $\frac{6}{8}$, $\frac{9}{8}$, and $\frac{12}{8}$). A state's tempo, $T(S)$, is defined as the average length of its beats.

Each possible initial state S_0 contains no tatums, and every possible metrical hierarchy is considered equally probable. To reduce our model's search space, we place a restriction on the range of allowed values for $T(S_1)$: $t_{min} \leq T(S_1) \leq t_{max}$. Nonetheless, because the possible tatum times for each state are unbounded, our model contains an infinite number of possible states. Thus, instead of predefined emission and transition probabilities, we define emission and transition functions, presented in the following sections.

2.2 Emission Function

After the initial state (which emits nothing), each state S_i emits a set of notes N_i , containing only notes n whose onset times lie between that state's first (inclusive) and last (exclusive) tatum. This set is allowed to be empty.

Each emitted note has an onset time $\text{On}(n)$, an offset time $\text{Off}(n)$, and a pitch $\text{Pitch}(n)$ (though it is unused).

The probability of a state S_i to emit the note set N_i is presented as $P(N_i|S_i)$ in Eqn (1). The first term, calculated entirely by the lexicalised probabilistic context-free grammar (LPCFG) presented in [13], is used to prefer generating rhythms which have a high probability according to the grammar. The LPCFG is a replacement grammar which first parses a given rhythm into a metrical tree structure. It then assigns strengths to nodes in the tree based on note duration in a process called lexicalisation. The probability of a tree is calculated by taking the product of the learned probabilities of each grammar transition, based on counting occurrences of a given transition from a training corpus of parsed rhythms. Each note is aligned to the nearest tatum by the LPCFG in order to calculate $P(\text{rhythm})$, but this alignment is neither saved nor emitted. The LPCFG is designed to work directly on monophonic melodies only. Therefore, for polyphonic input, this $P(\text{rhythm})$ term is in fact a product of one probability per voice, each of which is calculated by the LPCFG. For voice assignments, we use [12] as a preprocessing step.

$$P(N_i|S_i) = P(\text{rhythm}) \prod_{n \in N} P(n|S_i.t) \quad (1)$$

The second term in Eqn (1) is used to prefer states whose tatums align closely with the emitted notes, and is calculated as in Eqn (2), where $\mathcal{N}_1(\mu, \sigma, x)$ conceptually represents a normal distribution with mean μ and standard deviation σ evaluated at x .¹ Thus, $P(n|S_i.t)$ is used to assign higher probabilities to those states which emit notes which are closely-aligned with their tatums. In this equation, $\text{closest}(S_i.t)$ represents the tatum from S_i whose time is closest to the onset time of the note n .

$$P(n|S_i.t) = \mathcal{N}_1(0, \sigma_n, \text{On}(n) - \mathcal{T}(\text{closest}(S_i.t))) \quad (2)$$

2.3 Transition Function

A state S_{i-1} may transition to a state S_i if and only if: (1) the two states' metrical hierarchies are identical (our model cannot handle pieces with time signature changes) and (2) the time of the last tatum in S_{i-1} is equal to the time of the first tatum in S_i . Note that the second condition is invalid in the case of a transition from S_0 to S_1 since S_0 contains no tatums; in this case, we instead restrict $S_1.t_1$ to lie exactly on the first observed note's onset time.

The transition probability $P(S_i|S_{i-1})$ is shown in Eqn (3), where the first term, defined in Eqn (4), models the probability of a tempo change and the second term, defined in Eqn (5), models the spacing of the tatums themselves.

$$P(S_i|S_{i-1}) = P(T(S_i)|T(S_{i-1}))P(S.t) \quad (3)$$

¹ Normal distributions are used in multiple places throughout this model with potentially widely varying standard deviations, resulting in potentially wildly different results when evaluated at an identical number of standard deviations from the mean for different normal distributions. Since the distributions are used in contexts in which they cannot be properly normalised (due to their continuous domain), the precise probability value for $\mathcal{N}_1(\mu, \sigma, x)$ is calculated using a standard normal distribution with mean 0 and standard deviation 1 evaluated at $\frac{x-\mu}{\sigma}$.

$$P(T(S_i)|T(S_{i-1})) = \begin{cases} \mathcal{N}_1(\mu_{t_0}, \sigma_{t_0}, T(S_i)) & i = 1 \\ \mathcal{N}_1(0, \sigma_t, \frac{T(S_i) - T(S_{i-1})}{T(S_{i-1})}) & i \geq 2 \end{cases} \quad (4)$$

$$P(S.t) = E(b \in S.t) \prod_{b \in S.t} (E(sb \in b) \prod_{sb \in b} E(t \in sb)) \quad (5)$$

In Eqn (4), the tempo of the first bar (where $i = 1$) is assumed to be normally distributed around μ_{t_0} with standard deviation σ_{t_0} , while subsequent tempo changes are evaluated as the proportional change from the tempo of the previous bar, again normally distributed, this time with mean 0 and standard deviation σ_t . Percent change is used rather than absolute change because human perception of tempo changes have been shown to follow Weber's Law [21].

For the tatum timings in Eqn (5), the function $E(t)$, defined in Eqn (6), evaluates the probability of the evenness of any given list of times. $E(b \in S.t)$ calculates this for all of the beats b in the state, while the terms $E(sb \in b)$ and $E(t \in sb)$ perform the same calculation for the sub beats in each beat and the tatums in each sub beat respectively.

$$E(t) = \begin{cases} \mathcal{N}_1(\mu_e, \sigma_e, \frac{\sigma(t)}{\mu(t)}) / E_{norm} & \frac{\sigma(t)}{\mu(t)} \geq \mu_e \\ \mathcal{N}_1(\mu_e, \sigma_e, \mu_e) / E_{norm} & \frac{\sigma(t)}{\mu(t)} < \mu_e \end{cases} \quad (6)$$

$$E_{norm} = \frac{1}{2} + \frac{\mu_e}{\sigma_e} \mathcal{N}_1(\mu_e, \sigma_e, \mu_e) \quad (7)$$

$E(t)$ is a piecewise function which takes as input a list of the lengths of a group of tatums, sub beats, or beats (rather than their times). Here, $\mu(t)$ represents the mean of those lengths and $\sigma(t)$ represents the standard deviation of those lengths. The function is calculated as a modified normal distribution with mean μ_e and standard deviation σ_e , based on the input list's standard deviation as a proportion of its mean. If this proportion is greater than or equal to μ_e , the result is calculated from a straightforward normal distribution. Otherwise, the result is exactly the value of a standard normal distribution evaluated at its mean.

This value is then normalised so as to ensure the new distribution's integral to again sum to 1 by dividing by the factor E_{norm} , defined in Eqn (7) as the sum of two terms. $\frac{1}{2}$ is the area of the standard normal distribution greater than the mean, and $\frac{\mu_e}{\sigma_e} \mathcal{N}_1(\mu_e, \sigma_e, \mu_e)$ is the area of the rectangle formed by extending the peak of the standard normal distribution to the left until the value corresponding to 0 from the non-standardised normal distribution, as values less than this correspond to a negative $\sigma(t)$, which is not possible.

2.4 Search Space Reduction

We use a modified Viterbi search to perform inference on our model, using a beam search where at each step we save only the B most probable hypothesis states (not including those still at S_0 with no tatums yet).

For the transition from S_0 to S_1 , we introduce two heuristics: (1) the first tatum in S_1 must lie exactly on the onset of the first observed note and (2) the last tatum in S_1 must also lie exactly on a note onset, though which note specifically is not restricted by any means other than

limiting the tempo of the first bar using t_{min} and t_{max} . According to these heuristics, for each S_0 , the supervisor creates the observed note set for every possible S_1 . Allowed times for the tatums in $S_1.t$ are also restricted based on each observed note set N_1 . Essentially, all tatums are placed evenly unless there is a specific reason not to (i.e., unless a note onset lies close to a tatum).

Specifically, a given value for $S_1.t$ is legal if it can ever be generated by the following procedure. First, the appropriate number of beats (according to a given state's metrical hierarchy) are placed between the first and last tatum times, as if each tatum was evenly spaced. Next, each placed beat—excluding the last beat as well as the first—may be shifted to the location of any note whose onset time is within half of one sub beat length of the original beat location. Each beat (again excluding the first and last as appropriate) may then be nudged up to half of a tatum length around its location with a magnetism of M_b , as shown in Eqn (8). Here, t is the original time of the beat, M is the magnetism (M_b in this case) which is used to control how far the beat is nudged, and N is the set of notes which lie within the given window. This equation can always return the original time, though it is also allowed to nudge the given time towards either the onset time of the closest note ($closest(N)$) or the average onset time of all notes within the window ($avg(N)$), if N is large enough. Sub beats are placed similarly: initially evenly between any of the existing beats, then nudged up to one tatum length around its location with magnetism M_{sb} . Notice that the sub beats are not shifted. Finally, tatums are placed evenly between the sub beats (and neither shifted nor nudged).

$$nudge(t, M, N) = \begin{cases} t & \text{always} \\ t + M(closest(N) - t) & |N| > 0 \\ t + M(avg(N) - t) & |N| > 1 \end{cases} \quad (8)$$

Allowed times for the tatums in $S_i.t$ for $i > 1$ are restricted to those which can be generated by the same procedure, with the exception that the final beat in $S_i.t$ may now be shifted and nudged. Initial beat locations are calculated such that $T(S_{i-1}) = T(S_i)$.

Intuitively, this process of shifting and nudging allows a hypothesis' tempo to smoothly increase or decrease based on the observed notes. Beats are allowed to change the tempo more drastically than sub beats because they are more salient, and more likely to align with note onsets.

Even with the above restrictions, the search space is still large. As mentioned we use a beam search, where at each step we save only the top B most probable hypothesis states (not including those still at S_0 with no tatums yet). Before we remove those hypotheses which fall outside the beam, we remove hypotheses which are deemed to be too close to another more probable hypothesis based on a threshold Δ_{min} . Specifically, a hypothesis which has identical metrical hierarchy to a more probable hypothesis, and whose tempo and most recent tatum time both lie within Δ_{min} of that other hypothesis' tempo and most recent tatum time is removed.

2.5 Supervisor

It is important to note that due to the way in which the observed note sets are grouped by bar, the individual note sets for different paths through the HMM state space for a given piece will not be identical, although the union of all note sets on any given path equals exactly the set of notes present in the piece. To handle this complication, we introduce a *supervisor* during the HMM decoding process which takes each note individually in onset order, grouping them into note sets and passing the sets to the appropriate hypothesis state at each step. Specifically, for a given hypothesis state, the supervisor determines the longest and shortest possible lengths for the following bar (based on allowed shifts and nudges as described in the previous section). Then, it creates every possible note set given those bounds, and allows the hypothesis state to transition and branch on each of those note sets.

2.6 Optimisations

Here we describe two changes used to make our model more robust in regards to the idiosyncrasies of live performance such as staccato and ornamentation.

For handling staccato notes which are much shorter than their note values would suggest in the score, we extend each note's offset until either the onset of the following note in the bar within its voice (if one exists), or to the end of its bar. This allows the LPCFG, which is trained on metronomic MIDI where staccato is not present, to better recognise the rhythms present in live performance.

For handling ornamentation such as trills, we use a threshold $trill_{max}$. Any note whose onset time is within $trill_{max}$ of the onset time of the previous note within its voice is removed (though the removed notes are still used when deciding whether to remove the subsequent note). The overall effect of this process is that trills or any very fast ornamentation (which again would not be present in the LPCFG's training data) are reduced to a single short note with its onset at the start of the trill or ornamentation. If this optimisation is used in conjunction with the extend notes optimisation, the remaining notes are extended only after trills and ornamentation are removed, and the result is that a fast ornamentation is replaced by a single long note.

3. EVALUATION

3.1 Corpora

For evaluation, we use two corpora: one containing metronomic MIDI files of the 48 fugues from Bach's Well-Tempered Clavier (WTC)² and Bach's 15 Inventions,³ and another of 13 live performance MIDI files of Bach's fugues and preludes from the WTC, from Crest-MusePEDB⁴ [10]. For training, we also use the miscellaneous corpus, released and used by [20] for training, divided into a live performance portion (containing 22 pieces

by various composers recorded from a MIDI keyboard) and a metronomic portion (containing 45 non-performed pieces by various composers). For voice assignments in all corpora, we run [12] as a preprocessing step.

3.2 Training

To train most of the parameters for the beat tracking model, we measure statistics from the live performance portion of the miscellaneous corpus. This results in values of $\mu_{t_0} = 1.0885$ s, $\sigma_{t_0} = 709.918$ ms, $\sigma_t = 0.0743$, $\mu_e = 0.0181$, $\sigma_e = 0.0336$, $\sigma_n = 6.655$ ms, $t_{min} = 0.4$ s, and $t_{max} = 3$ s.

The remaining parameters are set in an ad hoc fashion through testing on the miscellaneous corpus, and we have found our model's performance not to be very sensitive to the precise values used. Specifically, we use $M_b = 1.0$, $M_{sb} = 0.5$, and $trill_{max} = 0.1$ s. Δ_{min} and B are simply optimisations used to improve the speed of our model, and we use values of 1 ms and 200 respectively, though in practice, lower values of Δ_{min} or higher values of B only improve our model's performance.

For our standard evaluation, we train the LPCFG's probabilities from the metronomic portion of the miscellaneous corpus, since this allows for a direct comparison with the model of [20]. However, it is noted in [13] that the grammar is sensitive to a lack of training data, particularly a lack of training data in the style of the evaluation corpus, which happens when training on the miscellaneous corpus for evaluation on Bach compositions. To investigate this further, we also run experiments when training the LPCFG's probabilities on a superset containing the metronomic portion of the miscellaneous corpus as well as the entire metronomic corpus of Bach compositions. Note that when evaluating this version of our model, we leave out the piece currently being evaluated from the grammar's training set so as to avoid overfitting. In all experiments, we train the LPCFG with data that has undergone the same optimisations as the data to be evaluated (in terms of extending notes and removing trills and ornamentation).

3.3 Metric

Quantitative evaluation of previous work on meter alignment, particularly with MIDI data, is uncommon, and a few possible metrics are discussed in [18]. [20] reports five values which take into account tempo, phase, and the branching factor at each level of the metrical tree. Work on acoustic meter detection (e.g. [11]) often reports F-measures of beats and downbeats, treated as points in time.

To evaluate our model's performance, we would rather use a metric similar to that from [13] which is a single value, takes into account the tree structure's groupings (rather than just its beat locations), and has some idea of the partial correctness of a metrical alignment. However, as it is designed for use mainly on beat-aligned data where a metrical hypothesis cannot move in and out of phase throughout a piece, a few adjustments must be made to adapt it for use on live performance data. We call our newly designed evaluation metric the metrical F-measure.

² The fugues were acquired from www.musedata.org.

³ The inventions were acquired from www.imslp.org.

⁴ We do not include the 13th prelude from WTC Book I due to an error in the file.

Method	Metronomic	Live Performance
Temperley [20]	67.65	47.62
This Work	78.71	39.63
+T	75.36	39.40
+X	79.89	45.27
+X +T	77.67	47.81
+Bach	80.48	38.21
+Bach +T	80.08	42.35
+Bach +X	80.50	55.43
+Bach +X +T	80.48	56.51

Table 1. The average metrical F-measure of our method compared against those of [20] on our two corpora. +T indicates use of the remove trills and ornamentation optimisation, +X indicates use of the extend notes optimisation, and +Bach indicates using the additional Bach training data for the LPCFG.

It takes into account every grouping at three levels of the metrical hierarchy throughout an entire piece: the sub beat level, the beat level, and the bar level.

For each hypothesised grouping at these metrical levels, we check if it matches a ground truth grouping. A hypothesised grouping is said to match a ground truth grouping if its beginning and ending times are each within 70 *ms* of the beginning and ending times of that particular ground truth grouping, regardless of the metrical level of either grouping.⁵ Each matched pair of groupings within a piece count as a true positive, while any unmatched hypothesis groupings count as false positives, and any unmatched ground truth groupings count as false negatives. The metrical F-measure of a piece is then calculated as the harmonic mean of precision and recall as usual, and our reported results average these metrical F-measures across all songs in each corpus.

3.4 Results

We compare our model against that of Temperley [20], which is trained entirely on the miscellaneous corpus. For direct comparison, the standard version of our model is trained on the same corpus, but we present an evaluation of a few different versions of it based on different optimisations or training data. Results can be found in Table 1, where +T indicates use of the remove trills and ornamentation optimisation, +X indicates use of the extend notes optimisation, and +Bach indicates that the LPCFG training was augmented with the additional Bach compositions. We do not also augment Temperley’s model with additional training data because there is no straightforward way to do so, and the model does not seem to be one which would be as sensitive to a lack of training data as our model.

The results show that on metronomic data, our model without optimisations clearly outperforms Temperley’s when using identical training data. The optimisations offer no significant improvement (which is unsurprising as they were designed specifically to help with live performance),

⁵ This 70 *ms* window is taken directly from a popular beat tracking metric [6].

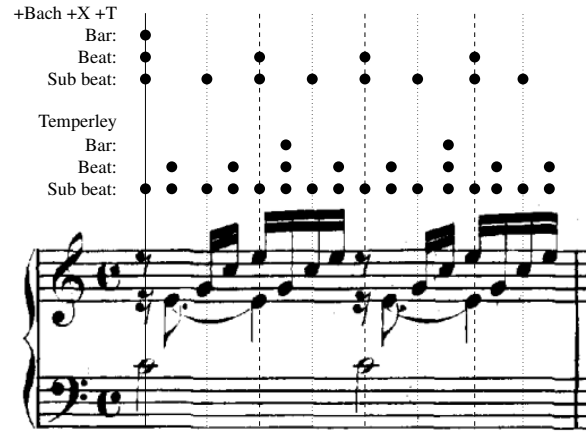


Figure 2. The first bar of the 1st prelude from WTC Book I (BWV 846). Above the music, the results from Temperley’s model (bottom) are shown as well as the results from our +Bach +X +T model (top).

but augmented training data leads to a small but consistent increase in performance across all optimisation configurations. On live performance, our model without optimisations underperforms Temperley’s, both with and without augmented training data. However, the optimisations lead to increased performance: our model using both optimisations matches Temperley’s performance with identical training data, and exceeds it by almost 9 points with augmented training data. The effect of each optimisation is discussed in detail in Section 3.4.1.

The distribution of metrical F-measures for Temperley’s model, run on live performance data, appears to be binomial: of the 13 pieces, three score below 20, while six score above 55, indicating that while the model performs well in general, it sometimes guesses a meter which is nearly entirely incorrect. With both optimisations, on the other hand, our model’s scores are normally distributed around 65, with 8 pieces scoring between 55 and 75. Additionally, no pieces score below 20, indicating that it is more likely to make some partially correct guess, even if it is not entirely correct. The 1st prelude from WTC Book I illustrates this difference in performance, and its first bar is shown in Figure 2 along with the results of Temperley’s model and our +Bach +X +T model. The piece is in $\frac{4}{4}$ time, and Temperley’s model achieves a score of only 15.74, guessing a $\frac{3}{8}$ time whose beats are even out of phase with the ground truth sub beats throughout much of the piece. On the other hand, our model scores 93.27, guessing a $\frac{4}{4}$ time which begins perfectly aligned, although it does shift slightly out of phase later in the piece.

One example of a piece for which Temperley’s model outperforms ours is the 2nd prelude of WTC Book II, the first bar of which is shown in Figure 3 along with the results of Temperley’s model and our +Bach +X +T model. For this piece, Temperley’s model achieves a score of 78.99 while ours only manages a score of 61.83. This piece is in $\frac{4}{4}$ time and contains relatively non-syncopated

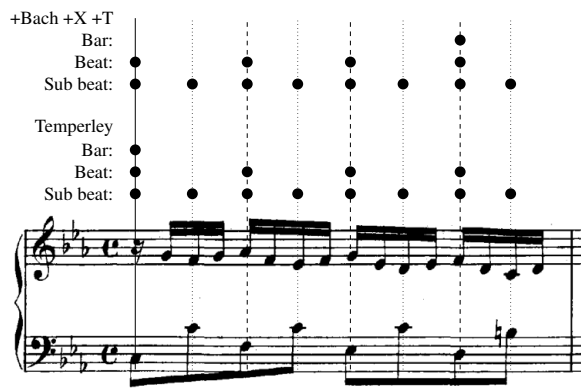


Figure 3. The first bar of the 2nd prelude from WTC Book II (BWV 871), showing an example the nearly isochronous bars which give our model problems. Above the music, the results from Temperley’s model (bottom) are shown as well as the results from our +Bach +X +T model (top).

rhythms, with many bars containing either only sixteenth notes or only eighth notes in a given voice, as can be seen in the figure. While Temperley’s model captures this meter correctly (with some phase errors), our model guesses a $\frac{4}{4}$ time which is early by a single beat. Our model has some difficulty finding the correct phase of isochronous melodies since it uses no pitch or harmonic information (which are the most salient indicators of metrical phase in such isochronous pieces). Temperley’s model, on the other hand, also includes chord detection, allowing it to better handle such melodies.

3.4.1 Optimisations

Another aspect of our model to investigate is the effect of the different optimisations on its performance. As can be seen from Table 1, they (+X and +T) have little effect on metronomic data (which is not surprising given that they are designed specifically for live performance). However, on live performance data, they improve performance significantly. Both with and without augmented training data, the remove trills optimisation has a small effect by itself (essentially none without the data and very small with it), but extending notes leads to a significant improvement. The combination of both optimisations improves performance even further, leading to peak performance both with and without augmented training data.

One specific example where the remove trills optimisation leads to improvement with augmented training data is on the 7th fugue from WTC Book I, where our +Bach and +Bach +T models achieve scores of 31.58 and 60.20 respectively. There is a repeated trill throughout this piece, leading the +Bach model to lengthen its beat length such that the trill is interpreted as 16th notes. With the remove trills optimisation, however, our model is able to find the correct metrical structure. Essentially, the remove trills optimisation frees our model from the constraint of trying to align its tatum with each note in a trill or ornamentation.

An example of a piece for which the extend notes opti-

misation makes an improvement is the 17th prelude from WTC Book I. In this piece, in $\frac{3}{4}$ time, the lowest voice has a very common repeated rhythm of an eighth note followed by two sixteenth notes followed by four more eighth notes, where the eighth notes are all played staccato. With the optimisation, our model correctly recognises the beat and sub beat levels, although it incorrectly guesses $\frac{2}{4}$ time rather than the correct $\frac{3}{4}$ time, scoring 53.59. Without the optimisation, on the other hand, these eighth notes are not as salient, and the model instead guesses a $\frac{2}{2}$ meter which moves in and out of phase throughout the piece, achieving a score of only 16.47. Throughout the corpus, the extend notes optimisation helps find strong notes whenever they are played staccato.

The combination of both optimisations improves overall performance even further, enabling the model to handle both staccato passages and ornamentation. The improvements from both optimisations are seen in the fully optimised model, alongside other slight improvements throughout the corpus such as fixing the placement of a single misaligned beat here or there. For example, in the previously discussed 17th prelude from WTC Book I, the fully optimised model achieves a metrical F-measure of 60.35 while no other model eclipses a score of 54, even though the basic metrical alignment (a $\frac{2}{4}$ meter) does not change between the it and the +Bach +X model.

4. CONCLUSION

In this paper, we have described a model for metrical structure detection and alignment from live performance MIDI.

Our model is in the form of an HMM which performs metrical structure detection and alignment given only a list of note pitches and onset and offset times, and we have shown that the model achieves state-of-the-art performance when evaluated on a corpus of metronomic data, as well as a second corpus of live performance data. The HMM incorporates a rhythmic grammar as one component, working with the grammar to align an input piece with a metrical structure. This joint model is probabilistic and incremental, and requires no information a priori except for note onset and offset times. We have also proposed a new metric for the task, which takes into account vertical misalignments (for example, those which align the beat level of a piece with bars) and partial correctness.

In future work, we would like to extend the evaluation of our model with more data. In particular, our corpus of 13 pieces of live performance MIDI would benefit from an expansion, and allow us to perform a more in-depth analysis of the results.

Metrical structure detection and alignment is clearly an important task for any complete transcription system, and we have shown that our joint model is able to perform the task well, even using only rhythmic data. Incorporating additional information such as pitch or harmony should only lead to better performance. Specifically, it has been shown that harmonic changes are most likely to occur at the beginnings of bars [16], and low notes may be a salient feature of strong beats in addition to note duration [5].

5. ACKNOWLEDGEMENTS

This work was partially funded by EU ERC H2020 Advanced Fellowship GA 742137 SEMANTAX and a Google faculty award.

6. REFERENCES

- [1] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *ISMIR*, pages 255–261, 2016.
- [2] Judith C. Brown. Determination of the meter of musical scores by autocorrelation. *The Journal of the Acoustical Society of America*, 94(4):1953, 1993.
- [3] A. T. Cemgil and B. Kappen. Monte carlo methods for tempo tracking and rhythm quantization. *Journal of Artificial Intelligence Research*, 18:45–81, January 2003.
- [4] W. Bas De Haas and Anja Volk. Meter detection in symbolic music using inner metric analysis. In *ISMIR*, pages 441–447, 2016.
- [5] Simon Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58, March 2001.
- [6] Simon Dixon. Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, 36(1):39–50, March 2007.
- [7] Simon Durand, Juan P. Bello, Bertrand David, and Gael Richard. Robust downbeat tracking using an ensemble of convolutional networks. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 25(1), 2017.
- [8] Douglas Eck and Norman Casagrande. Finding meter in music using an autocorrelation phase matrix and shannon entropy. In *ISMIR*, pages 504–509, 2005.
- [9] Thanos Fouloulis, Aggelos Pikrakis, and Emiliou Cambouropoulos. Traditional asymmetric rhythms: A refined model of meter induction based on asymmetric meter templates. In *Proceedings of the Third International Workshop on Folk Music Analysis*, pages 28–32, 2013.
- [10] Mitsuyo Hashida, Toshie Matsui, and Haruhiro Katayose. A new music database describing deviation information of performance expressions. *ISMIR*, pages 489–494, 2008.
- [11] Florian Krebs, Andre Holzapfel, A. Taylan Cemgil, and Gerhard Widmer. Inferring metrical structure in music using particle filters. 23(5):817–827, 2015.
- [12] Andrew McLeod and Mark Steedman. HMM-based voice separation of MIDI performance. *Journal of New Music Research*, 45(1):17–26, January 2016.
- [13] Andrew McLeod and Mark Steedman. Meter detection in symbolic music using a lexicalized PCFG. In *Proceedings of the 14th Sound and Music Computing Conference*, pages 373–379, 2017.
- [14] Benoit Meudic. Automatic meter extraction from MIDI files. In *Journées d’informatique musicale*, 2002.
- [15] Eita Nakamura, Kazuyoshi Yoshii, and Shigeki Sagayama. Rhythm Transcription of Polyphonic Piano Music Based on Merged-Output HMM for Multiple Voices. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(4):794–806, April 2017.
- [16] Hélène Papadopoulos and Geoffroy Peeters. Joint estimation of chords and downbeats from an audio signal. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(1):138–152, January 2011.
- [17] Christopher Raphael. Automated rhythm transcription. In *ISMIR*, 2001.
- [18] David Temperley. An evaluation system for metrical models. *Computer Music Journal*, 28(3):28–44, September 2004.
- [19] David Temperley. *Music and Probability*. The MIT Press, 2007.
- [20] David Temperley. A unified probabilistic model for polyphonic music analysis. *Journal of New Music Research*, 38(1):3–18, March 2009.
- [21] Kim Thomas. Just noticeable difference and tempo change. *Journal of Scientific Psychology*, pages 14–20, 2007.
- [22] Nick Whiteley, A. Taylan Cemgil, and Simon Godsill. Bayesian modelling of temporal structure in musical audio. In *ISMIR*, 2006.

A TIMBRE-BASED APPROACH TO ESTIMATE KEY VELOCITY FROM POLYPHONIC PIANO RECORDINGS

Dasaem Jeong, Taegyun Kwon, Juhan Nam

Graduate School of Culture Technology, KAIST, Korea

{jdasam, ilcobo2, juhannam} @kaist.ac.kr

ABSTRACT

Estimating the key velocity of each note from polyphonic piano music is a highly challenging task. Previous work addressed the problem by estimating note intensity using a polyphonic note model. However, they are limited because the note intensity is vulnerable to various factors in a recording environment. In this paper, we propose a novel method to estimate the key velocity focusing on timbre change which is another cue associated with the key velocity. To this end, we separate individual notes of polyphonic piano music using non-negative matrix factorization (NMF) and feed them into a neural network that is trained to discriminate the timbre change according to the key velocity. Combining the note intensity from the separated notes with the statistics of the neural network prediction, the proposed method estimates the key velocity in the dimension of MIDI note velocity. The evaluation on Saarland Music Data and the MAPS dataset shows promising results in terms of robustness to changes in the recording environment.

1. INTRODUCTION

Polyphonic piano transcription is one of the most active research topics in automatic music transcription [1]. However, the absolute majority of piano transcription algorithms so far have been concerned with detecting the presence of notes in term of pitch (or note number), onset and duration, while ignoring note dynamics, which is expressed by key velocity on piano.

Along with tempo, dynamics is a key feature that produces a musical “motion” [19]. Previous studies on piano performance analysis employed dynamics as one of two main features of performance characteristics in [22, 25]. Another study showed that, if dynamics is estimated for individual notes, a finer analysis is achievable [21].

There have been a few works that challenged the task of estimating individual note dynamics. To best of our knowledge, the first attempt was made by Ewert and Müller who tackled the problem using a parametric model of

polyphonic piano notes [7]. Our previous work estimated the note intensity using score-informed non-negative matrix factorization (NMF) in various training strategies [15]. Szeto and Wong used a sinusoidal model to separate chords tones into individual piano tones and estimated the note intensity as part of the source separation task [23].

All of them basically estimate individual note dynamics according to energy magnitude or loudness of the notes. However, this approach has an essential limitation in that a note produced by a certain key velocity can be recorded in different sound levels depending on the recording conditions. For example, a *pianissimo* note can be recorded loudly or a *forte* note can be quietly, depending on the input gain of the recording device or the distance from the microphone.

In this paper, we challenged to overcome this limitation by focusing on differences in timbral characteristics caused by the key velocity. According to previous research, loudness and tone of a piano note are uniquely determined by the velocity of the hammer at the time it strikes the strings [12]. This implies that the key velocity can be inferred not only from the loudness but also from the timbre of the note, assuming that the hammer velocity can be approximated by the key velocity. This idea was explored in [14] where a piano note shows different timbral characteristics such as a spectral envelope or inharmonicity, depending on the key velocity. While the previous work focused on single notes, we study it for polyphonic music.

The proposed system consists of three parts: an NMF module for note separation and intensity estimation, a neural network to discriminate key velocity, and intensity-to-velocity calibration using the results from the two modules. The NMF module is based on score-informed settings from [15] and [24]. After the decomposition of the audio spectrogram, we reproduce the note-separated spectrogram from the NMF module. The neural network takes the note-separated spectrogram as input and estimates its key velocity. The third part obtains proper mapping parameters between note intensity and key velocity using the distribution of velocity estimation from the neural network, and finally estimate individual key velocity in the dimension of MIDI note velocity. We evaluate the proposed method on Saarland Music Data and the MAPS dataset and show promising results in terms of robustness to changes in the recording environment.

The rest of paper is structured as follows. In Section 2 we introduce the scope of our work and define the terms



© Dasaem Jeong, Taegyun Kwon, Juhan Nam. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Dasaem Jeong, Taegyun Kwon, Juhan Nam. “A Timbre-based Approach to Estimate Key Velocity from Polyphonic Piano Recordings”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

that represent dynamics of a piano note. Section 3 summarizes the related works. In Section 4 we explain the NMF and neural network framework. The experiment and result are explained in Section 5 and 6. Finally, the conclusion is presented in Section 7.

2. BACKGROUND

To provide better understanding of the task and scope in this research, we first review key terms and define the problem that we attempt to solve.

2.1 Term Definitions

Note intensity is the term that represents the magnitude of acoustical energy of a note. It can be defined as sound-pressure level (SPL) [10] or the sum of spectral energy as in [7, 15]. Since the intensity is an acoustical feature, it is highly variable by the recording condition. For example, note intensity can be changed by simple post-processing such as gain adjustment. Therefore, the intensity of each note is comparable only when the recording conditions are consistent.

Key velocity refers to the kinetic velocity of the piano key and it is closely connected to the hammer velocity. It can be measured by detecting the elapsed time when the hammer shank passes two fixed points [10]. Unlike the note intensity, the key velocity is a feature measured directly from the mechanical movement, hence independent from the acoustic recording environment. If the recording condition is constant and the sympathetic resonance is ignored, the mapping between key velocity and note intensity for each pitch is linear [10].

MIDI velocity is the term that represents the key velocity in the MIDI format. It is a one-byte integer value between 0 and 127 inclusive in the note messages. Computer-controlled pianos or MIDI-compatible keyboards have their own mapping of key velocity to MIDI velocity.

2.2 Problem Definition

The aim of this study lies in estimating note key velocity in terms of MIDI velocity. Although our previous work attempted to produce the result in MIDI velocity, the method requires an additional data for intensity-to-velocity calibration with the same piano and recording condition [15]. In a real-world situation, however, it is almost impossible to obtain such mapping for a target recording. Instead of employing a target-suited training set, our work aims to learn a proper intensity-to-velocity mapping directly from a target audio recording.

One of the obstacles in the task is that most datasets represent the key velocity with MIDI velocity and the mapping between the two varies depending on the piano or keyboard model. To focus on the relation between timbre and key velocity in this study, we fix the key-to-MIDI velocity mapping by employing only one piano model but different recording conditions during the evaluation. However, we evaluate the trained model on recordings with a dif-

ferent piano to see how it generalizes. The details will be explained in the evaluation section.

3. RELATED WORKS

Our proposed method is based on the NMF framework from [15] but expand it by employing a recent work by Wang *et al* [24]. One of the main limitations in the NMF framework is that it is difficult to model the timbre changes over time. For example, the NMF model used in [8] and [15] assumes the spectral template of each pitch does not change over time. To overcome this limitation, Wang *et al* suggested using multiple spectral templates per pitch in NMF for piano modeling. This NMF model was adopted in our proposed system and will be discussed in more detail in the next section.

Identifying key velocity by its timbre can be compared to identification of musical instruments. The earlier works used various hand-crafted audio features [6, 14]. Recently, deep neural network has become a popular solution for this task [2, 11], which takes spectrograms or mel-frequency cepstral coefficients as input. There are a few work interested in timbral difference by the velocity [4, 14] but they did not aim to distinguish these difference explicitly.

Our task can also be compared to instrument identification in polyphonic audio. One of typical solutions for this task is using source separation and then handling it as monophonic audio sources. Heittola *et al.* suggested a framework with NMF-based source separation module [13]. Similar to this work, our method also employs NMF-based source separation. But we use the neural networks instead of the Gaussian mixture model to identify the separated sources.

4. METHOD

Our proposed system consists of three parts as shown in the Figure 1. The first part is score-informed NMF that factorizes the spectrogram of audio recording into note-separated spectrogram for every note in the score. This also returns the intensity of each note. The second part is neural network (NN) that takes the note-separated spectrogram and estimates the key velocity. The third part is intensity-to-velocity calibration which is conducted by comparing the estimated velocity from the NN module and the intensity from the NMF module on their distributions.

4.1 Note Separation

The first part of our framework is based on NMF, a matrix factorization for non-negative data which is usually spectrogram in audio processing domain.

Let us denote a given spectrogram as $\mathbf{V} \in \mathbb{R}_{\geq 0}^{F \times T}$, where F is the number of frequency bins and T is the number of time frames. With NMF, the spectrogram can be factorized into multiplication of two matrices $\mathbf{W} \in \mathbb{R}_{\geq 0}^{F \times (P \cdot R)}$ and $\mathbf{H} \in \mathbb{R}_{\geq 0}^{(P \cdot R) \times T}$ where P denotes the number of pitch in semitone and R denotes the number of spectral basis per

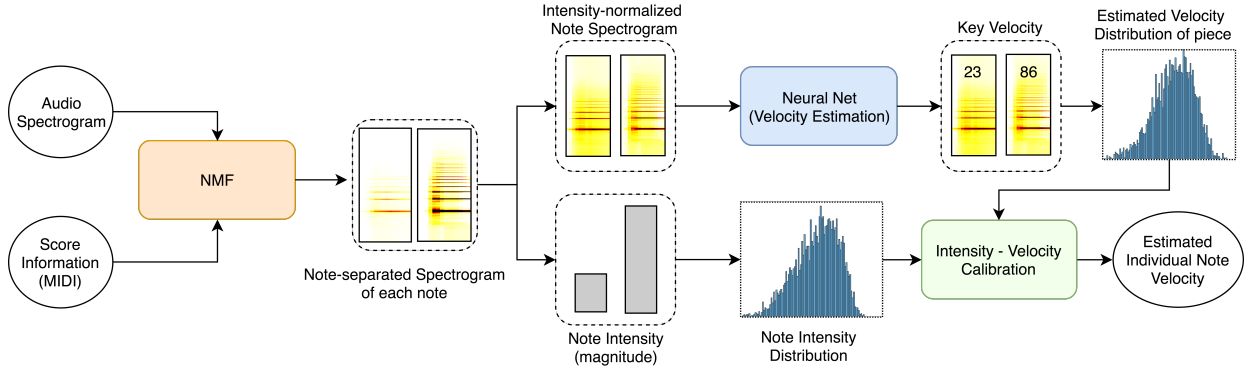


Figure 1. A diagram of the proposed system.

pitch. By doing so we can decompose the input spectrogram with spectral templates bases \mathbf{W} and the activation of the bases over time \mathbf{H} .

To clarify the relationship between spectral basis and pitch, we will follow the similar notation presented in [24], denoting $\mathbf{W}_{f,p,r} := \mathbf{W}_{f,(p-1) \cdot R + r}$ and $\mathbf{H}_{p,r,t} := \mathbf{H}_{(p-1) \cdot R + r,t}$ as below:

$$\mathbf{V}_{ft} = \sum_{p,r} \mathbf{W}_{f,p,r} \mathbf{H}_{p,r,t} \quad (1)$$

where $f \in [1, F]$, $t \in [1, T]$, $p \in [1, P]$, and $r \in [1, R]$ are index of frequency bin, time frame, pitch, and spectral basis in a pitch, respectively.

4.1.1 NMF Modeling

We employ an NMF model that learns multiple time-frequency patterns instead of single spectral templates [24], which was applied to the score-informed AMT task. This model captures various timbre of the same pitch and temporal evolution of timbre, which is a necessary part of our task. Since the main contribution of our paper lies on the velocity estimation by combining of the NMF and NN results, the following section will mainly explain several differences in the implementation. The details are found in [24].

Considering that an NMF model can be configured mainly by the number of basis, initialization method, and additional constraints with corresponding update rules, Wang *et al.*'s model for piano recording [24] is different from the previous models used in [8, 9, 15] in three aspects.

First, they suggested multi-basis per pitch so that each pitch has R number of corresponding bases. The previous models represent a piano note by the combination of percussive (onset) and harmonic (sustain) basis for the whole note duration. Since there is only one harmonic basis for each pitch, the spectral shape of the note does not change over time. This assumes that the most important timbre feature is constant in the sustain part within the single note as well as for different key velocities. But the multi-basis model can handle this subtle change of timbre by using multiple bases with different activation ratios.

Second, employing the multi-basis model requires a different initialization method for matrix \mathbf{W} and \mathbf{H} . To model

temporal progression of piano timbre, the r -th basis was initialized to be active after the $(r - 1)$ -th basis of the same pitch. Since the pitch bases are activated sequentially, they can model temporal evolution of the note tone. As the pitch bases are differed by their activation initialization, they also have different spectral characteristics. Among R bases of a pitch, the first basis handles percussive element and the second to the last represent harmonic elements in the temporal order. In addition, the harmonic area is set to be tapered as the rank index r increases. This makes the earlier bases include more inharmonicity.

Third, Wang *et al.*'s model suggested several additional costs for the multi-basis model. They include a soft constraint, temporal continuity, and energy decay in the template matrix. Among the suggested costs, we did not employ the decaying cost for \mathbf{W} , which encourages smooth decrease of energy in spectral templates in \mathbf{W} . We found that our system works better with L1 normalized \mathbf{W} so that the magnitude feature is assigned only to \mathbf{H} . We followed the NMF costs and update function strictly except that we ignore the decaying cost term by assign 0 to β_3 .

For better intensity estimation, we previously suggested using power spectrogram, instead of linear magnitude spectrogram [15]. We also showed that using synthesized monophonic scale tones helps to learn spectral template. Based on this observation, our system also uses power spectrogram and synthesized piano scale. Another difference with [24] is post-updating of \mathbf{H} . After the update converges, we set all constraints on \mathbf{H} to zeros and update \mathbf{H} for ten times with fixed \mathbf{W} so that our final reproduction can resemble the original gain.

The NMF module reproduces note-separated spectrogram $\hat{\mathbf{V}}^{(n)}$ for each note n in the score by multiplying the spectral bases of note's pitch and its activation over note's duration. The note intensity is defined as the maximum activation of $\hat{\mathbf{V}}^{(n)}$, which can be represented as $\max(\sum_f \hat{\mathbf{V}}_{ft}^{(n)})$. Then, we reproduce $\hat{\mathbf{V}}^{(n)}$ again around the time frame of the maximum activation and store it for the input for the neural network. This helps to fix the size of NN's input and maintain the relative position of each element in the cropped spectrogram.

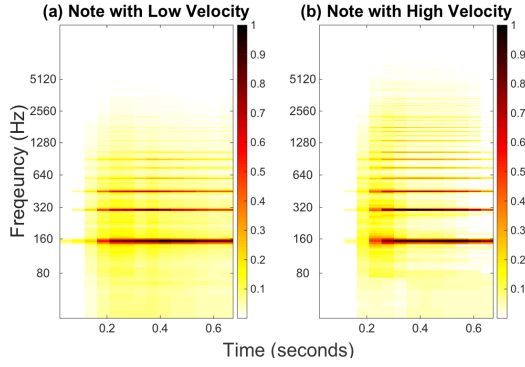


Figure 2. Comparison of the intensity-normalized note-separated spectrogram with different MIDI velocities. The spectrogram was reproduced from polyphonic piano recording (SMD). The MIDI note number is 50 and the MIDI velocities were 14 and 95, respectively.

4.2 Velocity Estimation

The neural network (NN) model takes the note-separated spectrograms from the NMF module as input and estimates the velocity of each note. The note-separated spectrogram is converted to a log-frequency spectrogram before it is used for the input of the NN module. The frequency resolution is set to 25 cent and the frequency range is from 27.5 Hz (the lowest pitch of piano) to 16.7 kHz (two octave higher than the highest pitch of piano), resulting in 445 frequency bins. After some preliminary test, we used 14 frames as input size. The spectrogram magnitude is normalized by the maximum value so that every entry in the spectrogram lies between 0 and 1 as shown in the Figure 2.

The neural network consists of 5 fully-connected hidden layers and each layer has 256 nodes. Every hidden layer uses SELUs as an activation function [17]. Applying SELUs aims to stabilize the network from internal covariance shifting without any additional complexity.

The loss function is set to mean square error of key velocity estimation, approaching the task as a regression problem. We also attempted to use softmax as a classification problem but the result was slightly worse. We used Adam optimization [16] with initial learning rate of $1e-4$, and early stopping on the validation set.

4.3 Intensity-to-Velocity Calibration

The NN module provides an absolute degree of note dynamics but the relative magnitude between each note from the NMF results is more stable than that from the NN results. Therefore, we combine the two results to find better estimation.

As described in Section 1, intensity is affected by both key velocity and recording condition. One cannot distinguish whether the high intensity from the NMF is caused by strong strike of hammer or high gain in the recording device. Therefore each recording condition needs its own mapping parameter.

Also, the intensity-velocity relation depends on a piano or a keyboard model [3]. Our previous study showed that the MIDI velocity of a note can be approximated by a linear relationship with the log value of the intensity $\text{Int}(n)$, so that $\text{Vel}(n) = a \cdot \log(\text{Int}(n)) + b$ for the Disklavier, which we use for the evaluation [15]. However, we need to know intensity-paired velocity in the target recording condition, which is not available in real-word recordings.

Our solution is estimating it from the overall velocity distribution of each piece from the NN module. If we assume the outcome velocity has a distribution with mean μ_V and standard deviation σ_V for each piece, we can obtain the mapping parameters by comparing it with the distribution of log of intensity, $\mu_{\log(I)}$ and $\sigma_{\log(I)}$. Then, the mapping parameter a and b correspond to $\sigma_V / \sigma_{\log(I)}$ and $\mu_V - (\sigma_V / \sigma_{\log(I)}) \mu_{\log(I)}$, respectively, with the assumption that every note has the same mapping parameters. Note that this neglects the note-specific difference of intensity-to-velocity mapping parameter. The error caused by this assumption will be also explained in Section 6.

Our system takes the result of the NN module to estimate μ_V and σ_V for each piece. The estimation can be also done by a simple global setting. During the evaluation, we used this scheme as a baseline to compare with our NN model.

5. EXPERIMENT

5.1 Experiment I: SMD

We used Saarland Music Dataset (SMD) MIDI-Audio Piano Music [18] for the evaluation. The dataset consists of fifty pairs of audio and MIDI recordings of performance on Yamaha Disklavier DCFIISM4PRO. The MIDI files of SMD contain every movement of piano key and pedal in high reliability, thus providing the ground truth of note dynamics in MIDI velocity.

The previous work pointed out that the recording condition of each piece in SMD is differed by its recording date [15]. Therefore, the intensity-to-velocity mapping had to be obtained separately for each subset of pieces that share the same recording condition. The difference in intensity-to-velocity mapping in SMD is represented in Figure 3. Since the goal of the proposed system is to estimate key velocity robustly against changes in the recording environment, such different recording conditions are ideal for evaluating this task.

We evaluate whether the proposed system can handle different recording conditions and estimate correct velocity distributions. We used fifteen pieces recorded in the year of 2011 as a test set, and other thirty-five pieces as a training set, which was recorded during the year of 2008 and 2010.

To evaluate the exact performance and usefulness of the NN module, we also present two upper boundary models and a baseline model. The first upper boundary assumes that the system obtained proper mapping parameters for every individual pitch from other pieces in the same test set, as in [15]. The second upper boundary assumes that our NN module guessed correct estimation of velocity dis-

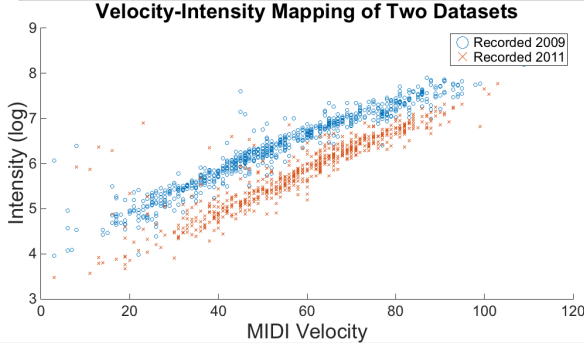


Figure 3. The difference in velocity-intensity mapping between two subsets from SMD. Each point represents a single note with MIDI note number 50. The notes recorded in 2009 show higher intensity compared to the notes recorded in 2011 given the same velocity.

tribution. In this upper boundary, we employed the ground truth of velocity distribution for each piece. The baseline is using global mean and standard deviation values. Based on the statistics of training set, we used $\mu_V = 57.87$ and $\sigma_V = 16.25$.

The evaluation measure is an absolute error of velocity between ground truth and estimated value. In MIDI velocity dimension, absolute error is a more meaningful criterion than relative error because MIDI velocity is already a logarithm of the intensity. We used the average of absolute velocity error in a piece, which can be represented as $\text{Err} = \sum_n |V_{\text{GT}}(n) - V_{\text{Est}}(n)| / N$, where $V_{\text{GT}}(n)$ and $V_{\text{Est}}(n)$ are ground truth velocity and estimated velocity of the n -th note in a piece, respectively.

5.2 Experiment II: MAPS

We also evaluate our NN module on unseen data to see whether the NN can learn generalized piano timbre from the training set. To this end, we designed another experiment with the MAPS database [5], which was recorded with a different piano and recording conditions.

From the MAPS dataset, we used two subsets performed by Yamaha Disklavier Mark III (upright) that consists of 30 recordings. One subset is recorded as “ambient” and the other is recorded as “close” condition. We did not use other MAPS dataset for training our NN module. The model trained from thirty-five pieces of SMD was used for this test.

In this experiment, the evaluation is made only with the estimated distribution from the NN module μ_{nn} and σ_{nn} and ground truth μ_{gt} and σ_{gt} . Since the mapping between key velocity and MIDI velocity in SMD and the MAPS dataset is different, we cannot compare these values directly. Also, we cannot figure out how the same key velocity will be recorded as MIDI velocity in SMD and MAPS or which velocity value will make most close reproduction of a note in MAPS with the instrument in SMD. What we can assure is that MIDI velocity ranking of notes or piece will be preserved both in SMD and MAPS. Therefore we

examine the Spearman correlation between the NN’s guess μ_{nn} and σ_{nn} and the ground truth MAPS MIDI value μ_{gt} and σ_{gt} .

5.3 Procedure

The experiment procedure is as follows. First, the NMF module calculates note intensity and reproduces note-separated spectrograms for each pieces in the training set and test set. Then, we train the NN module with the note spectrograms of the training set from SMD. After the training, the trained NN estimates the velocity of note spectrograms of the test set. Combining the distribution of estimated velocity from the NN and estimated intensity from the NMF as described in section 4.3, we can obtain final MIDI velocity for each note in the piece. For the Experiment II, the calibration part is omitted. During the experiment, we used STFT with window size 8192, hop size 2048, and 8 spectral bases per pitch in the NMF module.

6. RESULTS

6.1 Experiment I: SMD

We present our result on the SMD set recorded in 2011 on Table 1. The ground truth velocity distribution of each piece is represented as GT, and the estimated distribution from the NN module is as NN. The remaining columns on the right are the average errors of four different mapping parameter for the same NMF result. UB1 is the first upper boundary that uses other test pieces to obtain the velocity-to-intensity mapping as in [15]. UB2 is the second upper boundary that assumes our NN module estimated the correct μ_V and σ_V . The proposed method (Prop.) is from the NN estimation for μ_V and σ_V . The baseline (Base) always guessed $\mu_V = 57.87$ and $\sigma_V = 16.25$. The last column shows the error when we directly used the NN estimation in note level, instead of combining it with the NMF intensity.

The estimation of the NN module showed high error in a note level as shown in the NN column. We presume the reason for the error is mainly based on the imperfection of source-separation. Also, the different recording condition in the test set could make not only intensity difference but also timbral change. This inhomogeneity may also have had a negative impact on the performance of the NN module.

Even though the note-level accuracy was not reliable, we found that the overall distribution of the estimated velocity resembles the distribution of ground truth velocity as we expected. By employing the estimated velocity distribution, the note intensity from the NMF module could be successfully mapped into MIDI velocity as shown in the Prop. column. The proposed system outperforms the baseline estimation in most pieces. While the fixed guess ignored characteristic of each piece, the NN module successfully estimated a correct distribution from the note spectrograms.

The difference between two upper boundary UB1 and UB2 shows the error caused by the assumption that the

Composers	Piece	Ground Truth		NN Estimation		UB1	UB2	Proposed	Baseline	NN note
		Mean	STD	Mean	STD	Err	Err	Err	Err	Err
Bach	BWV 888-1	49.7	12.6	53.3	15.5	3.1	3.9	3.3	6.6	10.4
Bach	BWV 888-2	63.3	11.3	62.8	13.3	2.1	3.1	3.5	9.0	10.1
Bartok	op. 80-1	68.9	18.2	65.7	15.3	5.9	6.6	8.5	15.0	12.2
Bartok	op. 80-2	59.5	23.5	59.5	20.4	5.1	7.2	8.6	10.3	11.3
Bartok	op. 80-3	67.4	19.0	64.8	17.3	6.0	7.1	8.9	14.8	13.0
Brahms	op. 5-1	64.8	23.5	62.0	19.6	7.2	8.4	10.0	13.1	13.8
Haydn	HobXVI-52-01	57.9	14.6	58.4	14.7	3.9	5.5	4.6	6.1	11.9
Haydn	HobXVI-52-2	49.8	18.6	53.9	16.6	3.8	4.7	5.6	8.0	11.1
Haydn	HobXVI-52-3	60.4	12.9	59.1	15.5	3.6	5.4	5.5	7.6	12.9
Mozart	K. 265	57.5	13.2	57.1	14.4	3.2	6.2	6.2	6.7	10.5
Mozart	K. 398	58.6	13.2	57.7	16.5	3.6	5.6	8.5	8.6	11.2
Rachmaninoff	op. 36-1	56.5	18.7	54.5	16.9	6.4	6.1	6.9	5.9	11.7
Rachmaninoff	op. 36-2	54.7	19.5	50.2	18.1	5.2	5.5	6.4	6.9	11.5
Rachmaninoff	op. 36-3	66.3	19.8	66.4	16.0	6.6	9.0	8.5	14.7	12.7
Ravel	Jeux d'eau	55.3	17.0	57.8	17.6	5.8	5.5	5.0	5.1	12.5
Average						4.83	5.9	6.7	9.2	11.8

Table 1. The result of experiment on SMD. The first two columns show mean and standard deviation of note velocities from the ground truth and the estimation by neural network. “Err” stands for absolute mean error of note velocities. UB1 is an oracle model that learns key-dependent velocity mapping from other test pieces, and UB2 is another oracle model with ground-truth velocity mean and variance. The baseline model uses a global mean and variance. NN note represents mean error of velocity estimation of individual notes in the neural network

intensity-to-velocity mapping is consistent over the key. However, previous works showed that a piano stroke makes different intensity with the same velocity depending on the key [20]. This suggests the need of additional methods to compensate the key-dependent mapping in the future research.

The error is notable in *Rachmaninoff’s Op. 36-1*. A possible reason is that the global setting of velocity distribution in the baseline is closer to the ground truth compared to the NN estimation. The errors in *Ravel’s Jeux d’eau* is worth mentioning since the two upper boundary methods made the worse result. We presume that the reason is the frequent use of soft pedal during the performance. Soft pedal makes intensity lower, thus making our system estimate it softer than what is expected from its MIDI velocity.

6.2 Experiment II: MAPS

Figure 4 shows the correlation between the estimation from the NN module and the ground truth on the MAPS recordings. The absolute value of μ_{nn} and μ_{gt} has an error because of different key velocity to MIDI velocity mapping, thus cannot be compared directly. However, we can see that as the ground truth velocity mean of the piece increases, the estimated mean of NN also tends to catch it up. The same tendency is also found in the standard deviation. The Spearman correlation between μ_{GT} and μ_{NN} is 0.838, and that between σ_{GT} and σ_{NN} is 0.597.

Figure 4 also shows that the estimation from the NN module is not affected much by whether the recording is ambient or close, indicating that our NN module is robust to different piano and recording conditions. We did not apply the baseline method to MAPS because the estimation would be always constant regardless of the piece.

7. CONCLUSIONS

We presented a system that estimates key velocity from polyphonic piano recordings. The main limitation of pre-

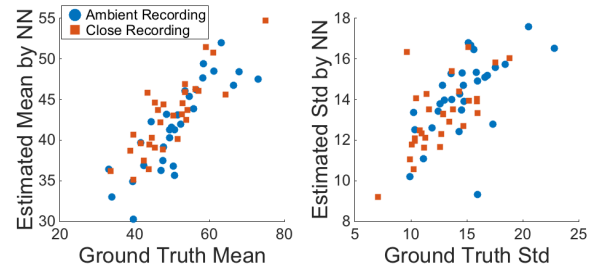


Figure 4. The test result on the MAPS dataset (Experiment II). Each point represents a single piece.

vious work was the lack of method for calibration between intensity and key velocity. To overcome the limitation, We proposed a neural network module that takes note-separated spectrogram and estimates the key velocity of each note. Though the accuracy of individual notes is not reliable, the overall distribution resembles the distribution of ground truth velocity for each piece. Our system obtains a proper intensity-to-velocity mapping by employing the estimated velocity distribution, and then estimate the key velocity.

We evaluated our system on two different datasets. Overall, the evaluation showed a promising result of this timbre-based approach. The velocity estimation from the NN module showed a similar distribution with the ground truth velocity distribution despite the different recording conditions. Employing this estimated distribution, our system mapped note intensity to MIDI velocity reliably. Also, the result showed that our NN module learns robust features that can be applied to unseen data.

For the future work, we plan to apply our solution to real-world recordings with various timbre and recording conditions and, by combining other AMT and audio-to-score alignment algorithms, and obtain more full-fledged performance transcription.

8. ACKNOWLEDGEMENTS

This research was supported/partially supported by Samsung Research Funding & Incubation Center for Future Research.

9. REFERENCES

- [1] Emmanouil Benetos, Simon Dixon, Dimitrios Gianoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.
- [2] D. G. Bhalke, C. B. Rama Rao, and D. S. Bormane. Automatic musical instrument classification using fractional fourier transform based- MFCC features and counter propagation neural network. *Journal of Intelligent Information Systems*, 46(3):425–446, Jun 2016.
- [3] Roger B Dannenberg. The interpretation of MIDI velocity. In *Proc. of International Computer Music Conference (ICMC)*, pages 193–196, 1996.
- [4] Patrick Joseph Donnelly et al. *Learning spectral filters for single-and multi-label classification of musical instruments*. PhD thesis, Montana State University-Bozeman, College of Engineering, 2015.
- [5] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.
- [6] Antti Eronen and Anssi Klapuri. Musical instrument recognition using cepstral coefficients and temporal features. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages II753–II756, 2000.
- [7] Sebastian Ewert and Meinard Müller. Estimating note intensities in music recordings. In *Proc. of 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 385–388, 2011.
- [8] Sebastian Ewert and Meinard Müller. Using score-informed constraints for NMF-based source separation. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 129–132, 2012.
- [9] Sebastian Ewert, Siying Wang, Meinard Müller, and M Sandler. Score-informed identification of missing and extra notes in piano recordings. In *Proc. of International Society of Music Information Retrieval Conference (ISMIR)*, pages 30–36, 2016.
- [10] Werner Goebel and Roberto Bresin. Measurement and reproduction accuracy of computer-controlled grand pianos. *The Journal of the Acoustical Society of America*, 114(4):2273–2283, 2003.
- [11] Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 25(1):208–221, 2017.
- [12] Harry C Hart, Melville W Fuller, and Walter S Lusby. A precision study of piano touch and tone. *The Journal of the Acoustical Society of America*, 6(2):80–94, 1934.
- [13] Toni Heittola, Anssi Klapuri, and Tuomas Virtanen. Musical instrument recognition in polyphonic audio using source-filter model for sound separation. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, pages 327–332, 2009.
- [14] Kristoffer Jensen. *Timbre models of musical sounds*. PhD thesis, Department of Computer Science, University of Copenhagen, 1999.
- [15] Dasaem Jeong and Juhan Nam. Note intensity estimation of piano recordings by score-informed NMF. In *Proc. of Audio Engineering Society Semantic Audio Conference*, 2017.
- [16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computing Research Repository*, abs/1412.6980, 2014.
- [17] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 972–981, 2017.
- [18] Meinard Müller, Verena Konz, Wolfgang Bogler, and Vlori Arifi-Müller. Saarland music data (SMD). In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR): Late Breaking session*, 2011.
- [19] Bruno H Repp. Music as motion: A synopsis of Alexander Truslit’s (1938) *Gestaltung und Bewegung in der Musik*. *Psychology of Music*, 21(1):48–72, 1993.
- [20] Bruno H Repp. Some empirical observations on sound level properties of recorded piano tones. *The Journal of the Acoustical Society of America*, 93(2):1136–1144, 1993.
- [21] Bruno H Repp. The dynamics of expressive piano performance: Schumann’s “Träumerei” revisited. *The Journal of the Acoustical Society of America*, 100(1):641–650, 1996.
- [22] Craig Stuart Sapp. Comparative analysis of multiple musical performances. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 497–500, 2007.
- [23] Wai Man Szeto and Kin Hong Wong. Source separation and analysis of piano music signals using instrument-specific sinusoidal model. In *Proc. of 16th International Conference on Digital Audio Effects (DAFx)*, 2013.

- [24] Siying Wang, Sebastian Ewert, and Simon Dixon. Identifying missing and extra notes in piano recordings using score-informed dictionary learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(10):1877–1889, 2017.
- [25] Gerhard Widmer, Simon Dixon, Werner Goebel, Elias Pampalk, and Asmir Tobudic. In search of the Horowitz factor. *AI Magazine*, 24(3):111, 2003.

TIMBRE DISCRIMINATION FOR BRIEF INSTRUMENT SOUNDS

Francesco Bigoni

Sound and Music Computing
Aalborg University – Copenhagen
fbigon17@student.aau.dk

Sofia Dahl

Dept. of Architecture, Design and Media Technology
Aalborg University – Copenhagen
sof@create.aau.dk

ABSTRACT

Timbre discrimination, even for very brief sounds, allows identification and separation of different sound sources. The existing literature on the effect of duration on timbre recognition shows high performance for remarkably short time window lengths, but does not address the possible effect of musical training. In this study, we applied an adaptive procedure to investigate the effect of musical training on individual thresholds for instrument identification. A timbre discrimination task consisting of a 4-alternative forced choice (4AFC) of brief instrument sounds with varying duration was assigned to 16 test subjects using an adaptive staircase method. The effect of musical training has been investigated by dividing the participants into two groups: musicians and non-musicians. The experiment showed lowest thresholds for the guitar sound and highest for the violin sound, with a high overall performance level, but no significant difference between the two groups. It is suggested that the test subjects adjust the weightings of the perceptual dimensions of timbre according to different degrees of acoustic degradation of the stimuli, which are evaluated both by plotting extracted audio features in a feature space and by considering the timbral specificities of the four instruments.

1. INTRODUCTION

Timbre is a primary vehicle for sound source recognition and, from a cognitive perspective, sound identity [10]. The auditory system is designed to identify sound sources: this enables us to discern a melody in a complex soundscape, follow what is being said by a speaker, or step aside when something fast and dangerous appears to be approaching. As an example which is more related to music consumption, listeners are able to identify musical genres better than chance in a fraction of a second (the shortest duration tested is 125 ms [9]).

Although so important to our auditory system, timbre is often defined in a negative manner — as, for instance, in Plomp’s (1970) operational definition: “Timbre is that attribute of sensation in terms of which a listener can judge

that two steady complex tones having the same loudness, pitch and duration are dissimilar” (quoted in [12]). Timbre can be described as a set of perceptual attributes which are either continuously varying (timbral semantics such as attack sharpness, brightness, richness) or discrete (perceptual features such as the pinched offset of a harpsichord sound) [10]. For the former category of attributes, a number of objective acoustic correlates can generally be found among spectro-temporal audio features, e.g. spectral centroid, attack time and spectral envelope; for the latter, the objective correlates are harder to identify [10].

Being complex and multidimensional, timbre is usually modelled employing a so-called multidimensional scaling (MDS), i.e. fitting the dissimilarity ratings given by a group of listeners on a set of sounds to a *timbre space* of perceptual attributes and respective acoustic correlates [10]. While the basic MDS model assumes the same perceptual dimensions for all listeners, more recent models (e.g. CLASCAL by McAdams et al. [11]) account for different weightings of the perceptual dimensions (by individual listeners or classes of listeners) and for the effect of the features that are specific to an individual timbre, called “specificities” (basically related to the aforementioned discrete features).

The studies that evaluate the effect of brief duration on timbre perception exhibit a decidedly different approach from MDS research: quoting Suied et al., MDS models aim at finding the most *prominent* perceptual dimensions of specific sounds through dissimilarity ratings, whereas the problem of timbre recognition for brief sounds asks to identify the most *informative* ones [21]. Inside this field, the prevalent area of interest is speech: in a seminal paper from 1942, Gray investigated phoneme cues for short vowel sounds, and coined the term “phonemic microtomy” [5]. More recently, Robinson and Patterson found that timbral cues for brief vowel stimuli are not pitch-assisted [18]. Generally, the measured performance is above chance for durations as short as a single glottal pulse cycle, on the order of 3 ms.

Only a few studies deal with non-speech sounds: Clark et al. asked their test subjects to identify orchestral instruments for varying window length and position for gated stimuli [3]; Robinson and Patterson replicated their previous study using synthesized instrument sounds, achieving slightly lower performance levels than for voice stimuli [17]. In later articles, the sound recognition problem has been stated in different terms, by taking the subjective reaction



© Francesco Bigoni, Sofia Dahl. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Francesco Bigoni, Sofia Dahl. “Timbre Discrimination for Brief Instrument Sounds”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

times rather than the temporal thresholds of the stimuli into account [1, 22].

In 2014, Suied et al. published what we consider by far the most exhaustive contribution on the topic, as well as the most relevant reference for our paper [21]. In a series of timbre discrimination experiments, participants were asked to identify whether a sound belonged to a target category (e.g. strings, percussion, voice) or a distractor category. Very short duration thresholds were found, on the order of 8-16 ms. The best performance was for voice, followed by percussion (marimba and vibraphone). Subsequent experiments rejected the effect of feedback and training on the performance for the voice stimuli; and, finally, demonstrated that source recognition based on timbral cues is fast and robust to stimulus degradation, with a clear advantage for voice signals.

While it may not be surprising that humans are highly trained to identify sounds as belonging to the “voice” category, one could expect more variability in the exposure to instrumental sounds. Suied et al. [21] did not report any information regarding the musical training of their participants. We would expect that listeners who are trained as musicians would exhibit lower threshold values compared to non-musicians.

Previous studies [17, 21] have used constant stimuli lengths, with durations that are doubled. The increasing differences in durations help to reduce the test time, but also make it difficult to pinpoint where and how thresholds differ between individuals or groups of listeners. We expect musically trained and untrained listeners to differ in the overall threshold of instrument discrimination, but there may be an interaction with the instrument type. Suied et al. [21] found a lower performance for the “strings” category compared to “percussion”. In order to efficiently target the listeners’ individual thresholds, an adaptive approach is an attractive alternative.

In this paper, we investigate whether musical training has an effect on the perceptual interaction between timbre and duration through a timbre discrimination task, using brief sounds of varying length. Our goals were threefold: 1) applying an adaptive staircase method to estimate the temporal thresholds of timbre discrimination for a small sound set (four instruments: guitar, clarinet, trumpet and violin); 2) determining if musical training has an effect on the task; 3) relating the degradation of timbre descriptors (caused by the length shortening) to the perceptual adaptation strategies of the participants.

2. METHOD

We anticipated the range of thresholds to vary between participants, and therefore opted for an adaptive test procedure. Adaptive methods are designed to be time-efficient and focus the presentation of stimuli around the perceptual threshold of interest by adapting the level of presentation according to the past responses of the participant (in our case, the indication of the heard instrument). Compared to the method of constant stimuli, the adaptive procedure can quickly move from presenting clearly audible

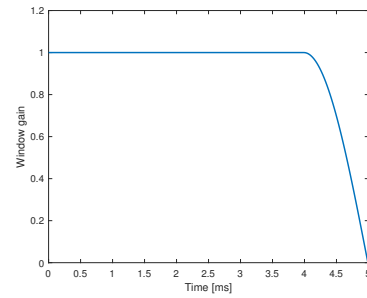


Figure 1. Time window employed for stimulus gating (in this case, the window length is 5 ms).

and distinguishable stimuli to a range where performance is more difficult. By gradually decreasing the *step size* after a change in test subject performance, the method allows to zoom in rapidly on the threshold level. Depending on the criteria for changes in presentation level and step size, the adaptive procedure can be designed to target different performance levels on the psychometric function (see [7] for an overview). For this study, we chose the simple up-down staircase method [8], as this does not require assumptions on the shape of the psychometric function.

2.1 Stimuli

The four stimuli (guitar, clarinet, trumpet and violin) were picked from an existing database of anechoic recordings of acoustic instruments [20] [23]. The audio files were recorded at a sample rate of 48 kHz and a resolution of 24 bits, using a 32-channel microphone array. The audio editing was performed in the digital audio workstation Reaper. Four source files were created by mixing down the respective 32 channels to a mono track (with no instrument-specific mix), bounced at 16-bit/44.1 kHz. In the source files, the instrumentalists are playing a C4 at a *ff* dynamic. The pitch of the source files was already normalized, as the instruments were all tuned at A4 = 443 Hz¹. Sounds were loudness-normalized to -18 LUFS using the SWS extension in Reaper. The sound snippets were prepared on the fly in MATLAB between the presentations, by applying a suitable window (i.e. a rectangular window with 4 samples of silence at the start and a 1 ms equal-power fade-out at the end) of the required duration, starting from time 0: an example is shown in Figure 1. Thus, onset information has been included in each snippet.

2.2 Participants

A convenience sample of 16 participants was tested, consisting of 13 males and 3 females with ages ranging from 22 to 50 ($\mu_{age} = 32$, $\sigma_{age} = 9$) recruited through author Bigoni’s personal network. Participants indicated their age and sex (if willing to disclose) and whether they had normal hearing (no testing was made to assess this); they

¹ This gives a fundamental frequency of $443/2^{(9/12)} = 263.41$ Hz at C4.

were informed that their personal data would be anonymized. Each test subject was assigned to one of two groups (*musicians* or *non-musicians*) by asking if he/she had 5 or more years of formal musical training and/or performance experience. This left the border between the two groups somewhat flexible, giving the option to music students and amateur musicians to choose their group according to their confidence level. The groups were fairly balanced with respect to sample size: 9 musicians and 7 non-musicians. Of the 9 musicians, 5 are primarily performing on wind instruments, whereas the other 4 play different combinations of guitar, piano, drums and electronics. Despite this inter-group difference, we do not assume that any of the subjects were biased towards a specific instrument type.

2.3 Setup

The playback system consisted of the laptop internal sound card, driven with ASIO4ALL drivers for Windows, and a pair of Beyerdynamic DT 990 Pro, 250 Ohm headphones. Even though the DT 990 Pro do not have a flat frequency response, we assume that the sound coloration introduced by the headphones did not alter the relative timbre perception.

2.4 Procedure

The experimental setup was implemented in MATLAB. It features a simple GUI and consists of three steps: 1) creation of a test subject entry in a database; 2) soundcheck: the subject can click on four buttons to play the source files (guitar, clarinet, trumpet, violin) while adjusting the headphones volume to a comfortable level. The soundcheck also constitutes a small training session on the four timbres, to avoid confusion at the beginning of the discrimination task; 3) timbre discrimination task: an adaptive staircase method (simple up-down) with four interleaved tracks (the four instrument timbres). For each sound presentation, the participants made a 4-alternative forced choice (4AFC) test. For each track, the following procedure applied: when a participant correctly identified the instrument, the duration of the next presentation (of the same instrument) would be reduced by the step size (initially, 40 ms); on the other hand, the duration of the next presentation would be augmented by the step size after a wrong answer. In the literature, right and wrong answers usually get represented by positive (+) and a negative (-) signs respectively. In the light of this notation, a *run* consists of a sequence of presentations that get answers of the same sign, and a *reversal* occurs at each change of sign. Thus, after the first misidentification, the first reversal would occur, the first run would end, the step size would be halved and the duration would be lengthened (by 20 ms) for each wrong answer; at the next right answer, another reversal would occur, the second run would end and the next presentation would, again, be shortened by the step size. For each track, the initial snippet length was set to 500 ms, the step sizes (halved at the end of each odd run) to 40/20/10/5/2 ms and the stop criterion to 8 reversals.

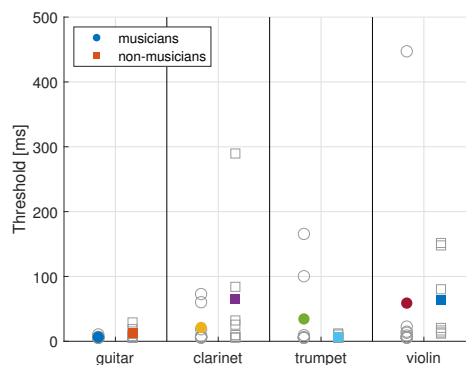


Figure 2. Thresholds for each group and instrument sound, both individual (grey) and group-based means (colour-filled symbol) for musicians (circle) and non-musicians (square). The different variability in thresholds between instruments is clearly seen.

The order of stimuli presentations was made by interleaving the four tracks using a random permutation of a 4x4 integer sequence of indices. This technique allows the same timbre to be replayed before a sequence of 4 is completed, removing a potential bias by avoiding the possibility of the subject anticipating the next sound. The typical test time was 15-20 minutes (setup + 150-200 presentations), with a shortest played duration of 1 ms.

2.5 Analysis

The typical shortest durations played during tests ranged between 1 and 10 ms across all participants. The four thresholds (one per instrument) were computed as the mean of the thresholds at reversals. The simple up-down estimates point $p = 0.50$ on the psychometric function, which is well above chance performance for 4AFC ($p = 0.25$).

The performance difference between the two groups (musicians and non-musicians) was estimated by performing a mixed ANOVA (between-subjects variable: 2 levels of musical training, within-subjects variable: 4 instrument sounds).

Moreover, eight sound snippets were created using the found thresholds and two audio descriptors (spectral centroid and spectral irregularity) were computed in MATLAB using *MIRtoolbox 1.7* [6].

3. RESULTS

The outcome of the experiment is shown in Figure 2, with the threshold means and standard deviations re-stated in Table 1. It can be seen that the threshold values vary considerably across groups and instruments, with very low mean values for guitar and trumpet (for non-musicians only), and mean values almost 10 times higher for violin. Furthermore, variability is large for all thresholds, except guitar. A Q-Q plot showed that the data violates

Stimulus	Mean (std) [ms]	
	Mus	Non-mus
Guitar	6.4 (1.6)	12.2 (8.8)
Clarinet	21.3 (26.1)	64.7 (102.9)
Trumpet	34.4 (58.2)	7.4 (2.7)
Violin	58.9 (145.7)	63.2 (63.7)

Table 1. Mean and standard deviation of thresholds of timbre discrimination. Mus = values from 9 musicians, Non-mus = values from 7 non-musicians.

the normality assumption, while a Levene’s test indicated that group variances are homogeneous. After improving normality with a 10-log transformation, we proceeded with a mixed ANOVA ($\alpha = 0.05$) on the transformed data, looking for statistically significant effects of musical training and stimulus. The between-subjects factor was group (musicians/non-musicians) and the within-subjects factor was target (instrument). The Q-Q plot of the ANOVA residuals is approximately linear, so we assume that this analysis is robust with respect to our dataset. While the stimulus effect was statistically significant ($F(3, 42) = 5.035$, $p = 0.005$), the musical training effect on timbre discrimination of brief sounds was not ($F(1, 14) = 1.134$, $p = 0.305$). The interaction effect was not significant either ($F(3, 56) = 2.416$, $p = 0.080$).

Post-hoc pairwise t-tests (two-sided, Holm-Bonferroni correction) on the instrument thresholds showed that the guitar mean threshold was significantly different from violin ($t(15) = -1.833$, $p = 0.024$), but not from clarinet ($t(15) = 4.873$, $p = 0.095$) or trumpet ($t(15) = -1.187$, $p = 0.871$). No other contrasts were significant — trumpet vs violin ($t(15) = -1.780$, $p = 0.081$), clarinet vs trumpet ($t(15) = -1.913$, $p = 0.472$), clarinet vs violin ($t(15) = -2.097$, $p = 0.871$).

As a rough approximation of an MDS model (with equal perceptual weightings and no specificities), we created a feature space using two calculated audio descriptors: spectral centroid and spectral irregularity. Spectral irregularity is a measure of the amplitude deviation between successive peaks of the spectrum (implemented in *MIR-Toolbox 1.7* [6]), a feature analogous to spectral deviation. The two descriptors were chosen for two reasons: 1) they are informative as a set, as they are not strongly correlated (see e.g. [15]); 2) they can be computed as single-number features, and are thereby easy to visualize and more robust to the short snippet durations than other descriptors which require frame-based analysis, e.g. spectral flux.

The feature space is seen in Figures 3 (mean thresholds) and 4 (individual thresholds), with colours denoting the four instruments: guitar (black), clarinet (blue), trumpet (red), and violin (green). The values for the 2 s source files are not plotted in Figure 4, thereby the different x-axis scale. As a general trend, the spectral centroid gets lowered for reduced duration. On the other hand, the spectral irregularity seem to either stay constant (clarinet), increase (guitar and violin), or fluctuate (trumpet) depending on the instrument sound.

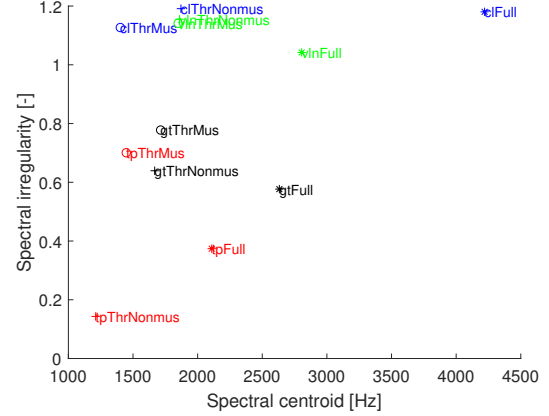


Figure 3. Mean thresholds for the four instruments in a feature space spanned by two audio descriptors: spectral centroid and spectral irregularity (computed using [6]) for the four instruments. Labels of the format xyz , with x defining the instrument (gt = guitar (black), cl = clarinet (blue), tp = trumpet (red), vln = violin (green)), y defining the duration of the audio file (Thr = audio snippet cut at threshold length, $Full$ = 2 s long source file) and z defining the test group (Mus = musicians, $Nonmus$ = non-musicians).

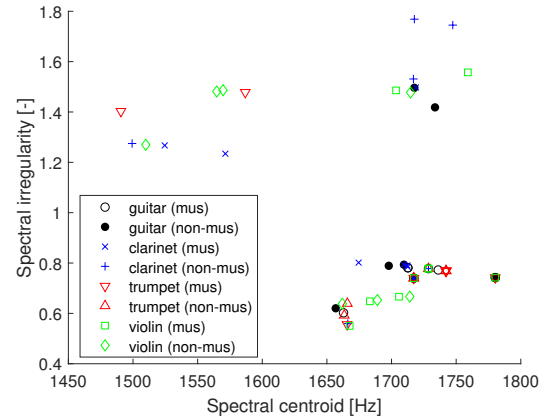


Figure 4. Thresholds for all participants and the four instruments in a feature space spanned by two audio descriptors: spectral centroid and spectral irregularity (computed using [6]) for the four instruments. The 2 s long source files are labelled $xFull$, with x defining the instrument (gt = guitar (black), cl = clarinet (blue), tp = trumpet (red), vln = violin (green)).

Additionally, we computed the log attack times of the four source files ($LAT_{guitar} = -1.921$, $LAT_{clarinet} = -0.930$, $LAT_{trumpet} = -0.506$, $LAT_{violin} = -1.092$). However, since the attack phase is incomplete for the threshold snippets, this feature was less informative in relation to the perceptual result.

4. DISCUSSION

Using an adaptive procedure, we investigated the temporal thresholds for timbre discrimination of different sounds for musically trained and untrained listeners. Our findings agree with the existing literature with respect to the overall high performance of both groups. The overall performance was best for guitar, both with respect to duration thresholds and variability, as confirmed by post-hoc tests.

While we measured an average violin threshold of about 60 ms, Suied et al. [21] report window lengths corresponding to above chance performance as small as 8 ms for string sounds in a first experiment, then doubled to 16 ms in a subsequent trial (with other instruments as distractors). Suied et al. offer two plausible interpretations of the high performance levels: a successful adjustment of the auditory representation of the stimuli, which is specific to the signal gating setup – which is described as a computationally challenging form of unsupervised learning – or an efficient activation of spectral cues even for deteriorated stimuli. Both interpretations could apply to our experiment.

Some differences between our method and that used by Suied et al. [21] are worth mentioning. While we asked our participants to indicate the instrument heard in a 4AFC-task that got more challenging over time, Suied et al. asked their subjects to indicate whether the sound was a target sound (50% of presentations) or not. The range of possible sounds was also wider in their study, with seven other instruments beside those belonging to the target. Before their test, however, the participants listened to the targets repeatedly for all stimuli durations. It is therefore difficult to judge whether this would result in a harder task for the participants compared to the one we chose. The task of categorizing (target vs distractor) or identifying (4AFC) sound are different: although the thresholds are still in the same range, the peculiarity of the tasks might explain the discrepancy between string instrument thresholds.

Our results showed no effect of musical training, possibly as a result of the moderate sample size and our operational definition of musician. Rather than dividing the participants in two groups (musicians and non-musicians), using an index of musical sophistication (e.g. [13, 14]) could provide a more sensitive measure and allow for a regression analysis. Moreover, differences between musicians and non-musicians have been shown in a combined instrument/voice discrimination task [2] as well as in brain activity [4], so group differences in terms of cognitive strategies should not be dismissed. On the other hand, our result agrees with the thesis of MDS researchers, meaning that the inter-individual differences in perceptual weighting of different timbral dimensions are independent of musical training [10].

The very low thresholds and low variability for guitar (both groups) and trumpet (non-musicians) seem to indicate the presence of early acoustical markers that could be identified by listeners. Even though it is commonly assumed that onset is highly significant for sound recognition (see e.g. [16] and [19]), this premise is not universally accepted by timbre/duration studies. It has been

doubted by Clark et al. [3] and then strongly disputed by Suied et al., who argue that onset information might even be misinformative for the discrimination of string instruments (due to the noisy transients caused by the initial contact between bow and string) [21]. However, the results shown by Suied et al. seem to indicate that the performance difference between the two window constraints (random and onset) is both stimulus-specific and inconsistent across window lengths. The gating used in their experiment applied a raised-cosine window, while we applied a rectangular window with a fixed fade-out length (1 ms) for all durations, as shown in Figure 1. Thus, our approach would be more likely to preserve the original amplitude for longer time (but with a sharper fade-out), while the stimuli prepared by Suied et al. would decrease in amplitude in a quicker and smoother manner. As for the acoustic analysis of the stimuli, Suied et al. explain the effect of gating in terms of “spectral splatter” (the smearing of spectral features when short time windows are applied) and refute the assumption that trivial spectral features are relevant to the timbre discrimination task, based on a simulation of auditory excitation patterns derived by the employed stimuli [21].

As a direct investigation of the stimuli, we placed the source files and threshold sound snippets in a feature space (Figures 3 and 4). Without perceptual weightings, this representation lacks the depth of MDS models, but it is useful to trace the deterioration of a set of audio descriptors (spectral centroid and spectral irregularity) for reduced duration. Even though the full set of thresholds forms three clusters (Figure 4) and most of the guitar data points are located in one of the clusters (lower right), it is hard to conclude that the guitar advantage is due to the fact that the stimulus retains specific audio features for brief durations. The threshold differences could instead be explained by the different placement of discrete timbral features (specificities), which are hard to correlate to the audio descriptors. The guitar advantage might be explained by our choice of the onset condition, which preserves the characteristic “twang” even for very brief window lengths. A more systematical investigation of the evolution of a set of audio descriptors for different stimuli and progressively decreasing duration would be worth considering for future work.

5. CONCLUSION

In this paper, we have investigated the temporal thresholds of timbre discrimination for four instrument sounds. Although the thresholds from the staircase method varied significantly between stimuli, with means ranging from < 15 ms (guitar) to ≈ 60 ms (violin), there was no significant effect of musical training on timbre discrimination. The guitar advantage can be explained by considering our choice of window position (always including the sound onset) and the timbral specificities of the guitar sound. The overall low thresholds agree with the findings of the existing literature, and the adaptive staircase method seems to constitute a viable alternative to the method of constant

stimuli for the chosen task. As a result, participants were able to adjust the weights of the perceptual dimensions of timbre to the acoustic degradation of the stimuli as the durations were reduced. Future research could use this investigation as a point for departure for a further examination of the duration thresholds, using larger sound sets and multiple window conditions.

6. ACKNOWLEDGEMENTS

The authors would like to thank all the participants and two anonymous reviewers for helpful comments on an earlier version of the manuscript.

Author Bigoni performed the main part of this work as part of a course in Music Perception and Cognition at the MSc program Sound and Music Computing, Aalborg University – Copenhagen. Author Dahl supervised the work and participated in writing the manuscript.

7. REFERENCES

- [1] Trevor R. Agus, Clara Suied, Simon J. Thorpe, and Daniel Pressnitzer. Fast recognition of musical sounds based on timbre. *The Journal of the Acoustical Society of America*, 131(5):4124–4133, May 2012.
- [2] Jean-Pierre Chartrand and Pascal Belin. Superior voice timbre processing in musicians. *Neuroscience Letters*, 405(3):164–167, Sep 2006.
- [3] Jr Clark, David Luce, Robert Abrams, Howard Schlossberg, and James Rome. Preliminary Experiments on the Aural Significance of Parts of Tones of Orchestral Instruments and on Choral Tones. *Journal of the Audio Engineering Society*, 11(1):45–54, January 1963.
- [4] Garry C. Crummer, Joseph P. Walton, John W. Wayman, Edwin C. Hantz, and Robert D. Frisina. Neural processing of musical timbre by musicians, non-musicians, and musicians possessing absolute pitch. *The Journal of the Acoustical Society of America*, 95(5):2720–2727, May 1994.
- [5] Giles Wilkeson Gray. Phonemic microtomy: The minimum duration of perceptible speech sounds. *Communications Monographs*, 9(1):75–90, 1942.
- [6] Olivier Lartillot and Petri Toivainen. A Matlab toolbox for musical feature extraction from audio. In *International Conference on Digital Audio Effects*, pages 237–244, 2007.
- [7] Marjorie R Leek. Adaptive procedures in psychophysical research. *Perception & psychophysics*, 63(8):1279–1292, 2001.
- [8] H. Levitt. Transformed Up–Down Methods in Psychoacoustics. *The Journal of the Acoustical Society of America*, 49(2B):467–477, Feb 1971.
- [9] Sandra T Mace, Cynthia L Wagoner, David J Teachout, and Donald A Hodges. Genre identification of very brief musical excerpts. *Psychology of Music*, 40(1):112–128, 2012.
- [10] Stephen McAdams. Musical Timbre Perception. In Diana Deutsch, editor, *The Psychology of Music*, pages 35–67. Elsevier, 3rd edition edition, 2013.
- [11] Stephen McAdams, Suzanne Winsberg, Sophie Donnadieu, Geert De Soete, and Jochen Krimphoff. Perceptual scaling of synthesized musical timbres: Common dimensions, specificities, and latent subject classes. *Psychological Research*, 58(3):177–192, Dec 1995.
- [12] Brian Moore. *An Introduction to the Psychology of Hearing: Sixth Edition*, pages 284–286. BRILL, 2013.
- [13] Daniel Müllensiefen, Bruno Gingras, Jason Musil, and Lauren Stewart. The musicality of non-musicians: an index for assessing musical sophistication in the general population. *PloS one*, 9(2):e89642, 2014.
- [14] Joy E Ollen. *A criterion-related validity test of selected indicators of musical sophistication using expert ratings*. PhD thesis, The Ohio State University, 2006.
- [15] Geoffroy Peeters, Bruno L. Giordano, Patrick Susini, Nicolas Misdariis, and Stephen McAdams. The timbre toolbox: Extracting audio descriptors from musical signals. *The Journal of the Acoustical Society of America*, 130(5):2902–2916, 2011.
- [16] Jean-Claude Risset and David L Wessel. Exploration of timbre by analysis and synthesis. In Diana Deutsch, editor, *The Psychology of Music*, pages 113–169. Elsevier, 1999.
- [17] Ken Robinson and Roy D. Patterson. The Duration Required to Identify the Instrument, the Octave, or the Pitch Chroma of a Musical Note. *Music Perception: An Interdisciplinary Journal*, 13(1):1–15, October 1995.
- [18] Ken Robinson and Roy D. Patterson. The stimulus duration required to identify vowels, their octave, and their pitch chroma. *The Journal of the Acoustical Society of America*, 98(4):1858–1865, October 1995.
- [19] E. L. Saldanha and John F. Corso. Timbre Cues and the Identification of Musical Instruments. *The Journal of the Acoustical Society of America*, 36(11):2021–2026, Nov 1964.
- [20] Noam R. Shabtai, Gottfried Behler, Michael Vorländer, and Stefan Weinzierl. Generation and analysis of an acoustic radiation pattern database for forty-one musical instruments. *The Journal of the Acoustical Society of America*, 141(2):1246–1256, Feb 2017.
- [21] Clara Suied, Trevor R. Agus, Simon J. Thorpe, Nima Mesgarani, and Daniel Pressnitzer. Auditory gist: Recognition of very short sounds from timbre cues.

The Journal of the Acoustical Society of America, 135(3):1380–1391, March 2014.

- [22] Clara Suied, Patrick Susini, Stephen McAdams, and Roy D. Patterson. Why are natural sounds detected faster than pips? *The Journal of the Acoustical Society of America*, 127(3):EL105–EL110, March 2010.
- [23] Stefan Weinzierl, Michael Vorländer, Gottfried Behler, Fabian Brinkmann, Henrik von Coler, Erik Detzner, Johannes Krämer, Alexander Lindau, Martin Pollow, Frank Schulz, and Noam R. Shabtai. A Database of Anechoic Microphone Array Measurements of Musical Instruments, Apr 2017. Available at <http://dx.doi.org/10.14279/depositonce-5861.2>.

FRAME-LEVEL INSTRUMENT RECOGNITION BY TIMBRE AND PITCH

Yun-Ning Hung and Yi-Hsuan Yang

Research Center for IT Innovation, Academia Sinica, Taipei, Taiwan
{biboamy, yang}@citi.sinica.edu.tw

ABSTRACT

Instrument recognition is a fundamental task in music information retrieval, yet little has been done to predict the presence of instruments in multi-instrument music for each time frame. This task is important for not only automatic transcription but also many retrieval problems. In this paper, we use the newly released MusicNet dataset to study this front, by building and evaluating a convolutional neural network for making frame-level instrument prediction. We consider it as a multi-label classification problem for each frame and use frame-level annotations as the supervisory signal in training the network. Moreover, we experiment with different ways to incorporate pitch information to our model, with the premise that doing so informs the model the notes that are active per frame, and also encourages the model to learn relative rates of energy buildup in the harmonic partials of different instruments. Experiments show salient performance improvement over baseline methods. We also report an analysis probing how pitch information helps the instrument prediction task. Code and experiment details can be found at <https://biboamy.github.io/instrument-recognition/>.

1. INTRODUCTION

Progress in pattern recognition problems usually depends highly on the availability of high-quality labeled data for model training. For example, in computer vision, the release of the ImageNet dataset [11], along with advances in algorithms for training deep neural networks [26], has fueled significant progress in *image-level* object recognition. The subsequent availability of other datasets, such as the COCO dataset [30], provide bounding boxes or even *pixel-level* annotations of objects that appear in an image, facilitating research on localizing objects in an image, semantic segmentation, and instance segmentation [30]. Such a move from image-level to pixel-level prediction opens up many new exciting applications in computer vision [16].

Analogously, for many music-related applications, it is desirable to have not only *clip-level* but also *frame-level* predictions. For example, expert users such as music composers may want to search for music with certain attributes

and require a system to return not only a list of songs but also indicate the time intervals of the songs that have those attributes [3]. Frame-level predictions of music tags can be used for visualization and music understanding [31,45]. In automatic music transcription, we want to know the musical notes that are active per frame as well as figure out the instrument that plays each note [13]. Vocal detection [40] and guitar solo detection [36] are another two examples that requires frame-level predictions.

Many of the aforementioned applications are related to the classification of sound sources, or instrument classification. However, as labeling the presence of instruments in multi-instrument music for each time frame is labor-intensive and time-consuming, most existing work on instrument classification uses either datasets of solo instrument recordings (e.g., the ParisTech dataset [24]), or datasets with only clip- or excerpt-level annotations (e.g., the IRMAS dataset [7]). While it is still possible to train a model that performs frame-level instrument prediction from these datasets, it is difficult to evaluate the result due to the absence of frame-level annotations.¹ As a result, to date little work has been done to specifically study frame-level instrument recognition, to the best of our knowledge (see Section 2 for a brief literature survey).

The goal of this paper is to present such a study, by taking advantage of a recently released dataset called MusicNet [44]. The dataset contains 330 freely-licensed classical music recordings by 10 composers, written for 11 instruments, along with over 1 million annotated labels indicating the precise time of each note in every recording and the instrument that plays each note. Using the *pitch* labels available in this dataset, Thickstun *et al.* [43] built a convolutional neural network (CNN) model that establishes a new state-of-the-art in multi-pitch estimation. We propose that the frame-level *instrument* labels provided by the dataset also represent a *valuable* information source. And, we try to realize this potential by using the data to train and evaluate a frame-level instrument recognition model.

Specifically, we formulate the problem as a multi-label classification problem for each frame and use frame-level annotations as the supervisory signal in training a CNN model with three residual blocks [21]. The model learns



© Yun-Ning Hung and Yi-Hsuan Yang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: Yun-Ning Hung and Yi-Hsuan Yang. “Frame-level Instrument Recognition by Timbre and Pitch”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ Moreover, these datasets may not provide high-quality labeled data for frame-level instrument prediction. To name a few reasons: the ParisTech dataset [24] contains only instrument solos and therefore misses the complexity seen in multi-instrument music; the IRMAS dataset [7] labels only the “predominant” instrument(s) rather than all the active instruments in each excerpt; moreover, an instrument may not be always active throughout an excerpt.

to predict instruments from a spectral representation of audio signals provided by the constant-Q transform (CQT) (see Section 4.1 for details). Moreover, as another technical contribution, we investigate several ways to incorporate pitch information to the instrument recognition model (Sections 4.2), with the premise that doing so informs the model the notes that are active per frame, and also encourages the model to learn the energy distribution of partials (i.e., fundamental frequency and overtones) of different instruments [2, 4, 14, 15]. We experiment with using either the ground truth pitch labels from MusicNet, or the pitch estimates provided by the CNN model of Thickstun *et al.* [43] (which is open-source). Although the use of pitch features for music classification is not new, to our knowledge few attempts have been made to jointly consider timbre and pitch features in a deep neural network model. We present in Section 5 the experimental results and analyze whether and how pitch-aware models outperform baseline models that take only CQT as the input.

2. RELATED WORK

A great many approaches have been proposed for (clip-level) instrument recognition. Traditional approaches used domain knowledge to engineer audio feature extraction algorithms and fed the features to classifiers such as support vector machine [25, 32]. For example, Diment *et al.* [12] combined Mel-frequency cepstral coefficients (MFCCs) and phase-related features and trained a Gaussian mixture model. Using the instrument solo recordings from the RWC dataset [17], they achieved 96.0%, 84.9%, 70.7% accuracy in classifying 4, 9, 22 instruments, respectively. Yu *et al.* [47] used sparse coding for feature extraction and support vector machine for classifier training, obtaining 96% accuracy in 10-instrument classification for the solo recordings in the ParisTech dataset [24]. Recently, Yip and Bittner [46] made open-source a solo instrument classifier that uses MFCCs in tandem with random forests to achieve 96% frame-level test accuracy in 18-instrument classification using solo recordings from the MedleyDB multi-track dataset [5]. Recognizing instruments in multi-instrument music has been proven more challenging. For example, Yu *et al.* [47] achieved 66% F-score in 11-instrument recognition using a subset of the IRMAS dataset [7].

Deep learning has been increasingly used in more recent work. Deep architectures can “learn” features by training the feature extraction module and the classification module in an end-to-end manner [26], thereby leading to better accuracy than traditional approaches. For example, Li *et al.* [27] showed that feeding raw audio waveforms to a CNN achieves 72% (clip-level) F-micro score in discriminating 11 instruments in MedleyDB, which MFCCs and random forest only achieves 64%. Han *et al.* [19] trained a CNN to recognize predominant instrument in IRMAS and achieved 60% F-micro, which is about 20% higher than a non-deep learning baseline. Park *et al.* [35] combined multi-resolution recurrence plots and spectrogram with CNN to achieved 94% accuracy in 20-instrument classification using the UIOWA solo instrument dataset [18].

Number of instruments used	Number of clips		Pitch est. accuracy
	Train set	Test set	
0	3	0	—
1	172	5	62.9%
2	33	1	56.2%
3	95	4	60.5%
4	15	0	56.6%
6	2	0	49.6%

Table 1: The number of clips in the training and test sets of MusicNet [44], divided according to the number of instruments used (among the seven instruments we consider in our experiment) per clip (e.g., a piano trio uses 3 instruments). We also show the average frame-level multi-pitch estimation accuracy (using *mir_eval* [38]) achieved by the CNN model proposed by Thickstun *et al.* [43].

Due to the lack of frame-level instrument labels in many existing datasets, little work has focused on frame-level instrument recognition. The work presented by Schlüter for vocal detection [40] and by Pati and Lerch for guitar solo detection [36] are exceptions, but they each addressed one specific instrument, rather than general instruments. Liu and Yang [31] proposed to use clip-level annotations in a weakly-supervised setting to make frame-level predictions, but the model is for general tags. Moreover, due to the assumption that CNN can learn high-level features on its own, domain knowledge of music has not been much used in prior work on deep learning based instrument recognition, though there are some exceptions [33, 37].

Our work differentiates itself from the prior arts in two aspects. First, we focus on frame-level instrument recognition. Second, we explicitly employ the result of multi-pitch estimation [6, 43] as additional inputs to our CNN model, with a design that is motivated by the observation that instruments have different pitch range and have unique energy distributions in the partials [14].

3. DATASET

Training and evaluating a model for frame-level instrument recognition is possible due to the recent release of the MusicNet dataset [44]. It contains 330 freely-licensed music recordings by 10 composers with over 1 million annotated pitch and instrument labels on 34 hours of chamber music performances. Following [43], we use the pre-defined split of training and test sets, leading to 320 and 10 clips in the training and test sets, respectively. As there are only seven different instruments in the test set, we only consider the recognition of these seven instruments in our experiment. They are *Piano*, *Violin*, *Viola*, *Cello*, *Clarinet*, *Bassoon* and *Horn*. For the training set, we do not exclude the sounds from the instruments that are not on the list, but these instruments are not labeled. Different clips use different number of instruments. See Table 1 for some statistics. For convenience, each clip is divided into 3-second segments. We use these segments as the input to our model. We zero-pad (i.e., adding silence) the last seg-

ment of each clip so that it is also 3 seconds. Due to space limit, for details we refer readers to the MusicNet website (check reference [44] for the URL) and also our project website (see the abstract for the URL).

We note that the MedleyDB dataset [5] can also be used for frame-level instrument recognition, but we choose MusicNet for two reasons. First, MusicNet is more than three times larger than MedleyDB in terms of the total duration of the clips. Second, MusicNet has pitch labels for each instrument, while MedleyDB only annotates the melody line. However, as MusicNet contains only classical music and MedleyDB has more Pop and Rock songs, the two datasets feature fairly different instruments and future work can be done to consider them both.

4. INSTRUMENT RECOGNITION METHOD

4.1 Basic Network Architectures that Uses CQT

To capture the timbral characteristics of each instrument, in our basic model we use CQT as the feature representation of music audio. CQT is a spectrographic representation that has a musically and perceptual motivated frequency scale [41]. We compute CQT by `librosa` [34], with sampling rate 44,100 and 512-sample window size. 88 frequency notes are extracted with 12 bins per octave, which forms a matrix $\mathbf{X} \in \mathcal{R}^{258 \times 88}$ as the input data, for each inputting 3-second audio segment.

We experiment with two baseline models. The first one is adapted from the CNN model proposed by Liu and Yang [31], which has been shown effective for music auto-tagging. Instead of using 6 feature maps as the input to the model as they did, we just use CQT as the input. Moreover, we use frame-level annotations as the supervisory signal in training the network, instead of training the model in a weakly-supervised fashion as they did. A batch normalization layer [23] is added after each convolution layer. Figure 1a shows the model architecture.

The second one is adapted from a more recent CNN model proposed by Chou *et al.* [10], which has been shown effective for large-scale sound event detection. Its design is special in two aspects. First, it uses 1D convolutions (along time) instead of 2D convolutions. While 2D convolutions analyze the input data as a chunk and convolve on both spectral and temporal dimensions, the 1D convolutions (along time) might better capture frequency and timbral information in each time frame [10, 29]. Second, it uses the so-called residual (Res) blocks [21, 22] to help the model learn deeper. Specifically, we employ three Res-blocks in between an *early* convolutional layer and a *late* convolutional layer. Each Res-block has three convolutional layers, so the network has a stack of 11 convolutional layers in total. We expect such a deep structure can learn well for a large-scale dataset such as MusicNet. Figure 1b shows its model architecture.

4.2 Adding Pitch

Although people usually expect neural networks can learn high-level feature such as pitch, onset and melody, our pi-

lot study shows that with the basic architecture the network still confuses some instruments (e.g., clarinet, bassoon and horn), and that onset frames for each instrument are not nicely located (see the second row of Figure 3). We propose to remedy this with a pitch-aware model that explicitly takes pitch as input, in a hope that doing so can amplify onset and timbre information. We experiment with several methods for inviting pitch to join the model.

4.2.1 Source of Frame-level Pitch Labels

We consider two ways of getting pitch labels in our experiment. One is using human-labeled ground truth pitch labels provided by MusicNet. However, in real-world applications, it is hard to get 100% correct pitch labels. Hence, we also use pitch estimation predicted by a state-of-the-art multi-pitch estimator proposed by Thickstun *et al.* [43]. The author proposed a translation-invariant network which combines traditional filterbank with a convolutional neural network. The model shares parameters in the log-frequency domain, which exploits the frequency invariance of music to reduce the number of model parameters and to avoid overfitting to the training data. The model reaches the top performance in the 2017 MIREX Multiple Fundamental Frequency Estimation evaluation [1]. The average pitch estimation accuracy, evaluated using `mir_eval` [38], is shown in Table 1.

4.2.2 Harmonic Series Feature

Figure 1c depicts the architecture of a proposed pitch-aware model. In this model, we aim to exploit the observation that the energy distribution of the partials constitutes a key factor in the perception of instrument timbre [14]. Being motivated by [6], we propose the *harmonic series feature* (HSF) to capture the harmonic structure of music notes, calculated as follows. We are given the input pitch estimate (or ground truth) $\mathbf{P}_0 \in \mathcal{R}^{258 \times 88}$, which is a matrix with the same size as the CQT matrix. The entries in \mathbf{P}_0 take the value of either 0 or 1 in the case of ground truth pitch labels, and the value in $[0, 1]$ in the case of estimated pitches. If the value of an entry is close to 1, we know that likely a music note with the fundamental frequency is active on that time frame.

First, we construct a *harmonic map* that shifts the active entries in \mathbf{P}_0 upwards by a multiple of the corresponding fundamental frequency (f_0). That is, the (t, f) -th entry in the resulting harmonic map $\mathbf{P}_n \in \mathcal{R}^{258 \times 88}$ is nonzero only if that frequency is $(n + 1)$ times larger than an active f_0 that frame, i.e., $f = f_0 \cdot (n + 1)$.

Then, a harmonic series feature up to the $(n + 1)$ -th harmonics,² denoted as $\mathbf{H}_n \in \mathcal{R}^{258 \times 88}$, is computed by an element-wise sum of $\mathbf{P}_0, \mathbf{P}_1, \dots$ up to \mathbf{P}_n , as illustrated in Figure 1c. In that follows, we also refer to \mathbf{H}_n as HSF- n .

When using HSF- n as input to the instrument recognition model, we concatenate CQT \mathbf{X} and \mathbf{H}_n along the channel dimension, to the effect that emphasizing the partials in the input audio. The resulting matrix is then used as the input to a CNN model depicted in Figure 1c. The CNN

² We note that the first harmonic is the fundamental frequency.

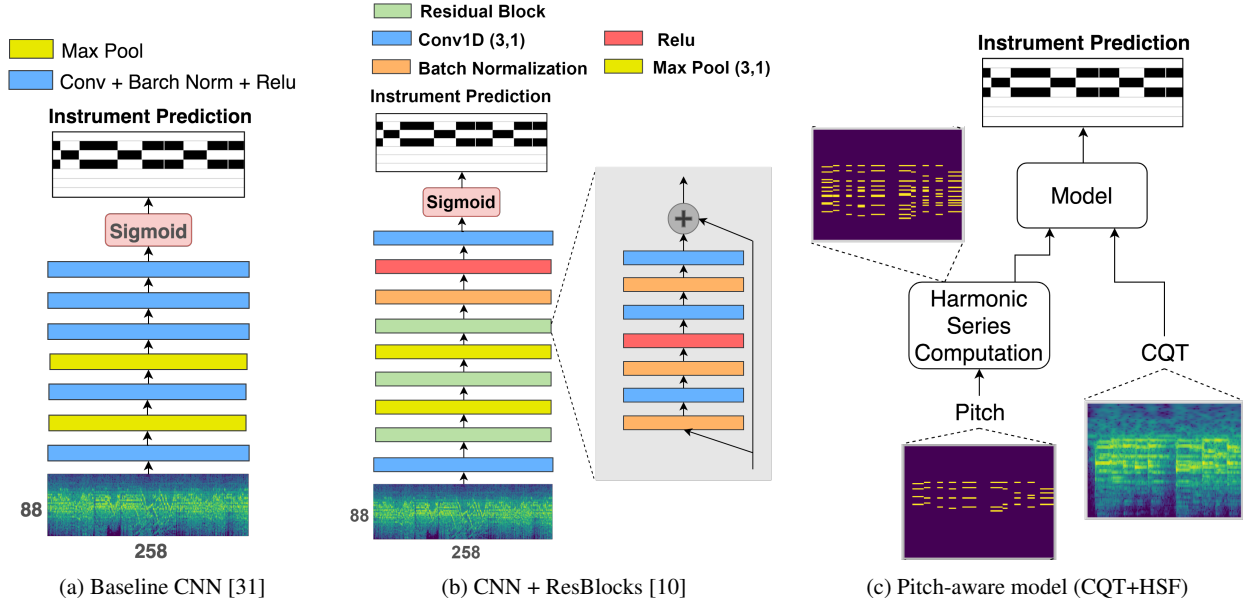


Figure 1: Three kinds of model structure used in this instrument recognition experiment.

model used here is also adapted from [10], using 1D convolutions, ResBlocks, and 11 convolutional layers in total. We call this model ‘*CQT+HSF-n*’ hereafter.

4.2.3 Other Ways of Using Pitch

We consider another two methods to use pitch information.

First, instead of stressing the overtones, the matrix \mathbf{P}_0 already contains information regarding which pitches are active per time frame. This information can be important because different instruments (e.g., violin, viola and cello) have different pitch ranges. Therefore, a simple way of taking pitch information into account is to concatenate \mathbf{P}_0 with the input CQT \mathbf{X} along the frequency dimension (which is fine since we use 1D convolutions), leading to a 258×176 matrix, and then feed it to the early convolutional layer. This method exploits pitch information right from the beginning of the feature learning process. We call it the ‘*CQT+pitch (F)*’ method for short.

Second, we can also concatenate \mathbf{P}_0 with the input CQT \mathbf{X} along the channel dimension, to allow the pitch information to directly influence the input CQT \mathbf{X} . It can tell us the pitch note and onset timing, which is critical in instrument recognition. We call this method ‘*CQT+pitch (C)*’.

4.3 Implementation Details

All the networks are trained using stochastic gradient descent (SGD) with momentum 0.9. The initial learning rate is set to 0.01. The weighted cross entropy, as defined below, is used as the cost function for model training:

$$l_n = -y_n [t_n \cdot \log \sigma(\hat{y}_n) + (1 - y_n) \cdot \log(1 - \sigma(\hat{y}_n))] , \quad (1)$$

where y_n and \hat{y}_n are the ground truth and predicted label for the n -th instrument per time frame, $\sigma(\cdot)$ is the sigmoid function to reduce the scale of \hat{y}_n to $[0, 1]$, and w_n is a weight computed to emphasize positive labels and

counter class imbalance between the instruments, based on the trick proposed in [39]. Code and model are built with the deep learning framework PyTorch.

Due to the final sigmoid layer, the output of the instrument recognition model is a continuous value in $[0, 1]$ for each instrument per frame, which can be interpreted as the likelihood of the presence for each instrument. To decide the existence of an instrument, we need to pick a threshold to binarize the result. Simply setting the threshold to 0.5 equally for all the instruments may not work well. Accordingly, we implement a simple threshold picking algorithm that selects the threshold (from 0.01, 0.02, ... to 0.99, in total 99 candidates) per instrument by maximizing the F1-score on the training set.

F1-score is the harmonic mean of precision and recall. In our experiments, we compute the F1-score independently (by concatenating the result for all the segments) for each instrument and then report the average result across instruments as the performance metric.

We do not implement any smoothing algorithm to post-process the recognition result, though this may help [28].

5. PERFORMANCE STUDY

The evaluation result is shown in Table 2. We first examine the result between two models without pitch information. From the first and second rows, we see that adding Res-blocks indeed leads to a more accurate model. Therefore, we also use Res-blocks for the pitch-aware models.

We then examine the result when we use ground truth pitch labels to inform the model. From the upper half of Table 2, pitch-aware models (i.e., CQT+HSF) indeed outperform the models that only use CQT. While the CQT-only model based on [10] attains 0.887 average F1-score, the best model CQT+HSF-3 reaches 0.933. Salient improvement is found for *Viola*, *Clarinet*, and *Bassoon*.

Pitch source	Method	Piano	Violin	Viola	Cello	Clarinet	Bassoon	Horn	Avg.
none	CQT only (based on [31])	0.972	0.934	0.798	0.909	0.854	0.816	0.770	0.865
	CQT only (based on [10])	0.982	0.956	0.830	0.933	0.894	0.822	0.789	0.887
ground-truth pitch	CQT+HSF-1	0.999	0.986	0.916	0.972	0.945	0.909	0.776	0.929
	CQT+HSF-2	0.997	0.984	0.912	0.968	0.941	0.906	0.799	0.930
	CQT+HSF-3	0.997	0.985	0.914	0.971	0.944	0.907	0.810	0.933
	CQT+HSF-4	0.997	0.986	0.909	0.969	0.944	0.904	0.815	0.932
	CQT+HSF-5	0.998	0.975	0.902	0.968	0.942	0.912	0.803	0.928
estimated pitch by [43]	CQT+HSF-1	0.983	0.955	0.841	0.935	0.901	0.822	0.793	0.890
	CQT+HSF-2	0.983	0.954	0.830	0.933	0.899	0.820	0.800	0.889
	CQT+HSF-3	0.983	0.955	0.829	0.934	0.903	0.818	0.805	0.890
	CQT+HSF-4	0.981	0.955	0.833	0.937	0.903	0.831	0.793	0.890
	CQT+HSF-5	0.984	0.956	0.835	0.935	0.915	0.839	0.805	0.896
	CQT+Pitch (F)	0.983	0.955	0.829	0.936	0.887	0.819	0.791	0.886
	CQT+Pitch (C)	0.982	0.958	0.819	0.921	0.898	0.827	0.794	0.886

Table 2: Recognition accuracy (in F1-score) of model with and without pitch information, using either ground truth pitches or estimated pitches. We use bold font to highlight the best result per instrument for the three groups of results.

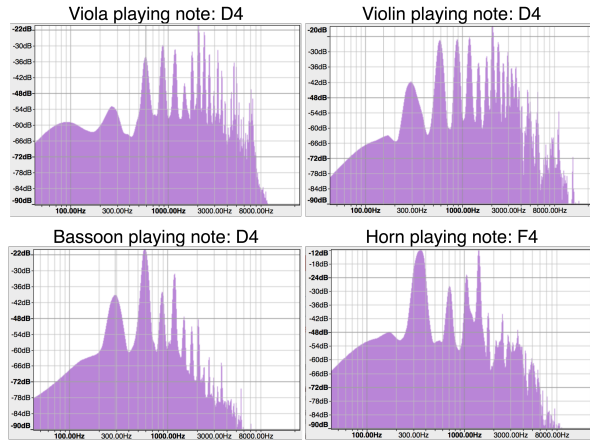


Figure 2: Harmonic spectrum of Viola (top left), Violin (top right), Bassoon (bottom left) and Horn (bottom right), created by the software Audacity [42] for real-life recordings of instruments playing a single note.

Moreover, a comparison among the pitch-aware models shows that different instruments seem to prefer different numbers of harmonics n . *Horn* and *Bassoon* achieve best F1-score with larger n (i.e., using more partials), while *Viola* and *Cello* achieves best F1-score with smaller n (using less partials). This is possibly because string instruments have similar amplitudes for the first five overtones, as Figure 2 exemplifies. Therefore, when more overtones are emphasized, it may be hard for the model to detect those trivial difference, and this in turn causes confusion between similar string instruments. In contrast, there is salient difference in the amplitudes of the first five overtones for *Horn* and *Bassoon*, making HSF-5 effective.

Figure 3 shows qualitative result demonstrating the prediction result for four clips in the test set. By comparing the result of the first two rows and the last row, we see that onset frames are clearly identified by the HSF-based model.

Furthermore, when adding HSF, it seems easier for a model to distinguish between similar instruments (e.g., violin versus viola). These examples show that adding HSF helps the model learn onset and timbre information.

Next, we examine the result when we use pitch estimation provided by the model of Thickstun *et al.* [43]. We know already from Table 1 that multi-pitch estimation is not perfect. Accordingly, as shown in the last part of Table 2, the performance of the pitch-aware models degrades, though still better than the model without pitch information. The best result is obtained by CQT+HSF-5, reaching 0.896 average F1-score. Except for *Violin*, CQT+HSF-5 outperforms CQT-only for all the instruments. We see salient improvement for *Viola*, *Clarinet*, *Bassoon* and *Horn*, for which the CQT-only model performs relatively worse. This shows that HSF helps highlight differences in the spectral patterns of the instruments.

Besides, similar to the case when using ground truth pitch labels, when using the estimated pitches, we see that *Viola* still prefers using fewer harmonic maps, whereas *Bassoon* and *Horn* prefer more. Given the observation that different instruments prefer different number of harmonics, it may be interesting to design an automatic way to dynamically decide the number of harmonic maps per frame, to further improve the result.

The fourth row of Figure 3 gives some result for CQT+HSF-5 based on estimated pitches. Compared to the result of CQT only (second row), we see that CQT+HSF-5 nicely reduces the confusion between *Violin* and *Viola* for the solo violin piece, and reinforces the onset timing for the string quartet piece.

Moving forward, we examine the result of the other two pitch-based methods, CQT+Pitch (F) and CQT+Pitch (C), using again estimated pitches. From the last two rows of Table 2, we see that these two methods do not perform better than even the second CQT-only baseline. As these two pitch-based methods take the pitch estimates directly as the model input, we conjecture that they are more sensitive

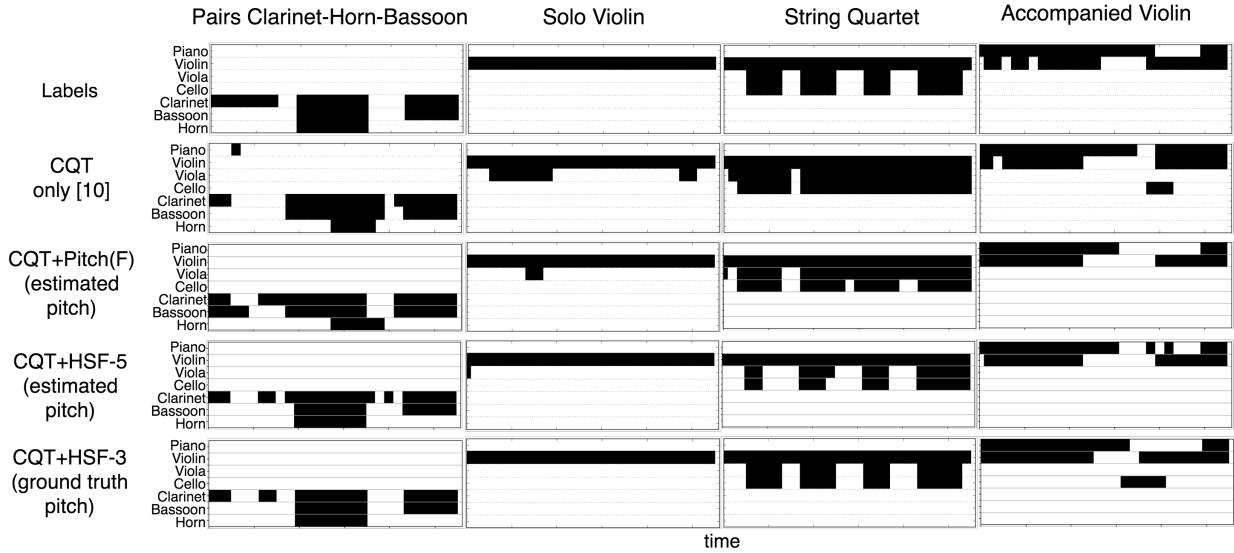


Figure 3: Prediction results of different methods for four test clips. The first row shows the ground truth frame-level instrument labels, where the horizontal axis denotes time. The other rows show the frame-level instrument recognition result for a model that only uses CQT (‘CQT only’; based on [10]) and three pitch-aware models that use either ground truth or estimated pitches. We use black shade to indicate the instrument(s) that are considered active in the labels or in the recognition result in each time frame.

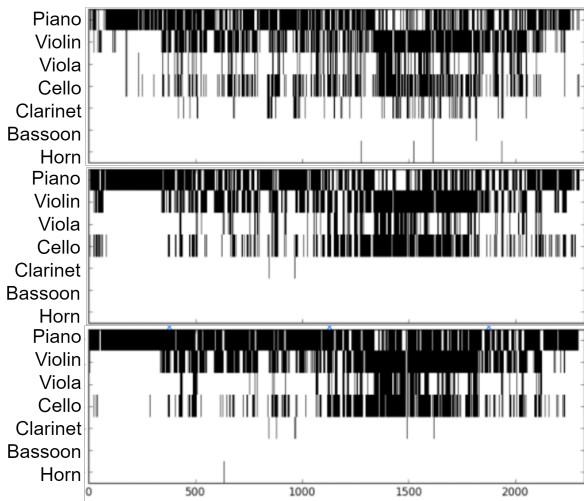


Figure 4: Frame-level instrument recognition result for a pop song, *Make You Feel My Love* by Adele, using the baseline CNN [31] (top), CNN + Res-blocks [10] (middle) and CQT+HSF-5 using estimated pitches (bottom).

to errors in multi-pitch estimation and accordingly cannot perform well. From the recognition result of the string quartet clip in the third row of Figure 3, we see that the CQT+Pitch (F) method cannot distinguish between similar instruments such as *Violin* and *Viola*. This suggests that HSF might be a better way to exploit pitch information.

Finally, out of curiosity, we test our models on a famous pop music (despite that our models are trained on classical music). Figure 4 shows the prediction result for the song *Make You Feel My Love* by Adele. It is encouraging to see

that our models correctly detect the *Piano* used throughout the song and the string instruments used in the middle solo part. Moreover, they correctly give almost zero estimate for the wind and brass instruments. Moreover, when using the Res-blocks, the prediction errors on *clarinet* are reduced. When using the pitch-aware model, the prediction errors on *Violin* and *Cello* at the beginning of the song are reduced. Besides, *Piano* timbre can also be strengthened when *Piano* and the strings play together at the bridge.

6. CONCLUSION

In this paper, we have proposed several methods for frame-level instrument recognition. Using CQT as the input feature, our model can achieve 88.7% average F1-score for recognizing seven instruments in the MusicNet dataset. Even better result can be obtained by the proposed pitch-aware models. Among the proposed methods, the HSF-based models achieve the best result, with average F1-score 89.6% and 93.3% respectively when using estimated and ground truth pitch information.

In future work, we will include MedleyDB to our training set to cover more instruments and music genres. We also like to explore joint learning frameworks and recurrent models (e.g., [8, 9, 20]) for better accuracy.

7. ACKNOWLEDGEMENT

This work was funded by a project with KKBOX Inc.

8. REFERENCES

- [1] MIREX multiple fundamental frequency estimation evaluation result, 2017. [Online] <http://>

- //www.music-ir.org/mirex/wiki/2017:Multiple_Fundamental_Frequency_Estimation_%26_Tracking_Results_-_MIREX_Dataset.
- [2] Giulio Agostini, Maurizio Longari, and Emanuele Polastri. Musical instrument timbres classification with spectral features. *EURASIP Journal on Applied Signal Processing*, 1:5–14, 2003.
 - [3] Kristina Andersen and Peter Knees. Conversations with expert users in music retrieval and research challenges for creative MIR. In *Proc. Int. Soc. Music Information Retrieval Conf.*, pages 122–128, 2016.
 - [4] Jayme Garcia Arnal Barbedo and George Tzanetakis. Musical instrument classification using individual partials. *IEEE Trans. Audio, Speech, and Language Processing*, 19(1):111–122, 2011.
 - [5] Rachel Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *Proc. Int. Soc. Music Information Retrieval Conf.*, 2014. [Online] <http://medleydb.weebly.com/>.
 - [6] Rachel M. Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan P. Bello. Deep salience representations for f_0 estimation in polyphonic music. In *Proc. Int. Soc. Music Information Retrieval Conf.*, pages 63–70, 2017.
 - [7] Juan J. Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera. A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. In *Proc. Int. Soc. Music Information Retrieval Conf.*, pages 559–564, 2012. [Online] <http://mtg.upf.edu/download/datasets/irmas/>.
 - [8] Ning Chen and Shijun Wang. High-level music descriptor extraction algorithm based on combination of multi-channel CNNs and LSTM. In *Proc. Int. Soc. Music Information Retrieval Conf.*, pages 509–514, 2017.
 - [9] Keunwoo Choi, Gyorgy Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 2017.
 - [10] Szu-Yu Chou, Jyh-Shing Jang, and Yi-Hsuan Yang. Learning to recognize transient sound events using attentional supervision. In *Proc. Int. Joint Conf. Artificial Intelligence*, 2018.
 - [11] Jia Deng et al. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. Conf. Computer Vision and Pattern Recognition*, 2009.
 - [12] Aleksandr Diment, Padmanabhan Rajan, Toni Heittola, and Tuomas Virtanen. Modified group delay feature for musical instrument recognition. In *Proc. Int. Symp. Computer Music Multidisciplinary Research*, 2013.
 - [13] Zhiyao Duan, Jinyu Han, and Bryan Pardo. Multi-pitch streaming of harmonic sound mixtures. *IEEE/ACM Trans. Audio, Speech, and Language Processing*, 22(1):138–150, 2014.
 - [14] Zhiyao Duan, Yungang Zhang, Changshui Zhang, and Zhenwei Shi. Unsupervised single-channel music source separation by average harmonic structure modeling. *IEEE Trans. Audio, Speech, and Language Processing*, 16(4):766 – 778, 2008.
 - [15] Slim Essid, Gaël Richard, and Bertrand David. Musical instrument recognition by pairwise classification strategies. *IEEE Trans. Audio, Speech, and Language Processing*, 14(4):1401–1412, 2006.
 - [16] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and José García Rodríguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.
 - [17] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC Music Database: Popular, classical and jazz music databases. In *Proc. Int. Society of Music Information Retrieval Conf.*, pages 287–288, 2002. [Online] <https://staff.aist.go.jp/m.goto/RWC-MDB/rwc-mdb-i.html>.
 - [18] Matt Hallaron et al. University of Iowa musical instrument samples. University of Iowa, 1997. [Online] <http://theremin.music.uiowa.edu/MIS.html>.
 - [19] Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Trans. Audio, Speech, and Language Processing*, 25(1):208 – 221, 2017.
 - [20] Curtis Hawthorne, Erich Elsen adn Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. *Proc. Int. Soc. Music Information Retrieval Conf.*, 2018.
 - [21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2016.
 - [22] Shawn Hershey et al. CNN architectures for large-scale audio classification. In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2017.
 - [23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. Int. Conf. Machine Learning*, pages 448–456, 2015.

- [24] Cyril Joder, Slim Essid, and Gaël Richard. Temporal integration for audio classification with application to musical instrument classification. *IEEE Trans. Audio, Speech and Language Processing*, 17(1):174–186, 2009.
- [25] Tetsuro Kitahara, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Instrument identification in polyphonic music: Feature weighting with mixed sounds, pitch-dependent timbre modeling, and use of musical context. In *Proc. Int. Soc. Music Information Retrieval Conf.*, pages 558–563, 2005.
- [26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [27] Peter Li, Jiyuan Qian, and Tian Wang. Automatic instrument recognition in polyphonic music using convolutional neural networks. *arXiv preprint arXiv:1511.05520*, 2015.
- [28] Dawen Liang, Matthew D. Hoffman, and Gautham J. Mysore. A generative product-of-filters model of audio. In *Proc. Int. Conf. Learning Representations*, 2014.
- [29] Hyungui Lim, Jeongsoo Park, Kyogu Lee, and Yoonchang Han. Rare sound event detection using 1D convolutional recurrent neural networks. In *Proc. Int. Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.
- [30] Tsung-Yi Lin et al. Microsoft COCO: Common objects in context. In *Proc. European Conf. Computer Vision*, pages 740–755, 2014.
- [31] Jen-Yu Liu and Yi-Hsuan Yang. Event localization in music auto-tagging. *Proc. ACM Int. Conf. Multimedia*, pages 1048–1057, 2016.
- [32] Arie Livshin and Xavier Rodet. The significance of the non-harmonic “noise” versus the harmonic series for musical instrument recognition. In *Proc. Int. Soc. Music Information Retrieval Conf.*, 2006.
- [33] Vincent Lostanlen and Carmine-Emanuele Cella. Deep convolutional networks on the pitch spiral for musical instrument recognition. In *Proc. Int. Soc. Music Information Retrieval Conf.*, pages 612–618, 2016.
- [34] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proc. Python in Science Conf.*, pages 18–25, 2015. [Online] <https://librosa.github.io/librosa/>.
- [35] Taejin Park and Taejin Lee. Musical instrument sound classification with deep convolutional neural network using feature fusion approach. *arXiv preprint arXiv:1512.07370*, 2015.
- [36] Kumar Ashis Pati and Alexander Lerch. A dataset and method for electric guitar solo detection in rock music. In *Proc. Audio Engineering Soc. Conf.*, 2017.
- [37] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *Proc. Int. Workshop on Content-based Multimedia Indexing*, 2016.
- [38] Colin Raffel, Brian Mcfee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. mir_eval: a transparent implementation of common mir metrics. In *Proc. Int. Soc. Music Information Retrieval Conf.*, 2014. [Online] https://github.com/craffel/mir_eval.
- [39] Rif A. Saurous et al. The story of audioset, 2017. [Online] http://www.cs.tut.fi/sgn/arg/dcase2017/documents/workshop_presentations/the_story_of_audioset.pdf.
- [40] Jan Schlüter. Learning to pinpoint singing voice from weakly labeled examples. In *Proc. Int. Soc. Music Information Retrieval Conf.*, 2016.
- [41] Christian Schoerhuber and Anssi Klapuri. Constant-Q transform toolbox for music processing. In *Proc. Sound and Music Computing Conf.*, 2010.
- [42] Audacity Team. Audacity. <https://www.audacityteam.org/>, 1999-2018.
- [43] John Thickstun, Zaid Harchaoui, Dean P. Foster, and Sham M. Kakade. Invariances and data augmentation for supervised music transcription. In *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 2018. [Online] <https://github.com/jthickstun/thickstun2018invariances>.
- [44] John Thickstun, Zaid Harchaoui, and Sham M. Kakade. Learning features of music from scratch. In *Proc. Int. Conf. Learning Representations*, 2017. [Online] <https://homes.cs.washington.edu/~thickstn/musicnet.html>.
- [45] Ju-Chiang Wang, Hsin-Min Wang, and Shyh-Kang Jeng. Playing with tagging: A real-time tagging music player. In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pages 77–80, 2014.
- [46] Hanna Yip and Rachel M. Bittner. An accurate open-source solo musical instrument classifier. In *Proc. Int. Soc. Music Information Retrieval Conf., Late-Breaking Demo Paper*, 2017.
- [47] Li-Fan Yu, Li Su, and Yi-Hsuan Yang. Sparse cepstral codes and power scale for instrument identification. In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pages 7460–7464, 2014.

Session B

Generation, visual

INTERACTIVE ARRANGEMENT OF CHORDS AND MELODIES BASED ON A TREE-STRUCTURED GENERATIVE MODEL

Hiroaki Tsushima Eita Nakamura Katsutoshi Itoyama Kazuyoshi Yoshii

Graduate School of Informatics, Kyoto University, Japan

{tsushima, enakamura}@sap.ist.i.kyoto-u.ac.jp, {itoyama, yoshii}@kuis.kyoto-u.ac.jp

ABSTRACT

We describe an interactive music composition system that assists a user in refining chords and melodies by generating chords for melodies (harmonization) and vice versa (melodization). Since these two tasks have been dealt with independently, it is difficult to jointly estimate chords and melodies that are optimal in both tasks. Another problem is developing an interactive GUI that enables a user to partially update chords and melodies by considering the latent tree structure of music. To solve these problems, we propose a hierarchical generative model consisting of (1) a probabilistic context-free grammar (PCFG) for chord symbols, (2) a metrical Markov model for chord boundaries, (3) a Markov model for melody pitches, and (4) a metrical Markov model for melody onsets. The harmonic functions (syntactic roles) and repetitive structure of chords are learned by the PCFG. Any variables specified by a user can be optimized or sampled in a principled manner according to a unified posterior distribution. For improved melodization, a long short-term memory (LSTM) network can also be used. The subjective experimental result showed the effectiveness of the proposed system.

1. INTRODUCTION

Music composition is a highly intelligent task that has been considered to be done only by musically trained people. To help musically untrained people create their own musical pieces, automatic music composition has actively been studied (*e.g.*, [4, 8, 19, 31]). While conventional studies have aimed at full automation of music composition, in the process of music composition, melodies (sequences of musical notes) and chord sequences are partially and incrementally refined by trial and error until the resulting musical piece has musically appropriate structure. Our aim is to develop an interactive arrangement system that can assist unskillful people to take such a process for reflecting their preference in creating melodies and chord sequences.

It is non-trivial to reflect user’s preference to a musical piece in a consistent and unified framework of statistical

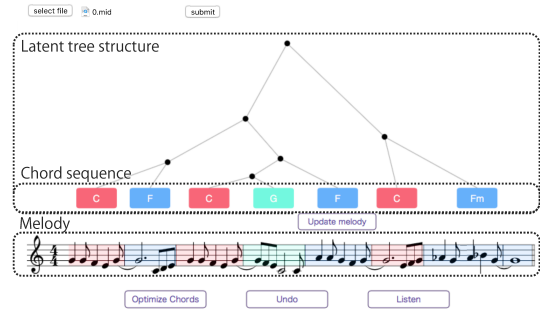


Figure 1: Our interactive music arrangement system based on a tree-structured generative model.

modeling. This problem is hard to solve especially when a black-box method (*e.g.*, neural end-to-end learning) is used for music generation. To incrementally refine a musical piece, one may iteratively use a harmonization method for generating a chord sequence from a melody [4, 19, 24, 28] and a melodization method for generating a melody from a chord sequence [3, 7, 8, 15, 22, 30, 31]. This approach, however, cannot enable a user to partially and incrementally refine melodies and chords in consideration of the optimality of the whole musical piece because each task has a unique evaluation criterion.

Since music is typically well-characterized by chords and melodies, it is important to be aware of complicated structures within and between chords and melodies. when composing a musical piece. To generate a musically appropriate sequence of chords, the harmonic functions of chords, which typically consist of three categories, *i.e.*, tonic (T), dominant (D), and subdominant (SD), should be considered because such functions represent syntactic roles in the same way as parts of speech in written texts. In addition, a sequence of harmonic functions of chords has a tree structure [21, 26]. For example, a chord sequence (C, Dm, G, Am, C, F, G, C) can be interpreted as (((T, SD), (D, T)), ((T, SD), (D, T))), where subtrees such as (T, SD), (D, T), and ((T, SD), (D, T)) appear repeatedly in a hierarchical manner. Therefore, it is desirable to consider such the hierarchical tree structure of chord sequences when we computationally help people to create a new music.

In this paper we propose an interactive music arrangement system that enables musically untrained users to create a melody and a chord sequence (Fig. 1). To partially and incrementally refine the piece, users can choose several types of operations that are often exploited by musically trained people. Specifically, the entire chord se-



© Hiroaki Tsushima, Katsutoshi Itoyama, Eita Nakamura, Kazuyoshi Yoshii. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Hiroaki Tsushima, Katsutoshi Itoyama, Eita Nakamura, Kazuyoshi Yoshii. “An Interactive System for Generating Chords and Melodies Based on a Tree-Structured Model”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

quence and the corresponding tree structure can be refined jointly for a melody; the onset time of a specified chord can be refined; two adjacent chords forming a subtree can be merged into a single chord or a chord can be split into two chords; and melody notes in the region of a specified chord can be refined. All a user needs to do is to specify where to update the piece and it is not necessary to manually edit individual musical elements.

To optimize a chord sequence and a melody in a unified criterion, we propose a tree-structured hierarchical generative model that consists of (i) a probabilistic context-free grammar (PCFG) generating chord symbols [28], (ii) a metrical Markov model generating chord rhythms, and (iii) a Markov model generating melody pitches conditionally on the chord sequence, and (iv) a metrical Markov model generating melody rhythms (Fig. 2). The rule probabilities of the PCFG are learned from chord sequences, with the expectation that the syntactic roles of chords are captured by the non-terminal symbols [29]. The other models are also learned from chord and/or note sequences. To improve the melodization process, a long short-term memory (LSTM) network can be used instead of the Markov models (iii) and (iv) for capturing the long-term characteristic of a melody. Using the generative model trained in advance, we can estimate any “missing” variables, *i.e.*, an unpleasant part of chords or musical notes specified by the user, in a statistical manner.

The major contribution of this study is the realization of a directability-aware music composition/arrangement system based on a unified probabilistic model. This system provides a user with an easy-to-use GUI that shows other possibilities for an unpleasant part of the piece and all operations on the GUI are implemented as posterior inference based on the probabilistic model. Our contribution lies in the marriage of AI and human creativity.

2. RELATED WORK

This section reviews related studies on automatic harmonization and melodization.

2.1 Automatic Harmonization

Many studies have been conducted for automatic harmonization for given melodies. Some studies aim to generate a sequence of chord symbols (as in this paper), and others aim to generate several (typically four) voices of musical notes. In the former type of research, Chuan and Chew [4] proposed a method consisting of three processes: selecting musical notes that might form chords from given melodies with a support vector machine (SVM), constructing triad chords from the selected notes, and generating chord progressions by using a rule-based method. Simon et al. [24] proposed a commercial system *MySong* based on hidden Markov models (HMMs) with Markovian chord transitions. Raczynski et al. [20] proposed similar Markov models in which chords are conditioned by melodies and time-varying keys. Tsushima et al. [28] proposed a harmonization method that considers the hierarchical repetitive structure of sequences of chord symbols obtained by

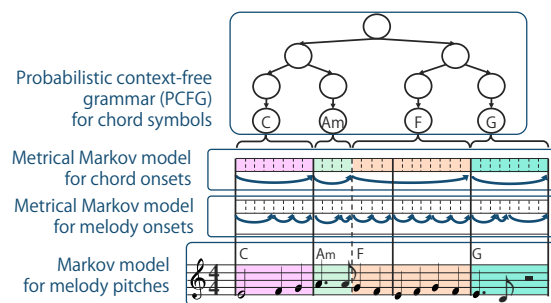


Figure 2: A tree-structured hierarchical generative model for chord symbols and melodies.

PCFGs and pitch transitions conditioned by chord symbols with Markov models. De Prisco et al. [19] proposed a harmonization method for only a base line of the input with a distinctive network that models the dependencies among bass notes, the previous chord, and the current chord.

In the latter type of research, Ebcioglu [6] proposed a rule-based method for generating four-part chorales in Bach’s style. Several methods of using variants of genetic algorithms (GAs) based on music theories have also been proposed [17, 18, 27]. Allan and Williams [2] proposed an HMM-based method that represents chords as hidden states and musical notes as observed outputs. A hidden semi-Markov model (HSMM) [11] has been used for explicitly representing the durations of chords. Paiement et al. [16] proposed a hierarchical tree-structured model that describes chord movements from the viewpoint of hierarchical time scales by dividing the notations of chords. To generate highly convincing four-part chorales, a deep recurrent neural network has also been used for capturing the long-term characteristic of a melody and a harmony [12].

2.2 Automatic Melodization

There have been many studies on automatic melodization [3, 8, 15, 22, 30, 31]. Fukayama et al. [8] developed a system named *Orpheus* that generates a melody for a given lyric in a way that the prosody of the lyric matches the dynamics of the melody. Roig et al. [22] proposed a method of generating a monophonic melody by using a probabilistic model of rhythm patterns and pitch contours.

Recent studies have applied deep learning techniques. In *Magenta* project [30], for example, recurrent neural networks (RNNs) are used for learning long-term dependency of music. Yang et al. [31] proposed a novel method for generating diverse monophonic melodies by combining a generative adversarial network (GAN) with a convolutional neural network (CNN). To generate diverse melodies, Morgen [15] proposed adversarial training of an RNN that works on continuous sequential data. The method based on a restricted Boltzmann machine (RBM) conditioned on RNNs that models temporal dependency has been proposed to generate polyphonic music [3]. In addition, Eck et al. [7] have proposed an LSTM-based method for generating both melodies and chords by capturing the characteristic of note-by-note transitions and the mutual dependency between musical notes and chord symbols.

3. USER INTERFACE

The proposed system, which is implemented as a web service based on HTML5, enables a user to incrementally refine a chord sequence and a melody on a GUI (Fig. 1). To use a system, a user is asked to upload a melody of eight bars. The system then estimates a chord sequence that harmonizes with the melody. The chord onsets are located at the bar lines. Supported arrangement operations are:

- **Updating the chord symbols:** The chord symbols and the latent tree structure behind the chord symbols are jointly optimized for the current melody.
- **Updating a chord onset:** One of the chord onsets (boundaries) specified by a user is optimized.
- **Splitting a chord:** One of the chords specified by a user is split into two adjacent chords.
- **Merging chords:** Two adjacent chords that form a subtree are merged into a single chord.
- **Updating the melody:** Melody notes in the region of a chord specified by a user are updated while keeping consistency with neighboring measures.

4. PROBABILISTIC MODELING

This section explains a unified probabilistic model that represents the hierarchical generative process of a chord sequence and a melody. The proposed model consists of four sub-models, which are trained independently.

4.1 Mathematical Notation

We assume that chord and melody onsets are on the 16th-note-level grid. Let L be the number of measures of a musical piece ($L = 8$ in this paper) and $T = 16L$ be the total number of time units. A sequence of chord symbols and that of chord onsets are denoted by $\mathbf{z} = \{z_n\}_{n=1}^N$ and $\phi = \{\phi_n\}_{n=1}^N$, respectively, where N is the number of chords and ϕ_n takes an integer in $[0, T)$. Similarly, a sequence of melody pitches and that of melody onsets in the region of chord z_n is denoted by $\mathbf{p}_n = \{p_{n,i}\}_{i=1}^{I_n}$ and $\psi_n = \{\psi_{n,i}\}_{i=1}^{I_n}$, respectively, where I_n is the number of musical notes in that time span, $p_{n,i}$ is a MIDI note number from 32 to 93, and $\psi_{n,i}$ takes an integer in $[\phi_n, \phi_{n+1})$. The whole melody is denoted by $\mathbf{p} = \{\mathbf{p}_n\}_{n=1}^N$ and $\psi = \{\psi_n\}_{n=1}^N$, where $I = \sum_{n=1}^N I_n$ is the number of melody notes.

Let \mathbf{t} be a latent tree that derives \mathbf{z} according to a PCFG and $t_{m:n}$ be an *inside* part (subtree) of \mathbf{t} that derives $z_{m:n}$. Thus $\mathbf{t} = t_{1:N}$. We often use $t_{m:n}$ to indicate the root node of the subtree for simplicity. Let $t_{\neg m:n}$ be an *outside* part of \mathbf{t} that derives $z_{1:m-1}$, $t_{m:n}$, and $z_{n+1:N}$.

4.2 Model Formulation

We formulate a unified probabilistic model that represents the generative process of a latent tree \mathbf{t} , chord symbols \mathbf{z} , chord onsets ϕ , melody pitches \mathbf{p} , and melody onsets ψ .

4.2.1 Probabilistic Context-Free Grammar for \mathbf{t} and \mathbf{z}

A derivation tree \mathbf{t} and chord symbols \mathbf{z} are generated in this order according to a PCFG $G = (V, \Sigma, R, S)$, defined

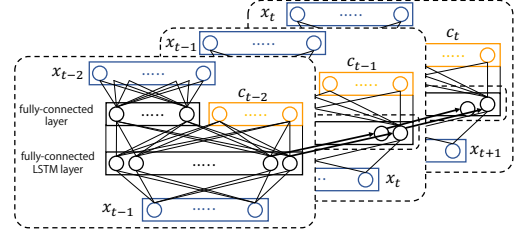


Figure 3: Configuration of the LSTM network

by a set of non-terminal symbols V that are expected to represent the hierarchical structure and syntactic roles of chords, a set of terminal symbols (chord symbols) Σ , a set of rule probabilities R , and a start symbol S (a non-terminal symbol located on the root of a syntax tree). There are three types of rule probabilities. $\theta_{A \rightarrow BC}$ is the probability that a non-terminal symbol $A \in V$ branches to non-terminal symbols $B \in V$ and $C \in V$. $\eta_{A \rightarrow \alpha}$ is the probability that $A \in V$ emits terminal symbol $\alpha \in \Sigma$. A non-terminal symbol $A \in V$ emits a terminal symbol with a probability of $0 < \lambda_A < 1$ and otherwise it branches. These probabilities are normalized as follows:

$$\sum_{B, C \in V} \theta_{A \rightarrow BC} = 1, \quad \sum_{\alpha \in \Sigma} \eta_{A \rightarrow \alpha} = 1. \quad (1)$$

We let $\theta_A = \{\theta_{A \rightarrow BC}\}_{B, C \in V}$ and $\eta_A = \{\eta_{A \rightarrow \alpha}\}_{\alpha \in \Sigma}$.

4.2.2 Metrical Markov Models for ϕ and ψ

The metrical Markov model for chord onsets ϕ on the regular 16th-note-level grid is defined by

$$p(\phi_n | \phi_{n-1}) = \pi_{\phi_{n-1} \bmod 16, \phi_n - \phi_{n-1}}, \quad (2)$$

where $\pi_{a,b}$ indicates the probability that a chord starting at the a -th position in a measure ($0 \leq a < 16$) continues for the duration of b time units ($0 < b \leq T$).

A similar model for melody onsets ψ is defined by

$$p(\psi_{n,1} | \psi_{n-1, I_{n-1}}) = \rho_{\psi_{n-1, I_{n-1}} \bmod 16, \psi_{n,1} - \psi_{n-1, I_{n-1}}}, \\ p(\psi_{n,i} | \psi_{n,i-1}) = \rho_{\psi_{n,i-1} \bmod 16, \psi_{n,i} - \psi_{n,i-1}} \quad (1 < i), \quad (3)$$

where $\rho_{a,b}$ indicates the probability that a musical note starts at the a -th position in a measure ($0 \leq a < 16$) and continues for the duration of b time units ($0 < b \leq T$).

4.2.3 Markov Model for \mathbf{p} Conditioned on \mathbf{z}

The Markov model for melody pitches \mathbf{p} conditioned by a chord sequence given by \mathbf{z} is defined by

$$p(p_{n,1} | p_{n-1, I_{n-1}}, z_n) = \tau_{p_{n-1, I_{n-1}}, p_{n,1}}^{z_n}, \quad (4)$$

$$p(p_{n,i} | p_{n,i-1}, z_n) = \tau_{p_{n,i-1}, p_{n,i}}^{z_n} \quad (2 \leq i \leq I_n), \quad (5)$$

where $\tau_{a,b}^c$ is the transition probability from pitch a to pitch b under chord symbol c .

4.2.4 Bayesian Integration of Four Sub-models

Letting $\Omega = \{\mathbf{t}, \mathbf{z}, \phi, \mathbf{p}, \psi\}$ be a set of the external random variables and $\Theta = \{\theta, \eta, \lambda, \pi, \rho, \tau\}$ be a set of the model parameters, the unified model is given by

$$p(\Omega, \Theta) = p(\mathbf{t}, \mathbf{z} | \theta, \eta, \lambda) p(\phi | \pi) p(\psi | \tau) p(\mathbf{p} | \mathbf{z}) p(\Theta), \quad (6)$$

where $p(\Theta) = p(\theta)p(\eta)p(\lambda)p(\pi)p(\rho)p(\tau)$ is a prior distribution over Θ . To make Bayesian inference tractable,

we use conjugate Dirichlet and beta priors as follows:

$$\theta_A \sim \text{Dir}(\xi_A), \eta_A \sim \text{Dir}(\zeta_A), \lambda_A \sim \text{Beta}(\iota_A), \quad (7)$$

$$\pi_a \sim \text{Dir}(\beta_a), \rho_a \sim \text{Dir}(\gamma_a), \tau_a^c \sim \text{Dir}(\delta_a^c), \quad (8)$$

where $\xi_A, \zeta_A, \iota_A, \beta_a, \gamma_a$, and δ_a^c are hyperparameters.

4.2.5 LSTM Network for x Conditioned on c

In melody arrangement, we can also use an LSTM model that can learn complicated long-term dynamics of melodies. Let $x = \{x_t\}_{t=1}^T$ be another representation of the entire melody, where x_t takes a MIDI note number at the t -th position ($0 \leq t < T$) if the note onset is at that position and otherwise takes 0. Let $c = \{c_t\}_{t=1}^T$ be another representation of the entire chord sequence given by z and ϕ , where c_t indicates a chord symbol at the t -th position. Given a sequence of musical notes $x_{1:t} = \{x_i\}_{i=1}^t$ and that of chord symbols $c_{1:t} = \{c_i\}_{i=1}^t$, the LSTM model determines the probability of the next musical note given by $p(x_{t+1}|x_{1:t}, c_{1:t})$ (Fig. 3).

4.3 Model Training

Our goal is to obtain the maximum a posteriori (MAP) estimates of the model parameters $\Theta = \{\theta, \eta, \lambda, \pi, \rho, \tau\}$. To estimate the parameters θ, η , and λ of the PCFG from a chord sequence z (multiple sequences are used in practice) in an unsupervised manner, we use an inside-filtering-outside-sampling algorithm [13, 28] for generating samples from the true posterior distribution $p(\theta, \eta, \lambda, t|z)$. More specifically, the latent tree t and the parameters θ, η , and λ are alternately sampled from the conditional posterior distributions $p(t|\theta, \eta, \lambda, z)$ and $p(\theta, \eta, \lambda|t, z)$, respectively.

The parameters π, τ and ρ of the Markov models are learned independently. Given a sequence of chord onsets ϕ and a sequence of melody onsets ψ , the posterior distribution of π and that of ρ can be calculated, respectively, because of the conjugacy between the Dirichlet and categorical distributions. Similarly, given a sequence of melody pitches p associated with a chord sequence specified by z and ϕ , the posterior distribution of τ can be calculated. The LSTM network is also trained from the same data.

5. CHORD AND MELODY ARRANGEMENT

This section explains how to leverage the unified model described in Section 4 for implementing the five operations described in Section 3. Let $\Omega = \{t, z, \phi, p, \psi\}$ be a set of random variables. To estimate a “missing” part $\chi \subset \Omega$, we take a principled statistical approach based on the conditional posterior distribution $p(\chi|\Omega_{-\chi}, \Theta)$, where A_{-B} indicates a subset of A obtained by removing the elements of B from A . Note that full automatic music composition can be achieved by sampling Ω from $p(\Omega|\Theta)$.

5.1 Updating the Chord Symbols

When the melody pitches p are fixed, the chord symbols z and the latent tree t can be optimized by maximizing the conditional posterior distribution $p(t, z|p, \Theta)$. Since both t and z are latent variables in this operation, we extend the Viterbi algorithm to infer t and z from p . First, the inside

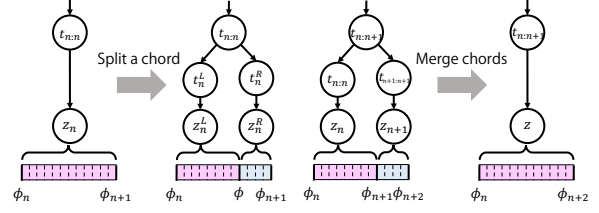


Figure 4: Split and merge operations.

probabilities are recursively calculated from the layer of terminal symbols z to the start symbol S according to

$$p_{n,n}^A = \lambda_A \max_{z \in \Sigma} \eta_{A \rightarrow z} p(p_n|z), \quad (9)$$

$$p_{n,n+k}^A = (1 - \lambda_A) \max_{\substack{B, C \in V \\ 1 \leq l \leq k}} \theta_{A \rightarrow BC} p_{n,n+l-1}^B p_{n+l,n+k}^C, \quad (10)$$

where $p(p_n|z_n)$ is the probability that a pitch subsequence p_n is generated conditionally on chord z_n :

$$p(p_n|z_n) = \prod_{i=1}^{I_n} p(p_{n,i}|p_{n,i-1}, z_n), \quad (11)$$

where $p_{n,0} = p_{n-1, I_{n-1}}$. The most likely t and z are obtained by recursively back-tracking the most likely paths from the start symbol S .

5.2 Updating a Chord Onset

When the melody pitches p and the melody onsets ψ are given and the chord symbols z are fixed, a chord onset ϕ_n can be optimized by maximizing the conditional posterior distribution given by

$$p(\phi_n|z, \phi_{-n}, p, \psi, \Theta) \propto p(p_{n-1}|z_{n-1}) p(p_n|z_n) p(\phi_n|\phi_{n-1}) p(\phi_{n+1}|\phi_n), \quad (12)$$

where ϕ_n is restricted such that $\psi_{n-1,1} \leq \phi \leq \psi_{n, I_n}$.

5.3 Splitting a Chord and Merging Chords

The chord symbols z and the chord onsets ϕ can be locally refined by splitting a chord into adjacent chords or merging adjacent chords into another chord (Fig. 4). A subtree of t is updated accordingly. The split operation can be applied to any chord z_n while the merge operation is restricted to adjacent chords $z_{n:n+1}$ forming a subtree $t_{n:n+1}$.

A chord z_n associated with a non-terminal symbol $t_{n:n}$ is split at a 16th-note-level position ϕ into two new chords z_n^L and z_n^R associated with two new symbols t_n^L and t_n^R by maximizing the conditional posterior distribution given by $p(t_n^L, t_n^R, z_n^L, z_n^R, \phi|t_{n:n}, z_{-n}, \phi, p, \psi, \Theta)$. This operation makes a new subtree that has $t_{n:n}$ as its root node, derives t_n^L and t_n^R , and generates z_n^L and z_n^R . To do this, we use the extended Viterbi algorithm for estimating the most likely subtree from p_n . First, the inside probabilities are recursively calculated from the layer of terminal symbols z_n^L and z_n^R to the root node $t_{n:n}$ according to

$$\alpha_\phi^A = \lambda_A \max_{z \in \Sigma} \eta_{A \rightarrow z} p(p_n^L|z, \phi), \quad (13)$$

$$\beta_\phi^A = \lambda_A \max_{z \in \Sigma} \eta_{A \rightarrow z} p(p_n^R|z, \phi), \quad (14)$$

$$p_\phi^{t_{n:n}} = \max_{B, C \in V} \theta_{t_{n:n} \rightarrow BC} \alpha_\phi^B \beta_\phi^C p(\phi|\phi_n) p(\phi_{n+1}|\phi), \quad (15)$$

where \mathbf{p}_n^L and \mathbf{p}_n^R are the subsequences of pitches obtained by splitting \mathbf{p}_n with a boundary ϕ . The most likely $z_n^L, z_n^R, t_n^L, t_n^R$, and ϕ are obtained by recursively back-tracking the most likely paths from $t_{n:n}$.

Two adjacent chords z_n and z_{n+1} associated with non-terminal symbols $t_{n:n}$ and $t_{n+1:n+1}$ are merged into a single chord z associated with a non-terminal symbol $t_{n:n+1}$ by maximizing the conditional posterior distribution given by $p(z|t_{n:n+1}, z_{-n:n+1}, \phi_{-n+1}, \mathbf{p}, \psi, \Theta)$. The most likely z is obtained as follows:

$$z = \arg \max_{z' \in \Sigma} \eta_{t_{n:n+1} \rightarrow z'} p(\mathbf{p}_n|z') p(\mathbf{p}_{n+1}|z'). \quad (16)$$

5.4 Updating the Melody

When a chord symbol z_n , the last pitch $p_{n-1, I_{n-1}}$ in the region of the previous chord z_{n-1} , and the first pitch $p_{n+1, 1}$ in the region of the next chord z_{n+1} are given, a sequence of musical notes in the region of z_n (between ϕ_n and ϕ_{n+1}) is obtained by maximizing the conditional posterior distribution $p(\mathbf{p}_n|z_n, p_{n-1, I_{n-1}}, p_{n+1, 1}, \Theta)$. To do this, we propose an efficient algorithm based on dynamic programming. Let α_{y_t, d_t} be the marginal likelihood that a note at the pitch y_t is located on the score time t and the duration of the previous note is d_t on a chord z_n :

$$\alpha_{y_t, d_t} = p(y_t, d_t|z_n) \quad (17)$$

This probability can be calculated recursively in the score time $t \in \{\phi_n, \dots, \phi_{n+1}, \psi_{n+1, 1}\}$.

$$\alpha_{y_t, d_t} = \rho_{t-d_t, t} \sum_{y_{t-d_t}, d_{t-d_t}} \alpha_{y_{t-d_t}, d_{t-d_t}} \tau_{y_{t-d_t}, y_t}^{z_n}$$

In each score time t , d_t can take values in $\{1, \dots, t, t - \psi_{n-1, I_{n-1}}\}$. By using this probability, we can recursively sample \mathbf{p}_n from the beat score time $\psi_{n+1, 1}$ to $\psi_{n-1, I_{n-1}}$.

Another improved way of partially updating the melody is to use the LSTM model. Suppose that we aim to update $x_{i:j}$ in the whole melody \mathbf{x} . Given a chord sequence \mathbf{c} and melody segments $x_{1:i-1}$ and $x_{j+1:T}$, the missing part $x_{i:j}$ can be sampled from the conditional posterior distribution $p(x_{i:j}|\mathbf{c}, x_{1:i-1}, x_{j+1:T}) \propto p(\mathbf{x}|\mathbf{c})$. First, the pitches $x_{1:i-1}$ and chords $c_{1:i-1}$ are fed to the network to update the hidden states. The missing part $x_{i:j}$ is then sampled sequentially according to the probability $p(x_{t+1}|x_{1:t}, c_{1:t})$ learned by the LSTM. This enables us to evaluate $p(\mathbf{x}|\mathbf{c})$. Among a sufficient number of generated samples of $x_{1:i-1}$, a sample with the highest $p(\mathbf{x}|\mathbf{c})$ is selected.

6. EVALUATION

This section reports objective and subjective evaluations on the user interface and the music arrangement method.

6.1 Experimental Conditions

To train the PCFG, we used 705 chord sequences of musical sections (e.g., verse, bridge, and chorus) from 468 pieces of popular music included in the SALAMI dataset [25]. Only chord sequences with a length between 8 and 32 measures were chosen. The vocabulary of chord symbols was limited to the combinations of the 12 root notes {C, C#, ..., B} and the 2 chord types {major, minor}. The

number of kinds of non-terminal symbols of the PCFG was set to 12. The values of the hyperparameter ι_A were all set to 1.0 and those of the other parameters were all set to 0.1. To train the three Markov models, we used 9902 pairs of melodies and the corresponding chord sequences from 194 pieces of popular music included in Rock Corpus [5]. To train the LSTM, we used 9265 melodies associated with chord sequences from pieces of popular music included in Rock Corpus and Nottingham Database [1]. Note that all of the data used in our experiments were transposed to the C major or C minor key. The number of the hidden units was 50 and the softmax-cross-entropy was used as a loss function. The parameters of the LSTM were optimized by using Adam [14]. The number of samples generated by the LSTM (described in Section 5.4) was 50.

6.2 Objective Evaluation of Melody Arrangement

We evaluated the function of updating a melody in terms of the note density of the generated musical notes via 10-fold cross validation on the Rock Corpus and Nottingham Database. For the region of each chord z_n , a sequence of melody \mathbf{p}_n is arranged by using the two methods based on the Markov model and the LSTM described in Section 5.4. We measured the mean squared error (MSE) between the note density per measure of the generated musical notes and the mean value of the density of other regions given by

$$\text{MSE} = \frac{1}{N-1} \sum_{n=1}^{N-1} \left\{ \frac{16I_n^*}{\phi_{n+1} - \phi_n} - \frac{\sum_{m \neq n} 16I_m}{\sum_{m \neq n} (\phi_{m+1} - \phi_m)} \right\}^2,$$

where I_n^* and I_n were the number of generated musical notes and that of the original musical notes, respectively. The average MSE was calculated over all melodies. The average MSE obtained by the LSTM model was 5.52 while that obtained by the Markov model was 6.42. This indicates that the LSTM-based method is a little more effective for updating a partial melody in consideration of the note density of the whole melody because it can capture the long-term dependency.

6.3 Subjective Evaluation of the Proposed System

We conducted the subjective evaluation of the system¹ in terms of usability and effectiveness in interactive chord and melody arrangement. Five melodies of 8 measures were extracted from the RWC music database [9, 10]. We asked 11 subjects to test our system. Four subjects who had the experience of playing musical instruments for more than five years were regarded as people with musical backgrounds. Each subject was asked to interactively make a musical piece by using each of the five melodies as an initial seed and then grade the system on a 5-point Likert scale (from “strongly agree (1)” to “strongly disagree (5)”) in terms of the following 15 criteria:

- The chord sequences obtained were suitable for the melodies (I).
- The chord sequences obtained by the split or merge operation were musically natural (II, III).

¹ The interface used in this experiment is available online: <http://sap.ist.i.kyoto-u.ac.jp/members/tsushima/ismir2018/>

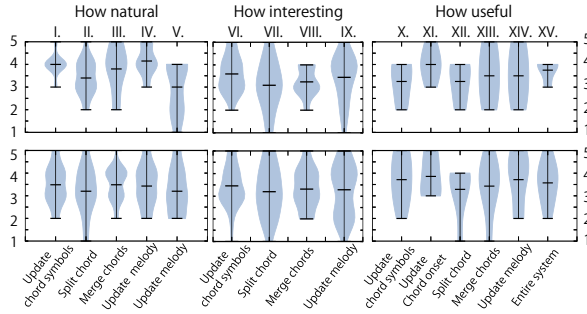


Figure 5: Results for people with musical backgrounds (top) and those for people without musical backgrounds (bottom). The middle bars indicate the mean value.

- The melodies obtained were suitable for the chord sequences (IV).
- The melodies obtained were musically natural (V).
- The musical pieces obtained by updating chord symbols, splitting a chord, merging chords, or updating a melody were interesting (VI, VII, VIII, IX).
- The function of updating chord symbols, splitting a chord, merging chords, or updating a melody was useful (X, XI, XII, XIII, XIV).
- The user interface has the capability of helping users make musical pieces (XV).

We also asked the subjects to tell us how each of them felt about the system.

The results of this user study is shown in Fig 5. In terms of the naturalness and the interestingness, the two operations, updating chord symbols and updating melodies, obtained the slightly high mean ratings of 3.67 in criterion (I), 3.69 in criterion (IV), and 3.51 in criterion (VI). As seen in the score for the criterion (V), the subjects with musical backgrounds, compared with the others, tended to feel that the updated melodies were less musically natural. As seen in the score for the criterion (IX), the subjects with musical backgrounds tended to feel that the updated melodies were more interesting. In terms of the usefulness of each operation, each operation obtained the reasonably high mean ratings (from 3.27 to 3.91).

We obtained the following opinions on the usability of our system:

- It was interesting that even a user without any experiences in music composition can edit a musical piece by iterating several operations.
- An operation that updates one chord symbol is necessary for more freely editing a chord sequence.

We also obtained the following opinions on the problems of some operations:

- The chord sequences obtained were almost always appropriate for all samples of melodies but the system tended to generate only basic chords (e.g., C major).
- The updated melodies were often unnatural when an original melody has some repeated sections.

The reason for the former problem may be that the chord symbols are updating by using the Viterbi algorithm. The reason for the latter problem is probably that the LSTM cannot capture the global repetitive structure of a melody.

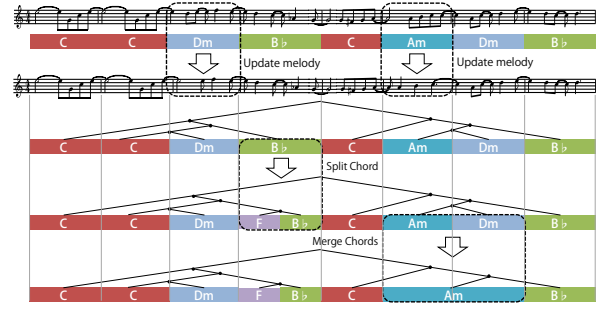


Figure 6: Example operation for interactive generation of chord sequences and melodies.

6.4 Example of Chord and Melody Arrangement

Fig. 6 shows how the proposed method generates chord sequences and melodies. The score (melodies and chords) at the top shows an initial state in which the chord symbols were optimized for the melody in the input file (the chord onsets were located at the bar lines). The second score shows the state in which the two regions of the melody under the 3rd and 6th chords were updated in order. The third chord sequence shows the state in which the 4th chord, B \flat major, was split into F major and B \flat major. The fourth chord sequence shows the state in which the 7th chord, A minor, and the 8th chord, D minor, were merged into A minor. This indicates that the proposed method can successfully help a user partially update a melody while keeping the consistency of the whole melody and that it can generate a chord sequence by considering the latent tree structure behind the chord sequence.

7. CONCLUSION

This paper presented an interactive music arrangement system that enables a user to incrementally refine a chord sequence and a melody. The experimental results showed that the proposed system has a great potential to help a user create his or her original musical pieces.

There would be much room for improving our method. To improve the diversity of generated chord symbols, the use of some sampling or beam-search method would be effective. To improve the naturalness of generated melodies, the use of a bidirectional LSTM [23] would be effective for considering the repetitive structures of melodies.

For more specific studies on the effectiveness of our system, we plan to measure how well test users can incrementally refine a musical piece compared with the conventional methods, by counting the number of necessary operations to make musical pieces meet their satisfaction. We also plan to conduct large-scale user studies of the system on the Web. Collecting time-series data of users' operations and created pieces, it would be possible to infer their musical preference and improve the model by reinforcement learning. Using the same data, it would be possible to reveal the process of music creation by humans in terms of edit operations and optimization strategies.

Acknowledgements: This study was partially supported by JST ACCEL No. JPMJAC1602, JSPS KAKENHI No. 26700020 and No. 16H01744, and Grant-in-Aid for JSPS Research Fellow No. 16J05486.

8. REFERENCES

- [1] ABC version of the Nottingham music database. <http://abc.sourceforge.net/NMD/>.
- [2] M. Allan and C. Williams. Harmonising chorales by probabilistic inference. In *NIPS*, pages 25–32, 2005.
- [3] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *ICML*, 2012.
- [4] C. H. Chuan and E. Chew. A hybrid system for automatic generation of style-specific accompaniment. In *IJWCC*, pages 57–64, 2007.
- [5] T. D. Clercq and D. Temperley. A corpus analysis of rock harmony. *Popular Music*, 30(01):47–70, 2011.
- [6] K. Ebcioglu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3):43–51, 1988.
- [7] D. Eck and J. Schmidhuber. A first look at music composition using LSTM recurrent neural networks. *IDSIA*, 103(07-02), 2002.
- [8] S. Fukayama et al. Orpheus: Automatic composition system considering prosody of Japanese lyrics. In *ICMC*, pages 309–310. Springer, 2009.
- [9] M. Goto. AIST annotation for the RWC music database. In *ISMIR*, pages 359–360, 2006.
- [10] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical and jazz music databases. In *ISMIR*, pages 287–288, 2002.
- [11] R. Groves. Automatic harmonization using a hidden semi-Markov model. In *AIIDE*, pages 48–54, 2013.
- [12] G. Hadjeres and F. Pachet. DeepBach: A steerable model for Bach chorales generation. In *ICML*, pages 1362–1371, 2017.
- [13] M. Johnson, T. L. Griffiths, and S. Goldwater. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *NAACL-HLT*, pages 139–146, 2007.
- [14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICMR*, pages 1–15, 2014.
- [15] O. Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. In *Constructive Machine Learning Workshop (NIPS 2016)*, 2016.
- [16] J. F. Païement, D. Eck, and S. Bengio. Probabilistic melodic harmonization. In *CSCSI*, pages 218–229, 2006.
- [17] G. Papadopoulos and G. Wiggins. AI methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB Symposium on Musical Creativity*, pages 110–117, 1999.
- [18] R. D. Prisco and R. Zaccagnino. An evolutionary music composer algorithm for bass harmonization. In *Applications of Evolutionary Computing*, pages 567–572. Springer, 2009.
- [19] R. De Prisco, A. Eletto, A. Torre, and R. Zaccagnino. A neural network for bass functional harmonization. In *European Conference on the Applications of Evolutionary Computation*, pages 351–360. Springer, 2010.
- [20] S. A. Raczynski, S. Fukayama, and E. Vincent. Melody harmonization with interpolated probabilistic models. *Journal of New Music Research*, 42(3):223–235, 2013.
- [21] M. Rohrmeier. Mathematical and computational approaches to music theory, analysis, composition and performance. *Journal of Mathematics and Music*, 5(1):35–53, 2011.
- [22] C. Roig, L. J. Tardón, T. Barbancho, and A. M. Barbancho. Automatic melody composition based on a probabilistic model of music style and harmonic rules. *Knowledge-Based Systems*, 71:419–434, 2014.
- [23] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
- [24] I. Simon, D. Morris, and S. Basu. Mysong: automatic accompaniment generation for vocal melodies. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 725–734. ACM, 2008.
- [25] J. B. L. Smith, J. A. Burgoyne, I. Fujinaga, D. D. Roure, and J. S. Downie. Design and creation of a large-scale database of structural annotations. In *ISMIR*, pages 555–560, 2011.
- [26] M. J. Steedman. A generative grammar for jazz chord sequence. *Music Perception*, 2(1):52–77, 1984.
- [27] M. Towsey, A. Brown, S. Wright, and J. Diederich. Towards melodic extension using genetic algorithms. *Educational Technology & Society*, 4(2):54–65, 2001.
- [28] H. Tsushima, E. Nakamura, K. Itoyama, and K. Yoshii. Function- and rhythm-aware melody harmonization based on tree-structured parsing and split-merge sampling of chord sequences. In *ISMIR*, pages 502–508, 2017.
- [29] H. Tsushima, E. Nakamura, K. Itoyama, and K. Yoshii. Generative statistical models with self-emergent grammar of chord sequences. *Journal of New Music Research*, 2018. To appear.
- [30] E. Waite. Generating long-term structure in songs and stories. <https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn>.
- [31] L. C. Yang, S. Y. Chou, and Y. H. Yang. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In *ISMIR*, pages 324–331, 2017.

A GENERALIZED PARSING FRAMEWORK FOR GENERATIVE MODELS OF HARMONIC SYNTAX

Daniel Harasim^{1,2}**Martin Rohrmeier^{1,2}**

Timothy J. O'Donnell³

¹ Digital and Cognitive Musicology Lab, École Polytechnique Fédérale de Lausanne, Switzerland

² Institut für Kunst- und Musikwissenschaft, TU Dresden, Germany

³ Department of Linguistics, McGill University, Canada

daniel.harasim@epfl.ch

ABSTRACT

Modeling the structure of musical pieces constitutes a central research problem for music information retrieval, music generation, and musicology. At the present, models of harmonic syntax face challenges on the tasks of detecting local and higher-level modulations (most previous models assume a priori knowledge of key), computing connected parse trees for long sequences, and parsing sequences that do not end with tonic chords, but in turnarounds. This paper addresses those problems by proposing a new generative formalism Probabilistic Abstract Context-Free Grammars (PACFGs) to address these issues, and presents variants of standard parsing algorithms that efficiently enumerate all possible parses of long chord sequences and to estimate their probabilities. PACFGs specifically allow for structured non-terminal symbols in rich and highly flexible feature spaces. The inference procedure moreover takes advantage of these abstractions by sharing probability mass between grammar rules over joint features. The paper presents a model of the harmonic syntax of Jazz using this formalism together with stochastic variational inference to learn the probabilistic parameters of a grammar from a corpus of Jazz-standards. The PACFG model outperforms the standard context-free approach while reducing the number of free parameters and performing key finding on the fly.

1. INTRODUCTION

The modeling of non-local relations between musical objects such as notes and chords constitutes a central research problem for music information retrieval, music generation, and music analysis. Hierarchical models express these relations by assuming a latent hierarchical structure [19,22–24,30,31]. Consider for example the Jazz chord sequence $\text{Am}^7 \text{D}^7 \text{G}^7 \text{C}^\Delta$ where C^Δ denotes a major-seventh chord. Since the first three chords form a II V I sequence

with reference to G^7 which is the dominant in C major, they form a *dominant phrase* [24]. The dominant phrase as a whole then refers to the tonic chord C^Δ . All four chords together thus form a *tonic phrase*.

Figure 1 presents a syntactic analysis of the A-part of the Jazz-standard *Afternoon in Paris* following the approach from [22]. It illustrates the idea of how pieces can be decomposed into hierarchically-structured *constituents* which stand in part-whole relationship with one another. Subdominant, dominant, and tonic phrases are denoted by the scale degrees II, V, and I, respectively. Note that the subsequence $\text{Cm}^7 \text{ F}^7 \text{ Bb}^\Delta$ is both a tonic progression in Bb major and a dominant progression in Eb major. It forms a dominant phrase in Ab major together with Bbm^7 and Eb^7 .

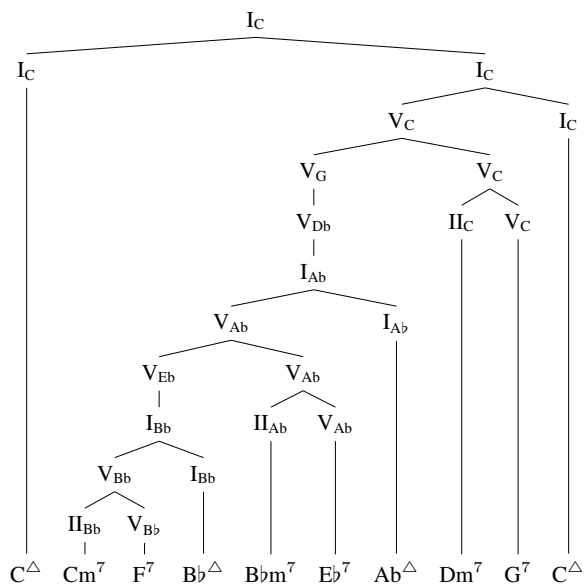



Figure 1. Hierarchical analysis of the A-part of the Jazz-standard *Afternoon in Paris*.

Models of harmonic syntax similar to Figure 1 have been successfully applied to melody harmonization [16], chord inference from audio [5, 6], and harmonic similarity [7]. There is also some empirical evidence for the psychological reality of hierarchical structures in music [15, 25]. While earlier theoretical and psychological work on hierar-



 © Daniel Harasim, Martin Rohrmeier, Timothy J. O'Donnell. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Daniel Harasim, Martin Rohrmeier, Timothy J. O'Donnell. “A Generalized Parsing Framework for Generative Models of Harmonic Syntax”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

chical models has provided important insight about musical structure, computational implementation of these models to date has been limited to relatively small datasets. Earlier work includes applications to monophonic melodic data [21], a corpus of 39 blues chord progressions with a maximum of 24 chords per progression [12], or a dataset of 76 chord progressions (avg. length 40) from Jazz-standards that was restricted to subsequences of pieces that did not change key [4]. All these earlier approaches assume the knowledge of the key of the pieces a priori.

In computational linguistics, Context-Free Grammars (CFGs) are a standard way of modeling hierarchical constituent structure. They formalize constituent structures using *rewrite rules* denoted by long right arrows. The rule $X \rightarrow Y Z$ for example states that the constituent X consists of the two constituents Y and Z . The existence of natural language treebanks makes it possible to read off the grammatical rewrite rules including their frequencies from syntactical analyses by experts. At present, there are music databases of simplified Schenkerian analyses [13], syntactic analyses of melodies based on the generative theory of tonal music [8], and annotated harmonic functions [4]. However, to the best of our knowledge there is currently no dataset of hierarchically analyzed chord sequences by human experts that could serve for the training or the evaluation of models of harmonic syntax. As a consequence, there exist no comparisons of models of harmonic syntax against expert analyses.

In the following, we introduce Abstract Context-Free Grammars (ACFGs), a generalization of the CFG framework designed to account for feature structures characteristic of musical categories. A first model of Jazz harmony is proposed in this framework that covers full pieces by incorporating modulations (i.e., changes in key). We train the model in a semi-supervised fashion on a dataset of Jazz-standards and evaluate it on a small set of hand-annotated hierarchical analyses. We further propose a solution for handling sequences that do not end with tonic chords, but in turnarounds. Simulations demonstrate that the ACFG model is able to outperform a PCFG model of the dataset. The implementation of the algorithms developed in this study are publicly available as a package of the Julia programming language [1].¹

2. OVERVIEW OF THE APPROACH

While the CFG framework has proven invaluable in computational linguistics, categories and part-whole relations between musical constituents have properties not possessed by linguistic structures. Musical categories such as scale degrees, for example, are equipped with an arithmetic structure that corresponds to musical transposition.

In the following, we refer to context-free rules of the form $X \rightarrow Y X$ as a *preparation* of X by Y . The preparation of the scale degree V_{Bb} by II_{Bb} in *Afternoon in Paris* (see Figure 1) for example is a concrete realization of the general principle that any category x_k consisting of a scale

degree x and a key k can be prepared by an ascending diatonic fifth $(x+4 \bmod 7)_k$. [24]. In addition to facts such as these, a framework for modeling musical structure has to account for the fact that the musical categories and rewrite rules are grouped into key-independent classes. For example, both V_{Bb} and V_{Ab} are fifth scale degrees. The probabilities of the application a rule to V_{Bb} and V_{Ab} should therefore be related.

This paper introduces Abstract Context-free Grammars (ACFGs), a modeling framework with a greater flexibility than CFGs. In particular, in ACFGs constituent categories are allowed to be of any data type and the rules are generalized partial functions. Unlike standard context-free rules, ACFG rules can therefore take advantage of the algebraic structure of categories. Probabilistic ACFGs extend probabilistic CFGs with the ability to express a wider range of probability distributions over rules.

3. ABSTRACT CONTEXT-FREE GRAMMARS

3.1 Definitions

Definition 1. A (*non-probabilistic*) *Abstract Context-free Grammar* (ACFG) $G = (T, C, C_0, \Gamma)$ consists of a set T of *terminal symbols*, a set C of *constituent categories*, a set of *start categories* $C_0 \subseteq C$, and a set of partial functions

$$\Gamma := \{ r \mid r : C \rightarrow (T \cup C)^* \},$$

called *rewrite rules* or *rewrite functions*. The arrow \rightarrow is used throughout the paper to denote partial functions. A sequence $\beta \in (T \cup C)^*$ can be *generated in one step* from a sequence $\alpha \in (T \cup C)^*$ by the application of a rewrite function $r \in \Gamma$, denoted by $\alpha \rightarrow_r \beta$, if there exist $\alpha_1, \alpha_2 \in (T \cup C)^*$ and $A \in C$ such that $\alpha = \alpha_1 A \alpha_2$ and $\beta = \alpha_1 r(A) \alpha_2$. A sequence of rewrite rules $r_1 \dots r_n$ is called a *derivation* of a sequence of terminals $\alpha \in T^*$ if there exists a start category $\alpha_1 \in C_0$, and $\alpha_2, \dots, \alpha_n \in (C \cup T)^*$ such that

$$\alpha_1 \rightarrow_{r_1} \alpha_2 \rightarrow_{r_2} \dots \rightarrow_{r_n} \alpha,$$

where r_i is always applied to the leftmost category of α_i for $i \in \{1, \dots, n-1\}$. The set of derivations of α is denoted by $D(\alpha)$. The language of the grammar G is the set of terminal sequences that have a derivation in G .

Note that if C is finite, the languages that can be described by ACFGs are exactly the languages that can be described by standard context-free grammars (CFGs). For each ACFG with finite C , a CFG with rule set R can be constructed by dividing each rewrite function with domain cardinality k into k standard context-free rewrite rules,

$$R := \bigcup_{r \in \Gamma} \{ (A, \alpha) \in C \times (T \cup C)^* \mid r(A) = \alpha \}.$$

Definition 2. A *Probabilistic Abstract Context-free Grammar* (PACFG) is an ACFG where each category $A \in C$ is associated with a random variable X_A over rewrite functions r such that $\mathbb{P}(X_A = r)$ is positive if and only if $r(A)$

¹ <https://github.com/dharasim/GeneralizedChartParsing.jl>

is defined, that is A is in the domain of r , $A \in \text{dom}(A)$. The probability $p(d)$ of a derivation $d = r_1 \dots r_n$ of a sequence of terminal symbols $\alpha \in T^*$ is defined as the product $\prod_{i=1}^n \mathbb{P}(X_{A_i} = r_i)$ where in each step r_i is applied to a category $A_i \in C$. The probability of α is then defined as $p(\alpha) = \sum_{d \in D(\alpha)} p(d)$.

Note that PACFG categories can share the same probability distribution over rewrite functions without rewriting to exactly the same right-hand sites. This important property allows us to model the structural relations between musical keys. We use this property in Section 4 to build a model that abstract chords sequences from their concrete scale by defining the probability that a rewrite function is applied to a scale degree independently of its key. The sharing of probability mass between rules additionally reduces the number of free parameters of a PACFG model.

To illustrate the different learning capabilities of PCFG and PACFG models, consider a toy PCFG with nonterminal symbols $C = \{S, A, B\}$, start symbol S , terminal symbols $T = \{a, b\}$, and rules $S \rightarrow A \mid B$, $A \rightarrow A A \mid a$, and $B \rightarrow B B \mid b$. The grammar thus generates sequences that solely consist either of as or bs . In a classical PCFG setting, no probability mass is shared between rules, but each rule has its separate probability. However, in the process of inferring the probabilities of the rules from data, it might be desirable to generalize the rules $A \rightarrow A A$ and $B \rightarrow B B$ to a meta rule $x \rightarrow x x$ where $x \in \{A, B\}$ and to put probability mass on this abstract entity. In that way, the grammar can learn something about $A \rightarrow A A$ when it observes $B \rightarrow B B$ and vice versa. The PACFG version of the PCFG presented above addresses the problem by replacing the classical context-free rules by the partial functions r_1, r_2, r_3, r_4 , and r_5 with $r_1(S) = A$, $r_2(S) = B$, $r_3(x) = x x$ for $x \in \{A, B\}$, $r_4(A) = a$, and $r_5(B) = b$. Analogously, a PACFG of Jazz chord sequences can generalize classical rewrite rules so that their probabilities do not depend on the keys of their left-hand sides to model transpositional invariance.

3.2 Parsing

Parsing a sequence of terminal symbols with respect to a formal grammar is the task of computing the distribution of parse trees conditioned on this sequence. Many parsers are based on versions of the CYK algorithm that assumes grammars to be given in Chomsky normal form. Since grammar transformations into Chomsky normal form considerably blow up the grammar, the here presented parser transforms grammars on the fly during parsing, similar to the transformation presented in [18]. Each rule of the form $A \rightarrow B_1 \dots B_k$ is transformed into a set of states $s_i = B_1 \dots B_i$ for $1 \leq i \leq k$, a transition function

$$\text{tran} : S \times (T \cup C) \rightarrow S, \quad \text{tran}(s_i, B_{i+1}) = s_{i+1}$$

and a completion function $\text{comp} : S \rightarrow 2^C$ such that $\{A\} \subseteq \text{comp}(s_k)$, where S denotes the set of all states. Note that the states and the transition function form a search trie where the completion function checks if there

items:	edges	$[s, i, j]$	for $s \in S$
	constituents	$[A, i, j]$	for $A \in C$
			for and $i, j \in \{1, \dots, \alpha + 1\}$
goal items:		$[A, 1, \alpha + 1]$	for $A \in S$
axioms:		$\overline{[\alpha_i, i, i + 1]}$	for $i \in \{1, \dots, \alpha \}$
introduce edge:		$\frac{[A, i, j]}{[s, i, j]}$	$s = \text{tran}(s_0, A)$
complete edge:		$\frac{[s, i, j]}{[A, i, j]}$	$A \in \text{comp}(s)$
fundamental rule:		$\frac{[s, i, j] \quad [A, j, k]}{[s', i, k]}$	$\text{tran}(s, A) = s'$

Figure 2. Description of the parsing algorithm in the parsing as deduction framework. Existing Constituents can start the parser to read a sequence of terminal symbols and categories by the *introduce edge* rule. The *fundamental rule* is then recursively applied to extend these sequences. The *complete edge* rule eventually merges sequences to single constituents if they are the right-hand side of a grammar rule.

is a rewrite rule that has a sequence of terminal symbols and categories as its right-hand side. This trie data structure leads to a compact representation of the forest of all trees for a given input sequence. More generally, the parser can handle any transition and completion functions derived from finite-state automata, see [14].

In the following, a generic bottom-up parsing algorithm for abstract grammars is presented in the parsing as deduction framework using the above defined transition and completion functions [3, 29]. The parsing as deduction framework is a meta-formalism to state and compare different parsing algorithms. It views the parses of a sequence as logical deductions of goals from axioms by using constituents as atomic logical formulas. The formula $[I_{Bb}, 2, 5]$ for example states the existence of a constituent with category I_{Bb} that spans over the second, third, and fourth terminal symbol. This formula is true in the analysis presented in Figure 1 because that analysis contains a constituent with label I_{Bb} over the span from the second to the forth leaf chord. The goals are constituents that span the full sequence and come from the set of start categories. The axioms are formulas of the form $[t_i, i, i + 1]$ for each terminal in the input sequence $t_1 \dots t_n$. The parsing strategy such as bottom-up parsing or Earley parsing is encoded in the deduction rules. These rules are denoted by a set of atomic formulas over a horizontal line, an atomic formula under this line, and an optional side condition (see Figure

2). The formula under the line can be deduced from the formulas above if the side condition holds.

The proposed algorithm makes use of two different kinds of atomic formulas: edges (not yet completed constituents) and constituents. A state $s \in S$ together with a start index i and an end index j is called an *edge* and denoted by $[s, i, j]$. Analogously, a category $A \in C$ together with start and end indices i and j is called a *constituent* and denoted by $[A, i, j]$. Figure 2 shows the axioms, goal items, and the deduction rules of our algorithm.

3.3 Inference of Rule Probabilities

In this section, we give an overview of an inference algorithm for the rule probabilities $\mathbb{P}(X_A = r)$. Let $\Gamma_A = \{r \in \Gamma \mid A \in \text{dom}(r)\}$ be the set of rewrite functions whose domain contains the constituent category A . We place a Dirichlet distribution on the probability vector describing the distribution over Γ_A , $\vec{\theta}_{\Gamma_A} \sim \text{Dirichlet}(\vec{\alpha}_{\Gamma_A})$ for pseudocount vector $\vec{\alpha}_{\Gamma_A}$. The inference problem is to compute the posterior distribution over this set of probability vectors, given the data D and pseudocounts $\{\vec{\alpha}_{\Gamma_A}\}$,

$$p(\{\vec{\theta}_{\Gamma_A}\} \mid D, \{\vec{\alpha}_{\Gamma_A}\}) \propto p(D \mid \{\vec{\theta}_{\Gamma_A}\})p(\{\vec{\theta}_{\Gamma_A}\} \mid \{\vec{\alpha}_{\Gamma_A}\}),$$

where $\{\vec{\theta}_{\Gamma_A}\}$ is an abbreviation for $\{\vec{\theta}_{\Gamma_A}\}_{A \in C}$, etc. *Variational Bayesian inference* (VB) is used to approximate this posterior distribution [2, 11, 32]. We introduce an approximating *variational distribution* $q(\{\vec{\theta}_{\Gamma_A}\} \mid \{\vec{v}_{\Gamma_A}\})$ with *variational parameters* $\{\vec{v}_{\Gamma_A}\}$ over our target hidden variables (rule weights) and minimize the Kullback-Leibler divergence between this approximation and the true posterior,

$$D_{\text{KL}}(q(\{\vec{\theta}_{\Gamma_A}\} \mid \{\vec{v}_{\Gamma_A}\}) \parallel p(\{\vec{\theta}_{\Gamma_A}\} \mid D, \{\vec{\alpha}_{\Gamma_A}\})),$$

by adjusting the variational parameters $\{\vec{v}_{\Gamma_A}\}$.

Following [17], we approximate the distribution over each probability vector with a Dirichlet distribution $\vec{\theta}_{\Gamma_A} \mid \vec{v}_{\Gamma_A} \sim \text{Dirichlet}(\vec{v}_{\Gamma_A})$, and make use of the *mean-field approximation*

$$q(\{\vec{\theta}_{\Gamma_A}\} \mid \{\vec{v}_{\Gamma_A}\}) = \prod_{A \in C} p(\vec{\theta}_{\Gamma_A} \mid \vec{v}_{\Gamma_A}).$$

We minimize the Kullback-Leibler divergence with a coordinate descent algorithm similar to the expectation-maximization algorithm. First, we compute the expectation of the counts of rule usages in the data under our current setting of the variational parameters, $\mathbb{E}_q[\#(r, D)]$ where $\#(r, D)$ is the number of times that rule r was used to generate the data D , and then we update our variational parameters based on these expectations. Since all of our distributions are in the exponential family, it can be shown that the optimal update is given by the equation $\vec{v}_{\Gamma_A} = \vec{\alpha}_{\Gamma_A} + \mathbb{E}_q[\#(r, D)]$ [2]. In other words, we set the pseudocounts of our variational distributions equal to the expected number of rule usages plus the pseudocount for each rule in the prior distribution.

Under the standard coordinate-ascent algorithm given in [17], expected counts must be computed for the whole

corpus before updating using the equation above. Hoffman et al. [9] propose a stochastic variant of the standard variational (inspired by *stochastic gradient descent*) where updates are computed with respect to randomly sampled *minibatches* of the data. We make use of this *stochastic variational Bayes* algorithm in the results reported below.

4. A GENERATIVE MODEL OF JAZZ HARMONY

This section presents a PACFG $G = (T, C, C_0, \Gamma)$ that models the syntax of Jazz harmony following the proposal in [24]. That work addressed the problem of finding a restrictive grammar that describes the full variety of syntactic relations in the musical idiom of Jazz-standards. The set of terminal symbols T is a set of pairs describing chords each of which consists of the root of the chord and a string describing the chord form—one of: a major triad, a major-seventh chord, a major sixth chord, a dominant-seventh chord, a minor triad, a minor-seventh chord, a half-diminished-seventh chord, a diminished seventh-chord, an augmented triad, or a suspended chord.

In the following, \mathbb{Z}_n denotes the ring of integers modulo $n \in \mathbb{N}$. The categories are modeled as pairs of scale degrees and keys, $C = \mathbb{Z}_7 \times K$, where a key consists of a pitch class representing its root and a string describing its mode, $K = \mathbb{Z}_{12} \times \{\text{major, min}\}$. Scale degrees are denoted by roman numerals from I to VII. All categories with scale degree I are start symbols, $C_0 = \{I\} \times K$. Let $k \in K$ denote an arbitrary key. The set of rewrite functions Γ consists of *prolongation*,

$$\text{PROLONG}(\langle x, k \rangle) = \langle x, k \rangle \langle x, k \rangle$$

for $x \in \mathbb{Z}_7$, *diatonic preparation*,

$$\text{DIAT-PREP}(\langle x, k \rangle) = \langle x + 4 \bmod 7, k \rangle \langle x, k \rangle$$

for $x \in \mathbb{Z}_7 \setminus \{IV\}$, *dominant preparation*,

$$\text{DOM-PREP}(\langle x, k \rangle) = \langle V, \mu(x, k) \rangle \langle x, k \rangle$$

for $x \in \mathbb{Z}_7 \setminus \{I\}$ where $\mu(x, k)$ denotes the modulation from k into the key of scale degree x (e.g. $\mu(II, (0, \text{maj})) = (2, \text{min})$, the key of the second scale degree of C major is D minor), *plagal preparation*,

$$\text{PLAGAL-PREP}(\langle I, k \rangle) = \langle IV, k \rangle \langle I, k \rangle,$$

modulation,

$$\text{MODULATION}(\langle x, k \rangle) = \langle I, \mu(x, k) \rangle,$$

mode change,

$$\text{MODE-CHANGE}(\langle I, (r, m) \rangle) = \begin{cases} \langle I, (r, \text{min}) \rangle, & \text{if } m = \text{maj} \\ \langle I, (r, \text{maj}) \rangle, & \text{if } m = \text{min}, \end{cases}$$

for $r \in \mathbb{Z}_{12}$, $m \in \{\text{maj, min}\}$, *diatonic substitution*,

$$\text{DIAT-SUBST}(\langle x, (r, m) \rangle) = \begin{cases} \langle VI, (r, m) \rangle, & \text{if } x = I, m = \text{maj} \\ \langle III, (r, m) \rangle, & \text{if } x = I, m = \text{min} \\ \langle IV, (r, m) \rangle, & \text{if } x = II \\ \langle VII, (r, m) \rangle, & \text{if } x = V \end{cases}$$

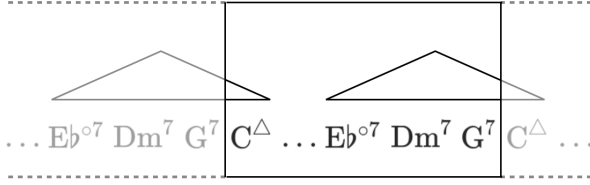


Figure 3. Parsing the turnaround of *All of me*

for $x \in \{I, II, V\}$, $r \in \mathbb{Z}_{12}$, $m \in \{\text{maj}, \text{min}\}$, and *dominant substitution*,

$$\text{DOM-SUBST}_i(\langle V, (r, m) \rangle) = \langle V, (r + i \bmod 12, m) \rangle$$

for $r \in \mathbb{Z}_{12}$, $m \in \{\text{maj}, \text{min}\}$, and $i \in \{3, 6, 9\}$. Additionally, Γ contains appropriate termination rules $C \rightarrow T$ according to standard Jazz harmony theory (e.g. seventh-chord-termination($\langle 4, (0, \text{maj}) \rangle$) = G^7 , see [20] for further explanation). The distribution of $X_{\langle x, k \rangle}$ over rules rewriting the category $\langle x, k \rangle$ is defined as a categorical distribution such that $\mathbb{P}(X_{\langle x, k \rangle} = r) = \mathbb{P}(X_{\langle x, k' \rangle} = r)$ for all scale degrees x , rules r , and keys k, k' that have the same mode. That is, the probability of r rewriting $\langle x, k \rangle$ does not depend on the root of k which enables the model to learn the parameters of its probability distributions key-independently.

These grammar rules can be grouped into three classes: the prolongation rule, preparation rules, and substitution rules. Preparation rules create categories that for the listener generate the expectation to hear the prepared chord. Substitution rules substitute chords for other chords that fulfill an equivalent function inside the sequence such as tritone substitutions of dominants in Jazz.

5. THE TURNAROUND PROBLEM

A lead-sheet of a Jazz-standard consists of a melody together with a chord sequence describing the fundamental harmonic structure of the piece. The chord sequence is repeated multiple times in a performance. While some lead-sheets end with tonic chords, others include harmonic upbeats to the first chord of the piece at the end of the sheet, called *turnarounds*. The final chord of a performance is nevertheless usually a tonic chord. The lead-sheet of the Jazz-standard *All of me* starts for example with a C^Δ chord and ends with the turnaround $E\flat^\circ 7 Dm^7 G^7$.

The grammar of Jazz harmony proposed above assumes that pieces end with a tonic chord. Therefore, a simple implementation of this grammar would not be able to parse lead-sheets that end in turnarounds. We solve this problem by *cyclic parsing*, meaning that we assume that constituents can have spans from the end of a piece back to the beginning, see Figure 3.

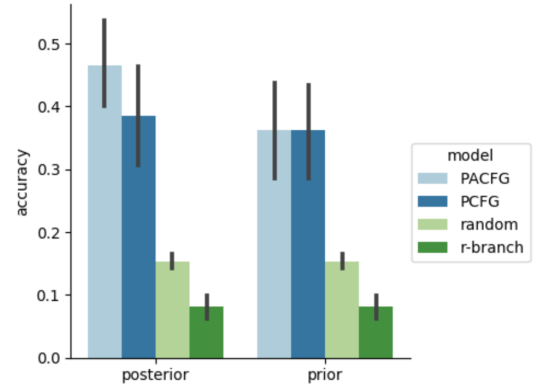


Figure 4. Tree accuracy plot

6. EXPERIMENTS

6.1 Dataset

The model is evaluated using the *iRealPro* dataset of Jazz-standards.² This dataset consists of 1173 chord sequences electronically-encoded by the Jazz musician community including metadata such as the titles, composers, and keys. The sequences were collected and converted into the Humdrum format [10] by Daniel Shanahan and Yuri Broze [28], and are available online.³ For other research that uses this dataset see [26, 27]. The chord forms in the *iRealPro* dataset include information about nineths and elevenths that are not considered in this study.

The subset of 394 Jazz-standards that consist of at most 40 chords was considered to train the models. 34.52% (136) of these pieces were parsable using the standard approach and 90.61% (357) pieces were parsable using the cyclic parsing approach described above. Less than 55% of the considered Jazz-standards therefore end in turnarounds.

6.2 Tree Accuracy Evaluation

We compare four models: (i) the proposed PACFG model that uses a representation of rules independent of key, (ii) its PCFG counterpart the rules of which are not independent of key, (iii) a baseline of randomly generated trees, and (iv) a *right-branching baseline* in which all constituents split into a constituent on the left and a terminal symbol on the right.

The models are trained on the 357 cyclic parsable sequences using minibatches of 8 sequences. They are evaluated on 13 pieces hand-annotated by the authors. We report the predicted tree accuracy. That is the precision of correctly predicted spans of internal tree nodes. A span of a tree node is defined as the start index of its leftmost leaf together with the end index of its rightmost leaf.

Figure 4 shows the means of the tree accuracies including 95% confidence intervals as error bars. The right-branching baseline performs at an accuracy level under 10%. The random baseline performs slightly better at an

² <https://irealpro.com>

³ https://musiccog.ohio-state.edu/home/index.php/iRb_Jazz_Corpus

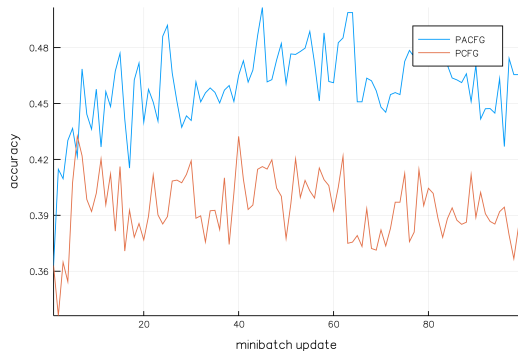


Figure 5. Predicted tree accuracy for each minibatch update. Note that the y-axis displays only values between 33% and 50%.

accuracy level of 15.35% Under a uniform prior, both the PACFG and the PCFG model perform at an accuracy level of 36.30% a priori of the data. As opposed to the trained PCFG model that only improves its performance by about 3% (in comparison to the uniform prior) reaching an accuracy of 39.43%, the trained PACFG model improves by about 10% (in comparison to the uniform prior) reaching an accuracy of 45.95%. The PACFG model was thus able to learn more from the data than the PCFG model. Note that since the PCFG model does not abstract the grammar rules from the concrete key wherein they are applied, the number of free parameters of the PCFG model is approximately 12 times higher than the number of free parameters of the APCFG model.

Despite the fact that the PACFG model learns key-independently, it is still much simpler than models that produce state-of-the-art parsing results in computational linguistics. In particular, state-of-the-art models in computational linguistics typically make use of conditioning information beyond the parent constituent categories used in the PACFG model—such as larger tree fragments, conditioning on heads and/or adjacent elements in the string, state-splitting, and other richer contextual information. We anticipate that the inclusion of similar structures into musical parsing models will lead to similar improvements in performance.

Figure 5 shows the mean predicted tree accuracies of the PACFG and the PCFG models for each minibatch update. Note that this figure is produced using a stochastic algorithm and is therefore inherently noisy. We see that the stochasticity of the inference algorithm leads to random jumps of the accuracy up to 0.5%. The models appear to do most of their learning in the first 10 minibatches.

6.3 Performance Diagnosis using Scale Degree Frequencies

Figure 6 shows the expected frequency of scale-degree use in the whole corpus. The scale degrees VI in major and III in minor are more frequently used by the model than expected. Because these scale degrees are substitutions for the first scale degrees and because they enable modulations

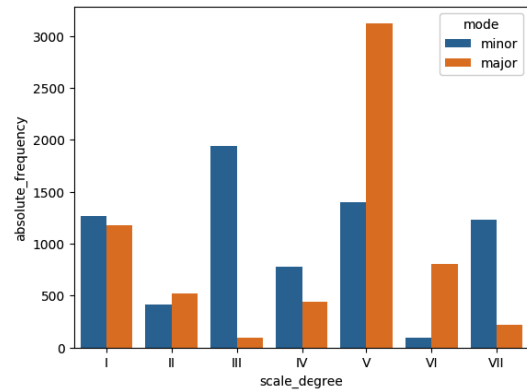


Figure 6. Expected usage of scale degrees to parse the full training dataset

into the relative key (e.g. from C major to A minor and vice versa), the model may be using them to alternate between relative keys. The prominence of the VII in minor keys is probably related to the fact that it has a dominant-seventh chord form. The model may be interpreting a I in major as a III in the relative minor key that is then prepared by the VII in minor. For example, the simple chord transition $G^7 C^\Delta$ would in this case be derived by

$$I_a \rightarrow III_a \rightarrow VII_a \quad III_a \rightarrow G^7 \quad III_a \rightarrow G^7 C^\Delta.$$

7. CONCLUSION AND FUTURE RESEARCH

The research presented here introduced a new general grammar and parsing framework tailored to the needs of music and showed how to perform inference for such a model.

Experiments show that in contrast to standard context-free models, the proposed model is able to learn characteristic structures of the observed data. To the best of our knowledge, this is the first computational approach that automatically performs hierarchical analyses of chord sequences and evaluates them on analyses by human experts.

This paper lays the groundwork for more advanced models of harmonic syntax. Our future research will in particular focus on expanding the dataset of hand-annotated expert analyses to provide significance tests of the performance comparison of different models, for example. Further studies can use the tools developed here to build models of unsupervised grammar induction, joint models of multiple musical levels of musical structure like harmony and rhythm, and models of musical structure that have more complex dependencies than those representable in simple tree structures.

8. ACKNOWLEDGEMENTS

Financial support for the research presented in this article has in part been provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the *Zukunftskonzept* at TU Dresden funded by the *Exzellenzinitiative* of the *Deutsche Forschungsgemeinschaft*.

9. REFERENCES

- [1] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM review*, 59(1):65–98, 2017.
- [2] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational Inference: A Review for Statisticians. *arXiv*, (arXiv:1601.00670), 2017.
- [3] Joshua T Goodman. *Parsing inside-out*. PhD thesis, 1998.
- [4] Mark Granroth-Wilding and Mark Steedman. A Robust Parser-Interpreter for Jazz Chord Sequences. *Journal of New Music Research*, 43(4):355–374, 10 2014.
- [5] W Bas De Haas. *Music information retrieval based on tonal harmony*. PhD thesis, 2012.
- [6] W Bas De Haas, Jos Pedro Magalhães, and Frans Wiering. Improving Audio Chord Transcription by Exploiting Harmonic and Metric Knowledge. *International Society for Music Information Retrieval Conference (ISMIR)*, (Ismir):295–300, 2012.
- [7] W Bas De Haas, Martin Rohrmeier, and Frans Wiering. Modeling Harmonic Similarity using a Generative Grammar of Tonal Harmony. *Proceedings of the Tenth International Conference on Music Information Retrieval (ISMIR)*, 2009.
- [8] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Musical Structural Analysis Database Based on Gttm. In *Proceedings of the 15th Conference of the International Society for Music Information Retrieval*, pages 325–330, 2014.
- [9] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.
- [10] David Brian Huron. *The Humdrum Toolkit: Reference Manual*. Center for Computer Assisted Research in the Humanities, 1994.
- [11] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakola, and Lawrence K Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37:183–233, 1999.
- [12] Jonah Katz. Harmonic Syntax of the Twelve-Bar Blues Form. *Music Perception: An Interdisciplinary Journal*, 35(2):165–192, 2017.
- [13] Phillip B Kirlin and David D Jensen. Using Supervised Learning to Uncover Deep Musical Structure. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 1770–1776, 2015.
- [14] Dan Klein and Christopher D. Manning. Parsing and Hypergraphs. *Proceedings of the 7th International Workshop on Parsing Technologies (IWPT-2001)*, (c):351372, 2001.
- [15] Stefan Koelsch, Martin Rohrmeier, Renzo Torrecuso, and Sebastian Jentschke. Processing of hierarchical syntactic structure in music. *Proceedings of the National Academy of Sciences*, 110(38):15443–15448, 2013.
- [16] Hendrik Vincent Koops, Jos Pedro Magalhães, and W. Bas de Haas. A functional approach to automatic melody harmonisation. In *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design - FARM '13*, page 47. ACM Press, 2013.
- [17] Kenichi Kurihara and Taisuke Sato. An Application of the Variational Bayesian Approach to Probabilistic Context-Free Grammars. 2004.
- [18] Martin Lange and Hans Leiß. To CNF or not to CNF - An Efficient Yet Presentable Version of the CYK Algorithm. *Informatica Didactica*, 8:1–21, 2009.
- [19] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. Cambridge, MA, 1983.
- [20] Mark Levine. *The jazz theory book*. Sher Music, 1995.
- [21] Eita Nakamura, Masatoshi Hamanaka, Keiji Hirata, and Kazuyoshi Yoshii. Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 276–280, 2016.
- [22] Markus Neuwirth and Martin Rohrmeier. Towards a syntax of the Classical cadence. In *What is a Cadence*, pages 287–338. 2015.
- [23] Martin Rohrmeier. A generative grammar approach to diatonic harmonic structure. *Proceedings SMC'07, 4th Sound and Music Computing Conference*, (July):11–13, 2007.
- [24] Martin Rohrmeier. Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music*, 5(1):35–53, 3 2011.
- [25] Martin Rohrmeier and Ian Cross. Tacit tonality : Implicit learning of context-free harmonic structure. In *Proceedings of the 7th Triennial Conference of European Society for the Cognitive Sciences of Music (ESCOM 2009) Jyväskylä, Finland*, number Escom, pages 443–452, 2009.
- [26] Keith Salley and Daniel T. Shanahan. Phrase Rhythm in Standard Jazz Repertoire: A Taxonomy and Corpus Study. *Journal of Jazz Studies*, 11(1):1, 2016.
- [27] Daniel Shanahan and Yuri Broze. Diachronic Changes in Jazz Harmony: A Cognitive Perspective. *Music Perception: An Interdisciplinary Journal*, 31(1):32–45, 2013.

- [28] Daniel Shanahan, Yuri Broze, and Richard Rodgers. A Diachronic Analysis of Harmonic Schemata in Jazz. In *Proceedings of the 12th International Conference on Music Perception and Cognition and the 8th Triennial Conference of the European Society for the Cognitive Sciences of Music*, pages 909–917, 2012.
- [29] Stuart M Shieber, Yves Schabes, and Fernando C.N. Pereira. Principles and Implementation of Deductive Parsing. *Journal of Logic Programming*, 1993.
- [30] Mark J. Steedman. A Generative Grammar for Jazz Chord Sequences. *Music Perception: An Interdisciplinary Journal*, 2(1):52–77, 1984.
- [31] Mark J Steedman. The blues and the abstract truth: Music and mental models. *Mental models in cognitive science: essays in honour of Phil Johnson-Laird*, pages 305–318, 1996.
- [32] Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

AN ENERGY-BASED GENERATIVE SEQUENCE MODEL FOR TESTING SENSORY THEORIES OF WESTERN HARMONY

Peter M. C. Harrison

Queen Mary University of London
Cognitive Science Research Group

Marcus T. Pearce

Queen Mary University of London
Cognitive Science Research Group

ABSTRACT

The relationship between sensory consonance and Western harmony is an important topic in music theory and psychology. We introduce new methods for analysing this relationship, and apply them to large corpora representing three prominent genres of Western music: classical, popular, and jazz music. These methods centre on a generative sequence model with an exponential-family energy-based form that predicts chord sequences from continuous features. We use this model to investigate one aspect of instantaneous consonance (harmonicity) and two aspects of sequential consonance (spectral distance and voice-leading distance). Applied to our three musical genres, the results generally support the relationship between sensory consonance and harmony, but lead us to question the high importance attributed to spectral distance in the psychological literature. We anticipate that our methods will provide a useful platform for future work linking music psychology to music theory.

1. INTRODUCTION

Music theorists and psychologists have long sought to understand how Western harmony may be shaped by natural phenomena universal to all humans [13, 27, 36]. Key to this work is the notion of *sensory consonance*, describing a sound’s natural pleasantness [32, 35, 38], and its inverse *sensory dissonance*, describing natural unpleasantness.

Sensory consonance has both *instantaneous* and *sequential* aspects. Instantaneous consonance is the consonance of an individual sound, whereas sequential consonance is a property of a progression between sounds.

Instantaneous sensory consonance primarily derives from *roughness* and *harmonicity*. Roughness is an unpleasant sensation caused by interactions between spectral components in the inner ear [8, 41], whereas harmonicity¹ is a pleasant percept elicited by a sound’s resemblance to the harmonic series [4, 20].

¹ Related concepts include *tonalness* [27], *toneness* [15], *fusion* [14, 36], *complex sonorousness* [29], and *multiplicity* [29].

Sequential sensory consonance is primarily determined by *spectral distance* and *voice-leading distance*. Spectral distance² describes how much a sound’s acoustic spectrum perceptually differs from neighbouring spectra [22–24, 27, 29]. Voice-leading distance³ describes how far notes in one chord have to move to produce the next chord [2, 39, 40]. Consonance is associated with low spectral and voice-leading distance.

Many Western harmonic conventions can be rationalized as attempts to increase pleasantness by maximizing sensory consonance. The major triad maximizes consonance by minimizing roughness and maximizing harmonicity; the circle of fifths maximizes consonance by minimizing spectral distance; tritone substitutions are consonant through voice-leading efficiency [39].

This idea – that Western harmony seeks to maximize sensory consonance – has a long history in music theory [31]. Its empirical support is surprisingly limited, however. The best evidence comes from research linking sensory consonance maximization to rules from music theory [15, 27, 39], but this work is constrained by the subjectivity and limited scope of music-theoretic textbooks.

A better approach is to bypass textbooks and analyse musical scores directly. Usefully, large datasets of digitised musical scores are now available, as are many computational models of consonance. However, statistically linking them is non-trivial. One could calculate distributions of consonance features, but this would give only limited causal insight into how these distributions arise. Better insight might be achieved by regressing transition probabilities against consonance features, but this approach is statistically problematic because of variance heterogeneity induced by the inevitable sparsity of the transition tables.

This paper presents a new statistical model developed for tackling this problem. The model is generative and feature-based, defining a probability distribution over symbolic sequences based on features derived from these sequences. Unlike previous feature-based sequence models, it is specialized for continuous features, making it well-suited to consonance modelling. Moreover, the model parameters are easily interpretable and have quantifiable un-

² Spectral distance is also known by its antonym *spectral similarity* [23]. *Pitch commonality* [29] is a similar concept. Psychological models of harmony and tonality in the auditory short-term memory (ASTM) tradition typically rely on some kind of spectral distance measure [1, 7, 17].

³ Voice-leading distance is termed *horizontal motion* in [2]. Parncutt’s notion of *pitch distance* [28, 29] is also conceptually similar to voice-leading distance.



certainty, enabling error-controlled statistical inference.

We use this new model to test sensory theories of harmony as follows. We fit the model to corpora of chord sequences from classical, popular, and jazz music, using psychological models of sensory consonance as features. We then compute feature importance metrics to quantify how different aspects of consonance constrain harmonic movement. This work constitutes the first corpus analysis comprehensively linking sensory consonance to harmonic practice.

2. METHODS

2.1 Representations

2.1.1 Input

Chord progressions are represented as sequences of pitch-class sets. Exact chord repetitions are removed, but changes of chord inversion are represented as repeated pitch-class sets.

2.1.2 Pitch-Class Spectra

Some of our features use *pitch-class spectra* as defined in [22, 24]. A pitch-class spectrum is a continuous function that describes *perceptual weight* as a function of pitch class (p_c). Perceptual weight is the strength of perceptual evidence for a given pitch class being present. Pitch classes (p_c) take values in the interval $[0, 12)$ and relate to frequency (f , Hz scale) as follows:

$$p_c = \left[9 + 12 \log_2 \left(\frac{f}{440} \right) \right] \bmod 12. \quad (1)$$

Pitch-class sets are transformed to pitch-class spectra by expanding each pitch class into its implied harmonics. Pitch classes are modelled as harmonic complex tones with 12 harmonics, after [22]. The j th harmonic in a pitch class has level $j^{-\rho}$, where ρ is the roll-off parameter ($\rho > 0$). Partial is represented by Gaussians with mass equal to partial level, mean equal to partial pitch class, and standard deviation σ . Perceptual weights combine additively.

Formally, $W(p_c, X)$ defines a pitch-class spectrum, returning the perceptual weight at pitch-class p_c for an input pitch-class set $X = \{x_1, x_2, \dots, x_m\}$:

$$W(p_c, X) = \sum_{i=1}^m T(p_c, x_i). \quad (2)$$

Here i indexes the pitch classes, and $T(p_c, x)$ is the contribution of a harmonic complex tone with fundamental pitch class x to an observation at pitch class p_c :

$$T(p_c, x) = \sum_{j=1}^{12} g(p_c, j^{-\rho}, h(x, j)). \quad (3)$$

Now j indexes the harmonics, $g(p_c, l, p_x)$ is the contribution from a harmonic with level l and pitch-class p_x to an observation at pitch-class p_c ,

$$g(p_c, l, p_x) = \frac{l}{\sigma\sqrt{2\pi}} \exp \left(-\frac{1}{2} \left(\frac{d(p_c, p_x)}{\sigma} \right)^2 \right), \quad (4)$$

$d(p_x, p_y)$ is the distance between two pitch classes p_x and p_y ,

$$d(p_x, p_y) = \min(|p_x - p_y|, 12 - |p_x - p_y|), \quad (5)$$

and $h(x, j)$ is the pitch class of the j th partial of a harmonic complex tone with fundamental pitch class x :

$$h(x, j) = (x + 12 \log_2 j) \bmod 12. \quad (6)$$

ρ and σ are set to 0.75 and 0.0683 after [22].

2.2 Features

2.2.1 Spectral Distance

Spectral distance is operationalised using the psychological model of [22, 24]. The spectral distance between two pitch-class sets X, Y is defined as 1 minus the continuous cosine similarity between the two pitch-class spectra:

$$D(X, Y) = 1 - \frac{\int_0^{12} W(z, X)W(z, Y) dz}{\sqrt{\int_0^{12} W(z, X)^2 dz} \sqrt{\int_0^{12} W(z, Y)^2 dz}} \quad (7)$$

with W as defined in Equation 2. The measure takes values in the interval $[0, 1]$, where 0 indicates maximal similarity and 1 indicates maximal divergence.

2.2.2 Harmonicity

Our harmonicity model is inspired by the template-matching algorithms of [21] and [29]. The model simulates how listeners search the auditory spectrum for occurrences of harmonic spectra. These inferred harmonic spectra are termed *virtual pitches*. High harmonicity corresponds to a strong virtual pitch percept.

Our model differs from previous models in two ways. First, it uses a pitch-class representation, not a pitch representation. This makes it voicing-invariant and hence more suitable for modelling pitch-class sets. Second, it takes into account the strength of all virtual pitches in the spectrum, not just the strongest virtual pitch.

The model works as follows. The *virtual pitch-class spectrum* Q defines the spectral similarity of the pitch-class set X to a harmonic complex tone with pitch class p_c :

$$Q(p_c, X) = D(p_c, X) \quad (8)$$

with D as defined in Equation 7. Normalising Q to unit mass produces Q' :

$$Q'(p_c, X) = \frac{Q(p_c, X)}{\int_0^{12} Q(y, X) dy}. \quad (9)$$

Previous models compute harmonicity by taking the peak of this spectrum. In our experience this works for small

chords but not for larger chords, where several virtual pitches need to be accounted for. We therefore instead compute a spectral peakiness measure. Several such measures are possible, but here we use Kullback-Leibler divergence from a uniform distribution. $H(X)$, the harmonicity of a pitch-class set X , can therefore be written as follows:

$$H(X) = \int_0^{12} Q'(y, X) \log_2 (12 Q'(y, X)) dy. \quad (10)$$

Harmonicity has a large negative correlation with the number of notes in a chord. Some correlation is expected, but not to this degree: the harmonicity model considers a tritone (the least consonant two-note chord) to be more consonant than a major triad (the most consonant three-note chord). We therefore separate the two phenomena by adding a ‘chord size’ feature, corresponding to the number of notes in a given chord, and rescaling harmonicity to zero mean and unit variance across all chords with a given chord size.

2.2.3 Roughness

Roughness has traditionally been considered to be an important contributor to sensory consonance, though some recent research disputes its importance [20]. We originally planned to include roughness in our model, but then discovered that the phenomenon is highly sensitive to chord voicing. Since the voicing of a pitch-class set is undefined, its roughness is therefore unpredictable. Roughness is therefore not modelled in the present study.

2.2.4 Voice-Leading Distance

A *voice leading* connects the individual notes in two pitch-class sets to form simultaneous melodies [39]. Pitch-class sets of different sizes can be connected by allowing pitch classes to participate in multiple melodies. *Voice-leading distance* is an aggregate measure of the resulting melodic distance. We operationalise voice-leading distance using [39]’s geometric model.

Consider two pitch-class sets $X = \{x_1, x_2, \dots, x_m\}$ and $Y = \{y_1, y_2, \dots, y_n\}$. A voice-leading between X and Y can be written $A \rightarrow B$ where $A = (a_1, a_2, \dots, a_N)$, $B = (b_1, b_2, \dots, b_N)$, and the following holds: if $x \in X$ then $x \in A$, if $y \in Y$ then $y \in B$, if $a \in A$ then $a \in X$, if $b \in B$ then $b \in Y$, and $n \leq N$.

The distance of the voice leading $A \rightarrow B$ is denoted $V(A, B)$ and uses the *taxicab* norm:

$$V(A, B) = \sum_{i=1}^N d(a_i, b_i) \quad (11)$$

with $d(a_i, b_i)$ as defined in Equation 5.

The voice-leading distance between pitch-class sets X, Y is then defined as the smallest value of $V(A, B)$ for all legal A, B . This minimal distance can be efficiently computed using the algorithm described in [39].

2.2.5 Summary

This section defined three sensory consonance features. These included one instantaneous measure (harmonicity) and two sequential measures (spectral distance, voice-leading distance). Harmonicity correlated strongly with chord size, which could have confounded our analyses. We therefore controlled for chord size by normalising harmonicity for each chord size and including chord size as a feature.

2.3 Statistical Model

2.3.1 Overview

The statistical model is *generative*, defining a probability distribution over chord sequences (e.g. [12, 25, 33]). It is *feature-based*, using features of the chord and its context to predict chord probabilities (e.g. [12]). It is *energy-based*, defining scalar energies for each feature configuration which are then transformed and normalised to produce the final probability distribution (e.g. [3, 10, 30]). It is *exponential-family* in that the energy function is a linear function of the feature vector (e.g. [10, 30]). Informally, the model might be said to generalise linear regression to symbolic sequences.

2.3.2 Form

Let \mathcal{A} denote the set of all possible chords, and let e_0^n denote a chord sequence of length n , where e_0 is always a generic start symbol. Let $e_i \in \mathcal{A}$ denote the i th chord and e_i^j the subsequence $(e_i, e_{i+1}, \dots, e_j)$. Let \mathbf{w} be the weight vector that parametrises the model.

The probability of a chord sequence is factorised into a chain of conditional chord probabilities.

$$P(e_0^n | \mathbf{w}) = \prod_{i=1}^n P(e_i | e_0^{i-1}, \mathbf{w}) \quad (12)$$

These are given energy-based expressions:

$$P(e_i | e_0^{i-1}, \mathbf{w}) = \frac{\exp(-E(e_0^{i-1}, e_i, \mathbf{w}))}{Z(e_0^{i-1}, \mathbf{w})} \quad (13)$$

where E is the *energy function* and Z is the *partition function*. Z normalises the probability distribution to unit mass:

$$Z(e_0^{i-1}, \mathbf{w}) = \sum_{x \in \mathcal{A}} \exp(-E(e_0^{i-1}, x, \mathbf{w})). \quad (14)$$

High E corresponds to low probability. E is defined as a sum of *feature functions*, f_j , weighted by $-\mathbf{w}$:

$$E(e_0^{i-1}, x, \mathbf{w}) = - \sum_{j=1}^m f_j(e_0^{i-1} :: x) w_j \quad (15)$$

where w_j is the j th component of \mathbf{w} , m is the dimensionality of \mathbf{w} , equalling the number of feature functions f_j , and $e_0^{i-1} :: x$ is the concatenation of e_0^{i-1} and $x \in \mathcal{A}$.

Feature functions measure a property of the last element of a sequence. Our feature functions are chord size,

harmonicity, spectral distance, and voice-leading distance. Chord size and harmonicity are context-independent, whereas spectral and voice-leading distance relate the last chord to the penultimate chord. When the penultimate chord is undefined, mean values are imputed for spectral and voice-leading distance, with the mean computed over all possible chord transitions.

2.3.3 Estimation

The model is parametrised by the weight vector \mathbf{w} . This weight vector is optimised using maximum-likelihood estimation on a corpus of sequences, as follows.

Let $e_{0,k}^{n_k}$ denote the k th sequence from a corpus of size N , where n_k is the sequence's length. The negative log-likelihood of the weight vector \mathbf{w} with respect to the corpus is then

$$C(\mathbf{w}) = - \sum_{k=1}^N \sum_{i=1}^{n_k} \log P(e_{i,k} | e_{0,k}^{i-1}, \mathbf{w}). \quad (16)$$

After some algebra, the gradient can be written

$$\frac{dC}{d\mathbf{w}} = \sum_{k=1}^N \sum_{i=1}^{n_k} \frac{Z'(e_{0,k}^{i-1}, \mathbf{w})}{Z(e_{0,k}^{i-1}, \mathbf{w})} - \mathbf{f}(e_{0,k}^i) \quad (17)$$

where

$$Z'(e_{0,k}^{i-1}, \mathbf{w}) = \sum_{x \in \mathcal{A}} \mathbf{f}(e_{0,k}^{i-1} :: x) \exp(-E(e_{0,k}^{i-1}, x, \mathbf{w})) \quad (18)$$

and \mathbf{f} is the vector of feature functions. This expression can be plugged into a generic optimiser to find a weight vector minimising the negative log-likelihood. The present work used the BFGS optimiser [37].

2.3.4 Feature Importance

This section introduces three complementary feature importance measures. These are *weight*, *explained entropy*, and *unique explained entropy*.

Weight describes a feature's relationship to chord probability. The weight for a feature function f_j is the parameter w_j , corresponding to (minus) the change in the energy function E in response to a one-unit change in the feature function f_j (Equation 15). Weight is a signed feature importance measure: the sign dictates whether the model prefers high (positive weight) or low (negative weight) feature values, and the magnitude dictates the strength of preference. To aid weight comparability between features, feature functions are scaled to unit variance over the set of all possible chord transitions.

Dividing the cost function (Equation 16) by the number of chords in the corpus ($\sum_{k=1}^N n_k$) gives an estimate of *cross entropy* in units of *nats*. Cross entropy measures chord-wise unpredictability with respect to a given model. From it we define two further measures: *explained entropy* and *unique explained entropy*.

Explained entropy for a feature f_j is computed by comparing cross entropy estimates for two models: a model

trained using feature f_j and a null model trained with no features. Explained entropy is the difference between the two cross entropies. Higher values indicate that the feature explains a lot of structure in the corpus.

Unique explained entropy for a feature f_j is the amount that cross entropy changes when feature f_j is removed from the full feature set. It measures the unique explanatory power of a feature while controlling for other features.

2.3.5 Related Work

The literature contains several alternative approaches for feature-based modelling of chord sequences. One is the *multiple viewpoint method* [11, 12]. However, this method is specialised for discrete features, not the continuous features required for consonance modelling. A second alternative is the *maximum-entropy* approach of [10, 30]. This approach has some formal similarities with the present work, but its binary feature functions are incompatible with our continuous features. A third possibility is the *feature-based dynamic networks* of [33]; however, these networks would need substantial modification to represent the kind of feature dependencies required here.

2.4 Corpora

Our corpora represent three musical genres: classical music (1,022 movements/pieces), popular music (739 pieces), and jazz music (1,186 pieces). The classical corpus was compiled from *KernScores* [34], including ensemble music and keyboard music from several major composers of common-practice tonal music (Bach, Haydn, Mozart, Beethoven, Chopin). Chord labels were obtained using the algorithm of [26] with an expanded chord dictionary, and with segment boundaries co-located with metrical beat locations as estimated from time signatures. Chord inversions were identified as the lowest-pitch chord tone in the harmonic segment being analysed. The popular and jazz corpora corresponded to publicly available datasets: the McGill Billboard corpus [6] and the *iRB* corpus [5].

2.5 Efficiency

Computation can be reduced by identifying repeated terms in the cost and cost gradient (Equations 16, 17). These repeated terms only need to be evaluated once. Our feature functions never look back further than the previous chord, and they are invariant to chord transposition; this means that repeated terms occur whenever a chord pair is repeated at some transposition. Collapsing over these repetitions reduces computation by a factor of 20–100 for our corpora.

2.6 Numeric Integration

The features related to pitch-class spectra all use integration. These integrals are numerically approximated using the rectangle rule with 1,200 subintervals, after [24].

2.7 Software

The statistical model was implemented in R and C++; source code is available from the authors on request.

3. RESULTS

3.1 Corpus level

Figure 1 plots feature importances for the three consonance measures: harmonicity (normalised by chord size), spectral distance, and voice-leading distance. Analyses are split by musical corpus, and confidence intervals are calculated using nonparametric bootstrapping [9].

3.1.1 Importance by Feature

All the consonance features contribute to harmonic structure in some way. The order of feature importance is fairly consistent between genres and importance measures. Broadly speaking, voice-leading distance is most important, followed by harmonicity, then spectral distance.

3.1.2 Importance by Corpus

Harmonicity is particularly important for popular music, less so for classical, and least for jazz. Spectral distance is most important for classical music, less so for popular, and unimportant for jazz.

The relative importance of voice-leading distance depends on the measure used: it scores highly on explained entropy, but less on weight and unique explained entropy. This may be because voice-leading distance and chord size capture some common information: moving from a small chord to a large chord typically involves a large voice-leading distance. If we wish to assess the unique effect of voice-leading distance, we can look at weight and unique explained entropy: these measures tell us that voice-leading distance is most important for jazz music, less for classical music, and least for popular music.

3.1.3 Signs of Weights

The sign of a feature weight determines whether the model prefers positive or negative values of the feature. The observed feature signs are all consistent with theory. Harmonicity has a positive weight for all genres, indicating that harmonicity is universally promoted. Spectral distance and voice-leading distance both have negative weights, indicating preference for lower values of these features.

3.2 Composition Level

We also explored the application of these techniques to individual compositions (Figure 2). While the composition-level analyses reflect the same trends as the corpus-level analyses (Figure 1), they also reveal substantial overlap between the corpora. We assessed the extent of this overlap by training a generic machine-learning classifier to predict genre from the complete set of feature importance measures. Our classifier was a random forest model trained using the `randomForest` package in R [18], with 2,000 trees and four variables sampled at each split. Performance was assessed using 10-fold cross-validation repeated and averaged over 10 runs, resulting in a classification accuracy of 86% and a kappa statistic of .79. This indicates that genre differences in sensory consonance are moderately salient, even at the composition level.

4. CONCLUSION

This paper introduces new methods for testing relationships between sensory consonance and Western harmony. The methods centre on a new statistical model that predicts symbolic sequences using continuous features. We demonstrate these methods through application to three corpora representing classical, popular, and jazz music.

The results strongly support theoretical relationships between sensory consonance and harmonic structure. The three aspects of sensory consonance tested – harmonicity, spectral distance, and voice-leading distance – all predicted harmonic movement. Not all aspects were equally important, however. Spectral distance performed poorly, particularly in jazz. This is interesting given the high importance attributed to spectral distance in recent psychological literature [1, 7, 22]. Harmonicity performed well in popular music, but less so in classical and jazz. In contrast, voice-leading distance performed consistently well.

The corpus analyses deserve further development. It would be worth probing the true universality of sensory consonance by exploring a broader range of styles and using more refined stylistic categories, possibly at the level of the composer. The validity of the classical analyses could also be improved through more principled sampling [19] and manual chord-labelling [16].

The three feature importance measures provide useful complementary perspectives, but it is unnecessary to plot each one every time. In future we recommend inspecting the weights to check whether a feature is promoted or avoided, but then plotting just unique explained entropy. Unique explained entropy is preferable to weight because its units are well-defined, and preferable to explained entropy because it controls for other features, thereby providing a better handle on causality.

We focused on interpreting the statistical model through feature importance measures, but an alternative strategy would be to use the model to generate chord sequences for subjective evaluation. This route lacks the objectivity of feature-importance analysis, but it would give a uniquely intuitive perspective on what the model has learned.

The modelling techniques could be developed further. An important limitation of the current model is the linearity of the energy function, which restricts it to monotonic feature effects. A polynomial energy function would address this problem. It would also be interesting to develop the psychological features further, perhaps adding echoic memory to the spectral distance measure [17], and introducing an octave-generalised roughness measure.

Despite these limitations, we believe that the current results have important implications for our understanding of Western tonal harmony. In particular, the results imply that voice-leading efficiency is a better candidate for a harmonic universal than spectral similarity. This result is important for music psychology, where voice-leading efficiency is relatively underemphasised compared to harmonicity and spectral similarity (though see [2, 27, 39]). Future psychological work may wish to re-examine the role of voice-leading efficiency in harmony perception.

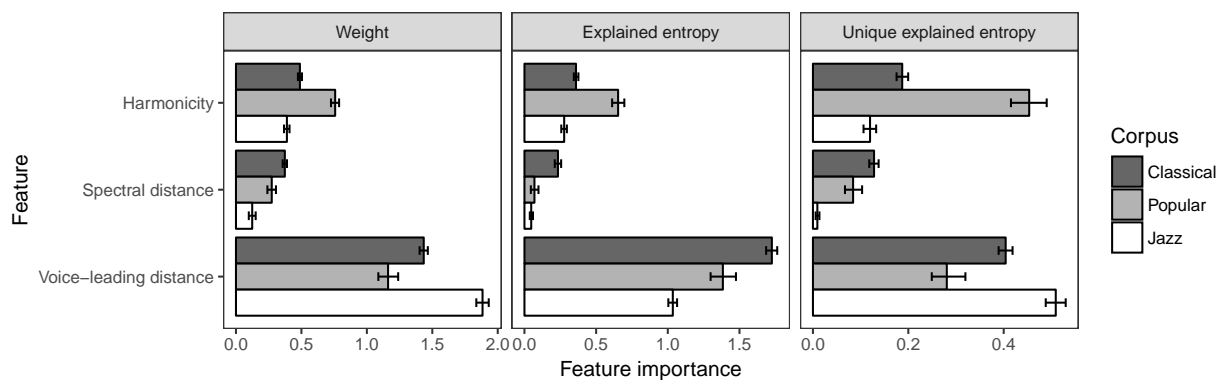


Figure 1. Measures of feature importance as a function of musical corpus. These measures are calculated from statistical models trained on the corpus level. Error bars represent 99% confidence intervals estimated by nonparametric bootstrapping [9]. Signs of feature weights are reversed for spectral distance and voice-leading distance, so that positive weights correspond to consonance maximisation.

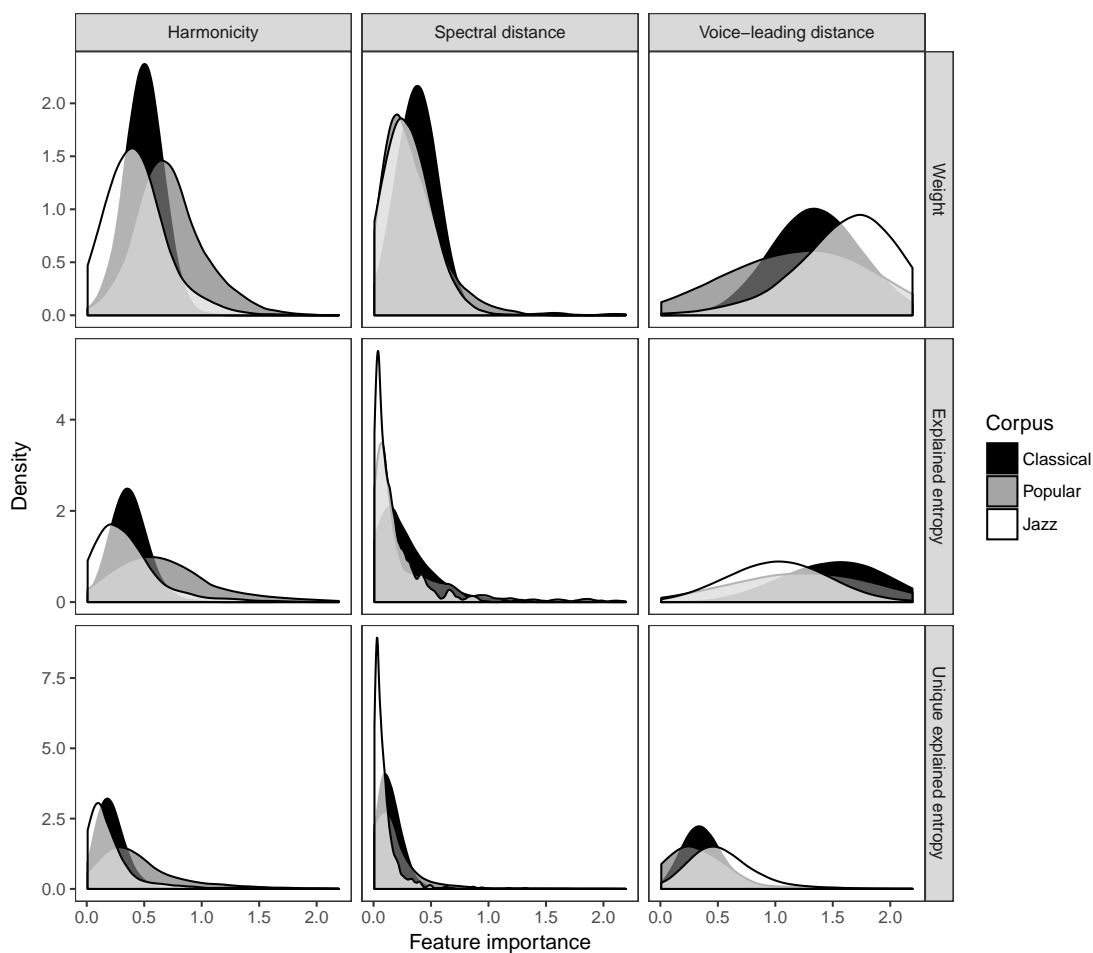


Figure 2. Distributions of feature importance measures as calculated for individual compositions within the three corpora. Distributions are represented by Epanechnikov kernel density functions. Signs of feature weights are reversed for spectral distance and voice-leading distance, so that positive weights correspond to consonance maximisation.

5. ACKNOWLEDGEMENTS

The authors would like to thank Emmanouil Benetos and Matthew Purver for useful feedback and advice regarding this project. PH is supported by a doctoral studentship from the EPSRC and AHRC Centre for Doctoral Training in Media and Arts Technology (EP/L01632X/1).

6. REFERENCES

- [1] Emmanuel Bigand, Charles Delbé, Bénédicte Poulin-Charronnat, Marc Leman, and Barbara Tillmann. Empirical evidence for musical syntax processing? Computer simulations reveal the contribution of auditory short-term memory. *Frontiers in Systems Neuroscience*, 8, 2014.
- [2] Emmanuel Bigand, Richard Parncutt, and Fred Lerda. Perception of musical tension in short chord sequences: The influence of harmonic function, sensory dissonance, horizontal motion, and musical training. *Perception & Psychophysics*, 58(1):124–141, 1996.
- [3] Nicolas Boulanger-Lewandowski, Pascal Vincent, and Yoshua Bengio. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proc. of the 29th International Conference on Machine Learning (ICML-12)*, 2012.
- [4] Daniel L. Bowling and Dale Purves. A biological rationale for musical consonance. *Proceedings of the National Academy of Sciences*, 112(36):11155–11160, 2015.
- [5] Yuri Broze and Daniel Shanahan. Diachronic changes in jazz harmony: A cognitive perspective. *Music Perception*, 31(1):32–45, 2013.
- [6] John Ashley Burgoyne. *Stochastic Processes & Database-Driven Musicology*. PhD thesis, McGill University, Montréal, Québec, Canada, 2011.
- [7] Tom Collins, Barbara Tillmann, Frederick S. Barrett, Charles Delbé, and Petr Janata. A combined model of sensory and cognitive representations underlying tonal expectations in music: From audio signals to behavior. *Psychological Review*, 121(1):33–65, 2014.
- [8] P. Daniel and R. Weber. Psychoacoustical roughness: Implementation of an optimized model. *Acta Acustica united with Acustica*, 83(1):113–123, 1997.
- [9] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. Chapman & Hall, Boca Raton, FL, 1993.
- [10] Gaëtan Hadjeres, Jason Sakellariou, and François Pachet. Style imitation and chord invention in polyphonic music with exponential families. <http://arxiv.org/abs/1609.05152>, 2016.
- [11] Peter M. C. Harrison and Marcus T. Pearce. A statistical-learning model of harmony perception. In *Proc. of DMRN+12: Digital Music Research Network One-Day Workshop*, page 15, London, UK, 2017.
- [12] Thomas Hedges and Geraint A. Wiggins. The prediction of merged attributes with multiple viewpoint systems. *Journal of New Music Research*, 45(4):314–332, 2016.
- [13] Hermann Helmholtz. *On the sensations of tone*. Dover, New York, NY, 1954. First published in 1863; translated by Alexander J. Ellis.
- [14] David Huron. Tonal consonance versus tonal fusion in polyphonic sonorities. *Music Perception*, 9(2):135–154, 1991.
- [15] David Huron. Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception*, 19(1):1–64, 2001.
- [16] Nori Jacoby, Naftali Tishby, and Dmitri Tymoczko. An information theoretic approach to chord categorization and functional harmony. *Journal of New Music Research*, 44(3):219–244, 2015.
- [17] Marc Leman. An auditory model of the role of short-term memory in probe-tone ratings. *Music Perception*, 17(4):481–509, 2000.
- [18] Andy Liaw and Matthew Wiener. Classification and regression by randomForest. *R news*, 2(3):18–22, 2002.
- [19] Justin London. Building a representative corpus of classical music. *Music Perception*, 31(1):68–90, 2013.
- [20] Josh H. McDermott, Andriana J. Lehr, and Andrew J. Oxenham. Individual differences reveal the basis of consonance. *Current Biology*, 20(11):1035–1041, 2010.
- [21] Andrew J. Milne. *A computational model of the cognition of tonality*. PhD thesis, The Open University, Milton Keynes, UK, 2013.
- [22] Andrew J. Milne and Simon Holland. Empirically testing Tonnetz, voice-leading, and spectral models of perceived triadic distance. *Journal of Mathematics and Music*, 10(1):59–85, 2016.
- [23] Andrew J. Milne, Robin Laney, and David Sharp. A spectral pitch class model of the probe tone data and scalar tonality. *Music Perception*, 32(4):364–393, 2015.
- [24] Andrew J. Milne, William A. Sethares, Robin Laney, and David B. Sharp. Modelling the similarity of pitch collections with expectation tensors. *Journal of Mathematics and Music*, 5(1):1–20, 2011.
- [25] Jean-Francois Paiement, Douglas Eck, and Samy Bengio. A probabilistic model for chord progressions. In *Proc. of the 6th International Conference on Music Information Retrieval*, London, UK, 2005.

- [26] Bryan Pardo and William P. Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, 2002.
- [27] Richard Parncutt. *Harmony: A psychoacoustical approach*. Springer-Verlag, Berlin, Germany, 1989.
- [28] Richard Parncutt and Graham Hair. Consonance and dissonance in music theory and psychology: Disentangling dissonant dichotomies. *Journal of Interdisciplinary Music Studies*, 5(2):119–166, 2011.
- [29] Richard Parncutt and Hans Strasburger. Applying psychoacoustics in composition: “Harmonic” progressions of “nonharmonic” sonorities. *Perspectives of New Music*, 32(2):88–129, 1994.
- [30] Jeremy Pickens and Costas Iliopoulos. Markov random fields and maximum entropy modeling for music information retrieval. In *Proc. of the 6th International Conference on Music Information Retrieval*, pages 207–214, London, UK, 2005.
- [31] Jean-Philippe Rameau. *Treatise on harmony*. Jean-Baptiste-Christophe Ballard, Paris, France, 1722.
- [32] Pascaline Regnault, Emmanuel Bigand, and Mireille Besson. Different brain mechanisms mediate sensitivity to sensory consonance and harmonic context: Evidence from auditory event-related brain potentials. *Journal of Cognitive Neuroscience*, 13(2):241–255, 2001.
- [33] Martin A. Rohrmeier and Thore Graepel. Comparing feature-based models of harmony. In *Proc. of the 9th International Symp. on Computer Music Modeling and Retrieval (CMMR)*, pages 357–370, London, UK, 2012.
- [34] C. S. Sapp. Online database of scores in the Humdrum file format. In *Proc. of the 6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, pages 664–665, 2005.
- [35] E. Glenn Schellenberg and Laurel J. Trainor. Sensory consonance and the perceptual similarity of complex-tone harmonic intervals: Tests of adult and infant listeners. *Journal of the Acoustical Society of America*, 100(5):3321–3328, 1996.
- [36] Carl Stumpf. *The Origins of Music*. Oxford University Press, Oxford, UK, 2012. First published in 1911; translated by David Trippett.
- [37] Wenyu Sun and Ya-Xiang Yuan. *Optimization Theory and Methods: Nonlinear Programming*. Springer Science & Business Media, New York, NY, 2006.
- [38] Laurel J. Trainor, Christine D. Tsang, and Vivian H. W. Cheung. Preference for sensory consonance in 2- and 4-month-old infants. *Music Perception*, 20(2):187–194, 2002.
- [39] Dmitri Tymoczko. The geometry of musical chords. *Science*, 313(5783):72–74, 2006.
- [40] Dmitri Tymoczko. *A Geometry of Music*. Oxford University Press, New York, NY, 2011.
- [41] Václav Vencovský. Roughness prediction based on a model of cochlear hydrodynamics. *Archives of Acoustics*, 41(2):189–201, 2016.

AUTOMATIC, PERSONALIZED, AND FLEXIBLE PLAYLIST GENERATION USING REINFORCEMENT LEARNING

Shun-Yao Shih

National Taiwan University
shunyaoshih@gmail.com

Heng-Yu Chi

KKBOX Inc., Taipei, Taiwan
henrychi@kkbox.com

ABSTRACT

Songs can be well arranged by professional music curators to form a riveting playlist that creates engaging listening experiences. However, it is time-consuming for curators to timely rearrange these playlists for fitting trends in future. By exploiting the techniques of deep learning and reinforcement learning, in this paper, we consider music playlist generation as a language modeling problem and solve it by the proposed attention language model with policy gradient. We develop a systematic and interactive approach so that the resulting playlists can be tuned flexibly according to user preferences. Considering a playlist as a sequence of words, we first train our attention RNN language model on baseline recommended playlists. By optimizing suitable imposed reward functions, the model is thus refined for corresponding preferences. The experimental results demonstrate that our approach not only generates coherent playlists automatically but is also able to flexibly recommend personalized playlists for diversity, novelty and freshness.

1. INTRODUCTION

Professional music curators or DJs are usually able to carefully select, order, and form a list of songs which can give listeners brilliant listening experiences. For a music radio with a specific topic, they can collect songs related to the topic and sort in a smooth context. By considering preferences of users, curators can also find what they like and recommend them several lists of songs. However, different people have different preferences toward diversity, popularity, and etc. Therefore, it will be great if we can refine playlists based on different preferences of users on the fly. Besides, as online music streaming services grow, there are more and more demands for efficient and effective music playlist recommendation. Automatic and personalized music playlist generation thus becomes a critical issue.

However, it is unfeasible and expensive for editors to daily or hourly generate suitable playlists for all users based on their preferences about trends, novelty, diversity,

etc. Therefore, most of previous works try to deal with such problems by considering some particular assumptions. McFee *et al.* [14] consider playlist generation as a language modeling problem and solve it by adopting statistical techniques. Unfortunately, statistical method does not perform well on small datasets. Pampalk *et al.* [16] generate playlists by exploiting explicit user behaviors such as skipping. However, for implicit user preferences on playlists, they do not provide a systematic way to handle it.

As a result, for generating personalized playlists automatically and flexibly, we develop a novel and scalable music playlist generation system. The system consists of three main steps. First, we adopt Chen *et al.*'s work [4] to generate baseline playlists based on the preferences of users about songs. In details, given the relationship between users and songs, we construct a corresponding bipartite graph at first. With the users and songs graph, we can calculate embedding features of songs and thus obtain the baseline playlist for each songs by finding their k-nearest neighbors. Second, by formulating baseline playlists as sequences of words, we can pretrain RNN language model (RNN-LM) to obtain better initial parameters for the following optimization, using policy gradient reinforcement learning. We adopt RNN-LM because not only RNN-LM has better ability of learning information progresses than traditional statistical methods in many generation tasks, but also neural networks can be combined with reinforcement learning to achieve better performances [10]. Finally, given preferences from user profiles and the pretrained parameters, we can generate personalized playlists by exploiting techniques of policy gradient reinforcement learning with corresponding reward functions. Combining these training steps, the experimental results show that we can generate personalized playlists to satisfy different preferences of users with ease.

Our contributions are summarized as follows:

- We design an automatic playlist generation framework, which is able to provide timely recommended playlists for online music streaming services.
- We remodel music playlist generation into a sequence prediction problem using RNN-LM which is easily combined with policy gradient reinforcement learning method.
- The proposed method can flexibly generate suitable personalized playlists according to user profiles us-



© Shun-Yao Shih, Heng-Yu Chi. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Shun-Yao Shih, Heng-Yu Chi. "automatic, personalized, and flexible playlist generation using reinforcement learning", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

ing corresponding optimization goals in policy gradient.

The rest of this paper is organized as follows. In Section 2, we introduce several related works about playlist generation and recommendation. In Section 3, we provide essential prior knowledge of our work related to policy gradient. In Section 4, we introduce the details of our proposed model, attention RNN-LM with concatenation (AC-RNN-LM). In Section 5, we show the effectiveness of our method and conclude our work in Section 6.

2. RELATED WORK

Given a list of songs, previous works try to rearrange them for better song sequences [1,3,5,12]. First, they construct a song graph by considering songs in playlist as vertices, and relevance of audio features between songs as edges. Then they find a Hamiltonian path with some properties, such as smooth transitions of songs [3], to create new sequencing of songs. User feedback is also an important consideration when we want to generate playlists [6, 7, 13, 16]. By considering several properties, such as tempo, loudness, topics, and artists, of users' favorite played songs recently, authors of [6, 7] can thus provide personalized playlist for users based on favorite properties of users. Pampalk *et al.* [16] consider skip behaviors as negative signals and the proposed approach can automatically choose the next song according to audio features and avoid skipped songs at the same time. Mailliet *et al.* [13] provides a more interactive way to users. Users can manipulate weights of tags to express high-level music characteristics and obtain corresponding playlists they want. To better integrate user behavior into playlist generation, several works are proposed to combine playlist generation algorithms with the techniques of reinforcement learning [11, 20]. Xing *et al.* first introduce exploration into traditional collaborative filtering to learn preferences of users. Liebman *et al.* take the formulation of Markov Decision Process into playlist generation framework to design algorithms that learn representations for preferences of users based on hand-crafted features. By using these representations, they can generate personalized playlist for users.

Beyond playlist generation, there are several works adopting the concept of playlist generation to facilitate recommendation systems. Given a set of songs, Vargas *et al.* [18] propose several scoring functions, such as diversity and novelty, and retrieve the top-K songs with higher scores for each user as the resulting recommended list of songs. Chen *et al.* [4] propose a query-based music recommendation system that allow users to select a preferred song as a seed song to obtain related songs as a recommended playlist.

3. POLICY GRADIENT REINFORCEMENT LEARNING

Reinforcement learning has got a lot of attentions from public since Silver *et al.* [17] proposed a general reinforcement learning algorithm that could make an agent achieve

superhuman performance in many games. Besides, reinforcement learning has been successfully applied to many other problems such as dialogue generation modeled as Markov Decision Process (MDP).

A Markov Decision Process is usually denoted by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where

- \mathcal{S} is a set of states
- \mathcal{A} is a set of actions
- $\mathcal{P}(s, a, s') = \Pr[s'|s, a]$ is the transition probability that action a in state s will lead to state s'
- $\mathcal{R}(s, a) = \mathbb{E}[r|s, a]$ is the expected reward that an agent will receive when the agent does action a in state s .
- $\gamma \in [0, 1]$ is the discount factor representing the importance of future rewards

Policy gradient is a reinforcement learning algorithm to solve MDP problems. Modeling an agent with parameters θ , the goal of this algorithm is to find the best θ of a policy $\pi_\theta(s, a) = \Pr[a|s, \theta]$ measured by average reward per time-step

$$J(\theta) = \sum_{s \in \mathcal{S}} d^{\pi_\theta}(s) \sum_{a \in \mathcal{A}} \pi_\theta(s, a) \mathcal{R}(s, a) \quad (1)$$

where $d^{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ .

Usually, we assume that $\pi_\theta(s, a)$ is differentiable with respect to its parameters θ , i.e., $\frac{\partial \pi_\theta(s, a)}{\partial \theta}$ exists, and solve this optimization problem Eqn (1) by gradient ascent. Formally, given a small enough α , we update its parameters θ by

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \quad (2)$$

where

$$\begin{aligned} \nabla_\theta J(\theta) &= \sum_{s \in \mathcal{S}} d^{\pi_\theta}(s) \sum_{a \in \mathcal{A}} \pi_\theta(s, a) \nabla_\theta \pi_\theta(s, a) \mathcal{R}(s, a) \\ &= \mathbb{E}[\nabla_\theta \pi_\theta(s, a) \mathcal{R}(s, a)] \end{aligned} \quad (3)$$

4. THE PROPOSED MODEL

The proposed model consists of two main components. We first introduce the structure of the proposed RNN-based model in Section 4.1. Then in Section 4.2, we formulate the problem as a Markov Decision Process and solve the formulated problem by policy gradient to generate refined playlists.

4.1 Attention RNN Language Model

Given a sequence of tokens $\{w_1, w_2, \dots, w_t\}$, an RNN-LM estimates the probability $\Pr[w_t|w_{1:t-1}]$ with a recurrent function

$$h_t = f(h_{t-1}, w_{t-1}) \quad (4)$$

and an output function, usually softmax,

$$\Pr[w_t = v_i | w_{1:t-1}] = \frac{\exp(W_{v_i}^\top h_t + b_{v_i})}{\sum_k \exp(W_{v_k}^\top h_t + b_{v_k})} \quad (5)$$

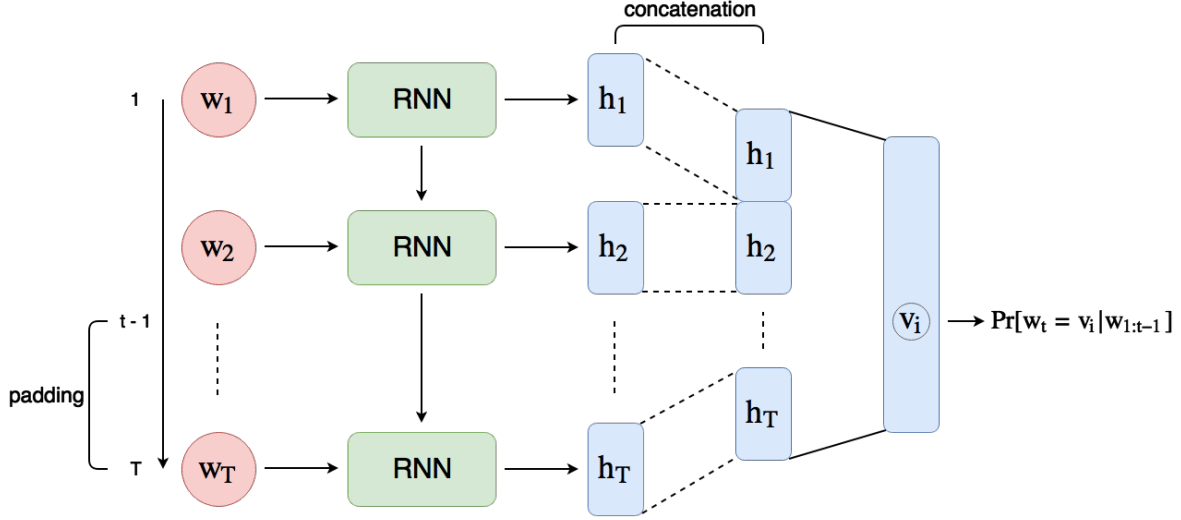


Figure 1. The structure of our attention RNN language model with concatenation

where the implementation of the function f depends on which kind of RNN cell we use, $h_t \in \mathbb{R}^D$, $W \in \mathbb{R}^{D \times V}$ with the column vector W_{v_i} corresponding to a word v_i , and $b \in \mathbb{R}^V$ with the scalar b_{v_i} corresponding to a word v_i (D is the number of units in RNN, and V is the number of unique tokens in all sequences).

We then update the parameters of the RNN-LM by maximizing the log-likelihood on a set of sequences with size N , $\{s_1, s_2, \dots, s_N\}$, and the corresponding tokens, $\{w_1^{s_i}, w_2^{s_i}, \dots, w_{|s_i|}^{s_i}\}$.

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^N \sum_{t=2}^{|s_n|} \log \Pr[w_t^{s_n} | w_{1:t-1}^{s_n}] \quad (6)$$

4.1.1 Attention in RNN-LM

Attention mechanism in sequence-to-sequence model has been proven to be effective in the fields of image caption generation, machine translation, dialogue generation, and etc. Several previous works also indicate that attention is even more impressive on RNN-LM [15].

In attention RNN language model (A-RNN-LM), given the hidden states from time $t - C_{ws}$ to t , denoted as $h_{t-C_{ws}:t}$, where C_{ws} is the attention window size, we want to compute a context vector c_t as a weighted sum of hidden states $h_{t-C_{ws}:t-1}$ and then encode the context vector c_t into the original hidden state h_t .

$$\beta_i = \nu^\top \tanh(W_1 h_t + W_2 h_{t-C_{ws}+i}) \quad (7)$$

$$\alpha_i = \frac{\exp(\beta_i)}{\sum_{k=0}^{C_{ws}-1} \exp(\beta_k)} \quad (8)$$

$$c_t = \sum_{i=0}^{C_{ws}-1} \alpha_i h_{t-C_{ws}+i} \quad (9)$$

$$h'_t = W_3 \begin{bmatrix} h_t \\ c_t \end{bmatrix} \quad (10)$$

where β is Bahdanau's scoring style [2], $W_1, W_2 \in \mathbb{R}^{D \times D}$, and $W_3 \in \mathbb{R}^{D \times 2D}$.

4.1.2 Our Attention RNN-LM with concatenation

In our work, $\{s_1, s_2, \dots, s_N\}$ and $\{w_1^{s_i}, w_2^{s_i}, \dots, w_{|s_i|}^{s_i}\}$ are playlists and songs by adopting Chen *et al.*'s work [4]. More specifically, given a seed song $w_1^{s_i}$ for a playlist s_i , we find top-k approximate nearest neighbors of $w_1^{s_i}$ to formulate a list of songs $\{w_1^{s_i}, w_2^{s_i}, \dots, w_{|s_i|}^{s_i}\}$.

The proposed attention RNN-LM with concatenation (AC-RNN-LM) is shown in Figure 1. We pad $w_{1:t-1}$ to $w_{1:T}$ and concatenate the corresponding $h'_{1:T}$ as the input of our RNN-LM's output function in Eqn (5), where T is the maximum number of songs we consider in one playlist. Therefore, our output function becomes

$$\Pr[w_t = v_i | w_{1:t-1}] = \frac{\exp(W_{v_i}^\top h' + b_{v_i})}{\sum_k \exp(W_{v_k}^\top h' + b_{v_k})} \quad (11)$$

where $W \in \mathbb{R}^{DT \times V}$, $b \in \mathbb{R}^V$ and

$$h' = \begin{bmatrix} h'_1 \\ h'_2 \\ \vdots \\ h'_T \end{bmatrix} \in \mathbb{R}^{DT \times 1} \quad (12)$$

4.2 Policy Gradient

We exploit policy gradient in order to optimize Eqn (1), which is formulated as follows.

4.2.1 Action

An action a is a song id, which is a unique representation of each song, that the model is about to generate. The set of actions in our problem is finite since we would like to recommend limited range of songs.

4.2.2 State

A state s is the songs we have already recommended including the seed song, $\{w_1^{s_i}, w_2^{s_i}, \dots, w_{t-1}^{s_i}\}$.

4.2.3 Policy

A policy $\pi_\theta(s, a)$ takes the form of our AC-RNN-LM and is defined by its parameters θ .

4.2.4 Reward

Reward $\mathcal{R}(s, a)$ is a weighted sum of several reward functions, i.e., $\mathcal{R}_i : s \times a \mapsto \mathbb{R}$. In the following introductions, we formulate 4 important metrics for playlists generation. The policy of our pretrained AC-RNN-LM is denoted as $\pi_{\theta_{RNN}}(s, a)$ with parameters θ_{RNN} , and the policy of our AC-RNN-LM optimized by policy gradient is denoted as $\pi_{\theta_{RL}}(s, a)$ with parameters θ_{RL} .

Diversity represents the variety in a recommended list of songs. Several generated playlists in Chen *et al.*'s work [4] are composed of songs with the same artist or album. It is not regarded as a good playlist for recommendation system because of low diversity. Therefore, we formulate the measurement of the diversity by the euclidean distance between the embeddings of the last song in the existing playlist, $w_{|s|}^s$, and the predicted song, a .

$$\mathcal{R}_1(s, a) = -\log(|d(w_{|s|}^s, a) - C_{\text{distance}}|) \quad (13)$$

where $d(\cdot)$ is the euclidean distance between the embeddings of $w_{|s|}^s$ and a , and C_{distance} is a parameter that represents the euclidean distance that we want the model to learn.

Novelty is also important for a playlist generation system. We would like to recommend something new to users instead of recommend something familiar. Unlike previous works, our model can easily generate playlists with novelty by applying a corresponding reward function. As a result, we model reward of novelty as a weighted sum of normalized playcounts in periods of time [19].

$$\mathcal{R}_2(s, a) = -\log\left(\sum_t w(t) \frac{\log(p_t(a))}{\log(\max_{a' \in A} p_t(a'))}\right) \quad (14)$$

where $w(t)$ is the weight of a time period, t , with a constraint $\sum_t w(t) = 1$, $p_t(a)$ is playcounts of the song a , and A is the set of actions. Note that songs with less playcounts have higher value of \mathcal{R}_2 , and vice versa.

Freshness is a subjective metric for personalized playlist generation. For example, latest songs is usually more interesting for young people, while older people would prefer old-school songs. Here, we arbitrarily choose one direction for optimization to the agent $\pi_{\theta_{RL}}$ to show the feasibility of our approach.

$$\mathcal{R}_3(s, a) = -\log\left(\frac{Y_a - 1900}{2017 - 1900}\right) \quad (15)$$

where Y_a is the release year of the song a .

Coherence is the major feature we should consider to avoid situations that the generated playlists are highly rewarded but lack of cohesive listening experiences. We therefore consider the policy of our pretrained language model, $\pi_{\theta_{RNN}}(s, a)$, which is well-trained on coherent playlists, as a good enough generator of coherent playlists.

$$\mathcal{R}_4(s, a) = \log(\Pr[a|s, \theta_{RNN}]) \quad (16)$$

Combining the above reward functions, our final reward for the action a is

$$\mathcal{R}(s, a) = \gamma_1 \mathcal{R}_1(s, a) + \gamma_2 \mathcal{R}_2(s, a) + \gamma_3 \mathcal{R}_3(s, a) + \gamma_4 \mathcal{R}_4(s, a) \quad (17)$$

where the selection of γ_1 , γ_2 , γ_3 , and γ_4 depends on different applications.

Note that although we only list four reward functions here, the optimization goal \mathcal{R} can be easily extended by a linear combination of more reward functions.

5. EXPERIMENTS AND ANALYSIS

In the following experiments, we first introduce the details of dataset and evaluation metrics in Section 5.1 and training details in Section 5.2. In Section 5.3, we compare pretrained RNN-LMs with different mechanism combination by perplexity to show our proposed AC-RNN-LM is more effectively and efficiently than others. In order to demonstrate the effectiveness of our proposed method, AC-RNN-LM combined with reinforcement learning, we adopt three standard metrics, diversity, novelty, and freshness (cf. Section 5.1) to validate our models in Section 5.4. Moreover, we demonstrate that it is effortless to flexibly manipulate the properties of resulting generated playlists in Section 5.5. Finally, in Section 5.6, we discuss the details about the design of reward functions with given preferred properties.

5.1 Dataset and Evaluation Metrics

The playlist dataset is provided by KKBOX Inc., which is a regional leading music streaming company. It consists of 10,000 playlists, each of which is composed of 30 songs. There are 45,243 unique songs in total.

For validate our proposed approach, we use the metrics as follows.

Perplexity is calculated based on the song probability distributions, which is shown as follows.

$$\text{perplexity} = e^{\frac{1}{N} \sum_{n=1}^N \sum_x -q(x) \log p(x)}$$

where N is the number of training samples, x is a song in our song pool, p is the predicted song probability distribution, and q is the song probability distribution in ground truth.

Diversity is computed as different unigrams of artists scaled by the total length of each playlist, which is measured by Distinct-1 [9]

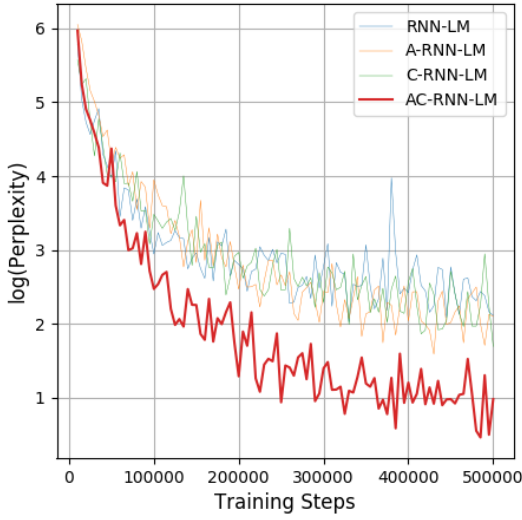


Figure 2. Log-perplexity of different pretrained models on the dataset under different training steps

Novelty is designed for recommending something new to users [19]. The more the novelty is, the lower the metric is.

Freshness is directly measured by the average release year of songs in each playlist.

5.2 Training Details

In the pretraining and reinforcement learning stage, we use 4 layers and 64 units per layer in all RNN-LM with LSTM units, and we choose $T = 30$ for all RNN-LM with padding and concatenation. The optimizer we use is Adam [8]. The learning rates for pretraining stage and reinforcement learning stage are empirically set as 0.001 and 0.0001, respectively.

5.3 Pretrained Model Comparison

In this section, we compare the training error of RNN-LM combining with different mechanisms. The RNN-LM with attention is denoted as A-RNN-LM, the RNN-LM with concatenation is denoted as C-RNN-LM, and the RNN-LM with attention and concatenation is denoted as AC-RNN-LM. Figure 2 reports the training error of different RNN-LMs as log-perplexity which is equal to negative log-likelihood under the training step from 1 to 500,000. Here one training step means that we update our parameters by one batch. As shown in Figure 2, the training error of our proposed model, AC-RNN-LM, can not only decrease faster than the other models but also reach the lowest value at the end of training. Therefore, we adopt AC-RNN-LM as our pretrained model.

Worth noting that the pretrained model is developed for two purposes. One is to provide a good basis for further optimization, and the other is to estimate transition

Table 1. Weights of reward functions for each model

Model	γ_1	γ_2	γ_3	γ_4
RL-DIST	0.5	0.0	0.0	0.5
RL-NOVELTY	0.0	0.5	0.0	0.5
RL-YEAR	0.0	0.0	0.5	0.5
RL-COMBINE	0.2	0.2	0.2	0.4

Table 2. Comparison on different metrics for playlist generation system (The bold text represents the best, and the underline text represents the second)

Model	Diversity	Novelty	Freshness
Embedding [4]	0.32	0.19	2007.97
AC-RNN-LM	0.39	0.20	2008.41
RL-DIST	<u>0.44</u>	0.20	2008.37
RL-NOVELTY	0.42	0.05	2012.89
RL-YEAR	0.40	0.19	2006.23
RL-COMBINE	0.49	<u>0.18</u>	<u>2007.64</u>

probabilities of songs in the reward function. Therefore, we simply select the model with the lowest training error to be optimized by policy gradient and an estimator of $\Pr[a|s, \theta_{RNN}]$ (cf. Eqn (16)).

5.4 Playlist Generation Results

As shown in Table 2, to evaluate our method, we compare 6 models on 3 important features, diversity, novelty, and freshness (cf. Section 5.1), of playlist generation system. The details of models are described as follows. Embedding represents the model of Chen *et al.*'s work [4]. Chen *et al.* construct the song embedding by relationships between user and song and then finds approximate k nearest neighbors for each song. RL-DIST, RL-NOVELTY, RL-YEAR, and RL-COMBINE are models that are pretrained and optimized by the policy gradient method (cf. Eqn (17)) with different weights, respectively, as shown in Table 1.

The experimental results show that for single objective such as diversity, our models can accurately generate playlists with corresponding property. For example, RL-Year can generate a playlist which consists of songs with earliest release years than Embedding and AC-RNN-LM. Besides, even when we impose our model with multiple reward functions, we can still obtain a better resulting playlist in comparison with Embedding and AC-RNN-LM. Sample result is shown in Figure 3.

From Table 2, we demonstrate that by using appropriate reward functions, our approach can generate playlists to fit the corresponding needs easily. We can systematically find more songs from different artists (RL-DIST), more songs heard by fewer people (RL-NOVELTY), or more old songs for some particular groups of users (RL-YEAR).

5.5 Flexible Manipulating Playlist Properties

After showing that our approach can easily fit several needs, we further investigate the influence of γ to the re-



Figure 3. Sample playlists generated by our approach. The left one is generated by Embedding [4] and the right one is generated by RL-COMBINE.

sulting playlists. In this section, several models are trained with the weight γ_2 from 0.0 to 1.0 to show the variances in novelty of the resulting playlists. Here we keep $\gamma_2 + \gamma_4 = 1.0$ and $\gamma_1 = \gamma_3 = 0$ and fix the training steps to 10,000.

As shown in Figure 4, novelty score generally decreases when γ_2 increases from 0.0 to 1.0 but it is also possible that the model may sometimes find the optimal policy earlier than expectation such as the one with $\gamma_2 = 0.625$. Nevertheless, in general, our approach can not only let the model generate more novel songs but also make the extent of novelty be controllable. Besides automation, this kind of flexibility is also important in applications.

Take online music streaming service as an example, when the service provider wants to recommend playlists to a user who usually listens to non-mainstream but familiar songs (i.e., novelty score is 0.4), it is more suitable to generate playlists which consists of songs with novelty scores equals to 0.4 instead of generating playlists which is composed of 60% songs with novelty scores equals to 0.0 and 40% songs with novelty scores equals to 1.0. Since users usually have different kinds of preferences on each property, to automatically generate playlists fitting needs of each user, such as novelty, becomes indispensable. The experimental results verify that our proposed approach can satisfy the above application.

5.6 Limitation on Reward Function Design

When we try to define a reward function \mathcal{R}_i for a property, we should carefully avoid the bias from the state s . In other words, reward functions should be specific to the corresponding feature we want. One common issue is that

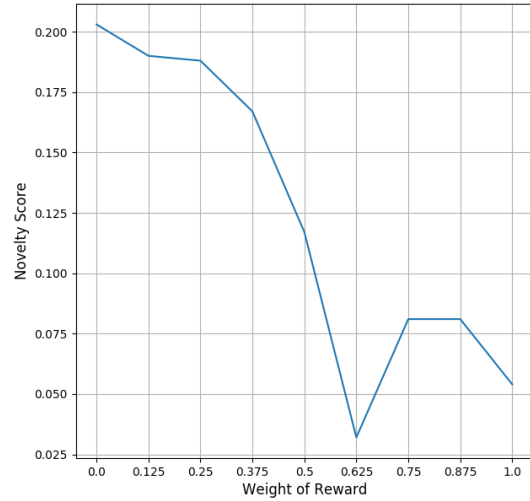


Figure 4. Novelty score of playlists generated by different imposing weights

the reward function may be influenced by the number of songs in state s . For example, in our experiments, we adopt Distinct-1 as the metric for diversity. However, we cannot also adopt Distinct-1 as our reward function directly because it is scaled by the length of playlists which results in all actions from states with fewer songs will be benefited. Therefore, difference between cR_1 and Distinct-1 is the reason that RL-DIST does not achieve the best performance in Distinct-1 (cf. Table 1). In summary, we should be careful to design reward functions, and sometimes we may need to formulate another approximation objective function to avoid biases.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we develop a playlist generation system which is able to generate personalized playlists automatically and flexibly. We first formulate playlist generation as a language modeling problem. Then by exploiting the techniques of RNN-LM and reinforcement learning, the proposed approach can flexibly generate suitable playlists for different preferences of users.

In our future work, we will further investigate the possibility to automatically generate playlists by considering qualitative feedback. For online music streaming service providers, professional music curators will give qualitative feedback on generated playlists so that research developers can improve the quality of playlist generation system. Qualitative feedback such as ‘songs from diverse artists but similar genres’ is easier to be quantitative. We can design suitable reward functions and generate corresponding playlists by our approach. However, other feedback such as ‘falling in love playlist’ is more difficult to be quantitative. Therefore, we will further adopt audio features and explicit tags of songs in order to provide a better playlist generation system.

7. REFERENCES

- [1] Masoud Alghoniemy and Ahmed H. Tewfik. A network flow model for playlist generation. In *Proceedings of International Conference on Multimedia and Expo*, pages 329–332, 2001.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.
- [3] Rachel M. Bittner, Minwei Gu, Gandalf Hernandez, Eric J. Humphrey, Tristan Jehan, P. Hunter McCurry, and Nicola Montecchio. Automatic playlist sequencing and transitions. In *Proceedings of the 18th International Conference on Music Information Retrieval*, pages 472–478, 2017.
- [4] Chih-Ming Chen, Ming-Feng Tsai, Yu-Ching Lin, and Yi-Hsuan Yang. Query-based music recommendations via preference embedding. In *Proceedings of the ACM Conference Series on Recommender Systems*, pages 79–82, 2016.
- [5] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. Playlist generation using start and end songs. In *Proceedings of the 9th International Society for Music Information Retrieval Conference*, pages 173–178, 2008.
- [6] Negar Hariri, Bamshad Mobasher, and Robin Burke. Context-aware music recommendation based on latent-topic sequential patterns. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 131–138, New York, NY, USA, 2012. ACM.
- [7] Dietmar Jannach, Lukas Lerche, and Iman Kamehkhosh. Beyond "hitting the hits": Generating coherent music playlist continuations with the right tracks. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 187–194, New York, NY, USA, 2015. ACM.
- [8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [9] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. *CoRR*, abs/1510.03055, 2015.
- [10] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *CoRR*, abs/1606.01541, 2016.
- [11] Elad Liebman and Peter Stone. DJ-MC: A reinforcement-learning agent for music playlist recommendation. *CoRR*, abs/1401.1880, 2014.
- [12] Beth Logan. Content-based playlist generation: Exploratory experiments. 2002.
- [13] François Maillet, Douglas Eck, Guillaume Desjardins, and Paul Lamere. Steerable playlist generation by learning song similarity from radio station playlists. In *Proceedings of the 10th International Society for Music Information Retrieval Conference*, 2009.
- [14] Brian McFee and Gert Lanckriet. The natural language of playlists. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 537–542, 2011.
- [15] Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. Coherent dialogue with attention-based language models. In *The Thirty-First AAAI Conference on Artificial Intelligence*, 2016.
- [16] Elias Pampalk, Tim Pohle, and Gerhard Widmer. Dynamic playlist generation based on skipping behavior. In *Proceedings of the 6th International Society for Music Information Retrieval Conference*, pages 634–637, 2005.
- [17] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharmashan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. 2017.
- [18] Saúl Vargas. New approaches to diversity and novelty in recommender systems. In *Proceedings of the Fourth BCS-IRSG Conference on Future Directions in Information Access*, FDIA'11, pages 8–13, Swindon, UK, 2011. BCS Learning & Development Ltd.
- [19] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. Auralist: Introducing serendipity into music recommendation. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 13–22, New York, NY, USA, 2012. ACM.
- [20] Xinxi Wang Zhe Xing and Ye Wang. Enhancing collaborative filtering music recommendation by balancing exploration and exploitation. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pages 445–450, 2014.

BRIDGING AUDIO ANALYSIS, PERCEPTION AND SYNTHESIS WITH PERCEPTUALLY-REGULARIZED VARIATIONAL TIMBRE SPACES

Philippe Esling, Axel Chemla-Romeu-Santos, Adrien Bitton

Institut de Recherche et Coordination Acoustique-Musique (IRCAM)

CNRS - UMR 9912, UPMC - Sorbonne Universite

1 Place Igor Stravinsky, F-75004 Paris, France

{esling, chemla, bitton}@ircam.fr

ABSTRACT

Generative models aim to understand the properties of data, through the construction of *latent spaces* that allow classification and generation. However, as the learning is unsupervised, the latent dimensions are not related to perceptual properties. In parallel, music perception research has aimed to understand timbre based on human dissimilarity ratings. These lead to timbre spaces which exhibit perceptual similarities between sounds. However, they do not generalize to novel examples and do not provide an invertible mapping, preventing audio synthesis. Here, we show that Variational Auto-Encoders (VAE) can bridge these lines of research and alleviate their weaknesses by regularizing the latent spaces to match perceptual distances collected from timbre studies. Hence, we propose three types of regularization and show that they lead to spaces that are simultaneously coherent with signal properties and perceptual similarities. We show that these spaces can be used for efficient audio classification. We study how audio descriptors are organized along the latent dimensions and show that even though descriptors behave in a non-linear way across the space, they still exhibit a locally smooth evolution. We also show that, as this space generalizes to novel samples, it can be used to predict perceptual similarities of novel instruments. Finally, we exhibit the generative capabilities of our spaces, that can directly synthesize sounds with continuous evolution of timbre perception.

1. INTRODUCTION

Generative models aim to understand the underlying distribution of data based on the observation of examples, in order to generate novel content. Recently, audio synthesis using these models has seen great improvements through efficient waveform models, such as *WaveNet* [19] and *SampleRNN* [17]. These models are able to generate high-quality audio matching the properties of the corpus

they have been trained on. However, these models give little control over the output or the hidden features it results from. More recently, *NSynth* [4] has been proposed to generate instrumental notes, while allowing to morph between specific instruments. However, these models remain highly complex, requiring very large number of parameters, long training times and a large number of examples.

Amongst recent generative models, a key proposal is the *Variational Auto-Encoder* (VAE) [11]. In these models, encoder and decoder networks are jointly trained through the construction of a *latent space*, that allow both analysis and generation. VAEs address all the limitations of control and analysis through this latent space, while remaining simple and fast to learn without requiring large sets of examples. Furthermore, the VAE seems able to disentangle underlying variation factors by learning independent latent variables [7]. However, these unsupervised dimensions are not related to perceptual properties, which might hamper the control and use of these spaces for analysis and synthesis. The potential of VAEs for audio applications has only been scarcely investigated and mostly for speech source separation [13] and transformation [8]. However, the use of variational latent spaces specifically for musical audio synthesis is yet to be investigated.

In parallel, music perception research has tried to understand the mechanisms behind the perception of instrumental *timbre*. Several studies [15] collected dissimilarity ratings between pairs of instrumental samples. Then, Multi-Dimensional Scaling (MDS) is applied to these ratings to obtain *timbre spaces*, which exhibit the perceptual similarities between instruments. Although these spaces provide interesting analyses, they are inherently limited by the fact that MDS produces a fixed discrete space, which has to be recomputed for any new sample. Therefore, these spaces do not generalize to novel examples and do not provide an invertible mapping, preventing audio synthesis.

Here, we show that we can bridge analysis, synthesis and perceptual audio research by regularizing the learning of latent spaces so that they match the perceptual distances from timbre studies. Our overall approach is depicted in Figure 1. First, we adapt the VAE to analyze musical audio content, by relying on the Non-Stationary Gabor Transform (NSGT) with a Constant-Q scale. This transform allows us to obtain a log-frequency scale while remaining invertible, which is critical to perform audio synthesis. Even



© Philippe Esling, Axel Chemla, Adrien Bitton. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Philippe Esling, Axel Chemla, Adrien Bitton. "Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

with a simple model on a small training set, we show that this provides a generative model with an interesting latent space, able to synthesize novel instrumental sounds.

Then, we propose three regularizations to the learning objective, aiming to enforce that the latent space exhibits the same topology as the topology of timbre spaces. We build a model of perceptual relationships by normalizing dissimilarity ratings from five timbre space studies [5, 9, 12, 14, 16]. We show that perceptually-regularized latent spaces are both coherent with perceptual dissimilarities, while being able to reconstruct audio samples with a high accuracy. Hence, we can drive the learning of the latent space to match the topology of any given target space.

We demonstrate that these spaces can be used for audio classification by training low-capacity classifiers on the spaces. We obtain high accuracy for *family* and *instrument* labels, but also for the *pitch* and *dynamics*, even though the model had no information on these during training. We exhibit the generative capabilities of our spaces, by assessing the reconstruction quality of the model on a test dataset. We show that the latent spaces can be directly used to synthesize sounds with continuous evolution of timbre perception. We also show that these spaces generalize to novel samples, by encoding instruments that were not part of the training set. Therefore, the spaces could be used to predict the perceptual similarities of novel instruments. Finally, we study how audio descriptors behave along the latent dimensions, by generating audio samples on a grid across space. We show that even though descriptors behave in a non-linear way across the space, they still follow a locally smooth evolution. Our source code, audio examples and additional figures and animations are available online¹.

2. STATE-OF-ART

2.1 Variational auto-encoders

Generative models are a flourishing class of machine learning approaches, aiming to find the underlying probability distribution of the data $p(\mathbf{x})$ [2]. Formally, based on a set of examples in $\mathbf{x} \in \mathbb{R}^{d_x}$, we assume that these follow an unknown probability distribution $p(\mathbf{x})$. Furthermore, we consider a set of *latent variables* defined in a lower-dimensional space $\mathbf{z} \in \mathbb{R}^{d_z}$ ($d_z \ll d_x$), a higher-level representation that could have led to generate a given example. These latent variables help govern the distribution of the data and enhance the *expressivity* of the model. The complete model is defined by the joint probability distribution $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x} | \mathbf{z})p(\mathbf{z})$. We could find $p(\mathbf{x})$ by marginalizing \mathbf{z} from the joint probability. However, for most models, this integral can not be found in closed form.

Recently, *variational inference* (VI) has been proposed to solve this problem through *optimization*. VI assumes that if the distribution is too complex to find, we could find a simpler approximate distribution that still models the data, while trying to minimize its difference to the real distribution. VI specifies a family \mathcal{Q} of approximate densities,

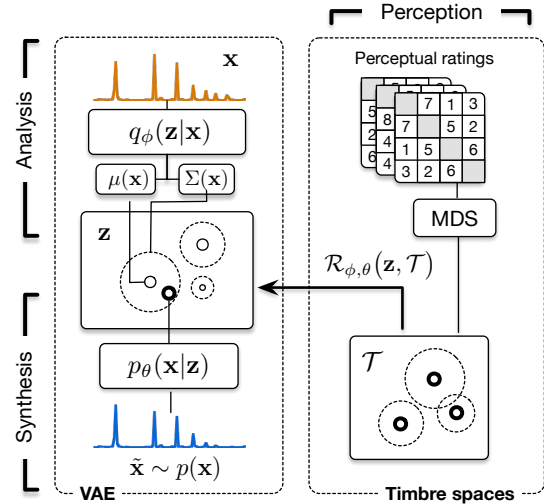


Figure 1. The VAE models audio samples \mathbf{x} by learning an encoder $q_\phi(\mathbf{z} | \mathbf{x})$ which maps them to a Gaussian $\mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))$ in latent space \mathbf{z} . The decoder $p_\theta(\mathbf{x} | \mathbf{z})$ samples from this Gaussian to generate a reconstruction $\tilde{\mathbf{x}}$. Perception studies use dissimilarity ratings to construct a *timbre space* that exhibits the perceptual distances between instruments. Here, we develop regularizations methods $\mathcal{R}(\mathbf{z}, \mathcal{T})$, to enforce that the variational model finds a topology of latent space \mathbf{z} that matches the topology of the timbre space \mathcal{T} .

where each member $q(\mathbf{z} | \mathbf{x}) \in \mathcal{Q}$ is a candidate approximation to the exact conditional $p(\mathbf{z} | \mathbf{x})$. Hence, the inference can be transformed into an optimization problem by minimizing the Kullback-Leibler (KL) divergence between the approximation and the original density

$$q^*(\mathbf{z} | \mathbf{x}) = \arg \min_{q(\mathbf{z} | \mathbf{x}) \in \mathcal{Q}} \mathcal{D}_{KL}[q(\mathbf{z} | \mathbf{x}) \| p(\mathbf{z} | \mathbf{x})] \quad (1)$$

By developing this KL divergence and re-arranging terms (the detailed development can be found in [11]), we obtain

$$\log p(\mathbf{x}) - \mathbb{E}_{\mathbf{z}} [\log p(\mathbf{x} | \mathbf{z})] - \mathcal{D}_{KL}[q(\mathbf{z} | \mathbf{x}) \| p(\mathbf{z})] = \quad (2)$$

This formulation describes the quantity we want to maximize $\log p(\mathbf{x})$ minus the error we make by using an approximate q instead of p . Therefore, we can optimize this alternative objective, called the *evidence lower bound* (ELBO). Now, to optimize this objective, we will rely on parametric distributions $q_\phi(\mathbf{z})$ and $p_\theta(\mathbf{z})$. Optimizing our generative model will amount to optimizing these parameters $\{\theta, \phi\}$ of these distributions with

$$\mathcal{L}_{\theta, \phi} = \mathbb{E} [\log p_\theta(\mathbf{x} | \mathbf{z})] - \beta \cdot \mathcal{D}_{KL}[q_\phi(\mathbf{z} | \mathbf{x}) \| p_\theta(\mathbf{z})] \quad (3)$$

We can see that this equation involves $q_\phi(\mathbf{z} | \mathbf{x})$ which *encodes* the data \mathbf{x} into the latent representation \mathbf{z} and a *decoder* $p_\theta(\mathbf{x} | \mathbf{z})$, which allows generating a data \mathbf{x} given a latent configuration \mathbf{z} . Hence, this structure defines the *Variational Auto-Encoder* (VAE), depicted in Figure 1 (Left).

¹ <https://github.com/acids-ircam/ismir2018>

The VAE objective can be interpreted intuitively. The first term increases the likelihood of the data generated given a configuration of the latent, which amounts to minimize the *reconstruction error*. The second term represents the error made by using a simpler distribution $q_\phi(\mathbf{z} | \mathbf{x})$ rather than the true distribution $p_\theta(\mathbf{z})$. Therefore, this *regularizes* the choice of approximation q so that it remains close to the true posterior distribution [11]. Here, we also introduced a weight β on the KL divergence, which has been shown to improve the capacity of the model to disentangle factors of variations in the data [7].

VAEs are powerful representation learning frameworks, while remaining simple and fast to learn without requiring large sets of examples [18]. Their potential for audio applications have been only scarcely investigated yet and mostly in topics related to speech processing such as blind source separation [13] and speech transformation [8]. However, to our best knowledge, the use of VAE to perform musical audio analysis and generation has yet to be investigated.

2.2 Timbre spaces and auditory perception

For decades, researchers have tried to understand the mechanisms of *timbre* perception. Timbre is the set of properties that distinguishes two instruments playing the same note at the same intensity. Several studies tried to understand this phenomenon by relying on *timbre spaces* [6], a model that aims to organize audio samples based on human dissimilarity ratings. The experimental protocol consists of presenting pairs of sounds to subjects. Each subject has to rate the perceptual dissimilarity of all pairs of samples inside a selected set of instruments. Then, these ratings are compiled into a set of dissimilarity matrices that are analyzed with Multi-Dimensional Scaling (MDS). The MDS algorithm provides a timbre space that exhibits the perceptual distances between different instruments. This process is depicted in Figure 1 (Right). Here, we briefly detail the studies and redirect the interested readers to the full articles for more details.

In his seminal paper, Grey [5] performed a study with 16 instrumental sound samples in which 22 subjects had to rate their dissimilarities on a continuous scale from 0 (most similar) to 1 (most dissimilar), leading to the first construction of a timbre space. Following this study, Krumhansl [12] used 21 instruments with 9 subjects on a discrete scale from 1 to 9, Iverson et al. [9] with 16 samples and 10 subjects on a continuous scale from 0 to 1, McAdams et al. [16] with 18 instruments and 24 subjects on a discrete scale from 1 to 16 and, finally, Lakatos [14] with 17 subjects and different instrument sets on a continuous scale from 0 to 1. Each of these studies shed light on different aspects of audio perception, depending on the interpretation of the dimensions. However, all studies produced different spaces with different dimensions, preventing a generalization on the acoustic cues that might correspond to timbre dimensions. Furthermore, these studies are inherently limited by the fact that ordination techniques (e.g. MDS) produce fixed spaces that must be recomputed for any new data point [16]. Hence, these spaces are unable

to generalize nor can we generate data from these as they do not provide an invertible mapping. Here, we show that learning latent spaces, while regularizing their topology to fit perceptual ratings can alleviate these limitations.

3. REGULARIZING THE TOPOLOGY OF LATENT SPACES

We show that we can influence the learning of the latent space \mathbf{z} so that it follows the topology of a given target space \mathcal{T} . Here, we rely on timbre spaces based on perceptual ratings as a target space. However, it should be noted that this idea can be applied to any target space. Here, we consider a set of audio samples x_i where each have relations in both latent space \mathbf{z}_i and target space \mathcal{T}_i . In order to relate the elements of the audio set to the perceptual space, we consider that each sample is labeled with its instrumental class \mathcal{C}_i , that has an equivalent in the timbre space.

3.1 Penalty regularization

First, we define an additive *penalty* regularization $\mathcal{R}(\mathbf{z}, \mathcal{T})$ that imposes that the properties of the latent \mathbf{z} are similar to that of the target \mathcal{T} . Our objective becomes

$$\mathbb{E}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta D_{KL}[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})] + \alpha \mathcal{R}(\mathbf{z}, \mathcal{T})$$

Hence, amongst two otherwise equal solutions, the model is pushed to select the one that comply with the penalty. The weight α allows us to control the influence of this regularization. In our case, we want the distances between instruments to follow the perceptual distances. Therefore, we need to minimize the differences between the distances in latent space $\mathcal{D}_{i,j}^{\mathbf{z}} = \mathcal{D}(\mathbf{z}_i, \mathbf{z}_j)$ and in target space $\mathcal{D}_{i,j}^{\mathcal{T}} = \mathcal{D}(\mathcal{T}_i, \mathcal{T}_j)$. The regularization criterion will minimize the differences between these sets of distances

$$\mathcal{R}(\mathbf{z}, \mathcal{T}) = \sum_{i \neq j} \mathcal{R}_{i,j}(\mathbf{z}, \mathcal{T}) = \sum_{i \neq j} \mathcal{R}(\mathcal{D}_{i,j}^{\mathbf{z}}, \mathcal{D}_{i,j}^{\mathcal{T}}) \quad (4)$$

Euclidean. First, we rely on the Euclidean distance to compute the distance between points in both spaces with $\mathcal{D}_{i,j}^{\mathcal{S}} = \|\mathcal{S}_i - \mathcal{S}_j\|^2$ and also to compare distance matrices

$$\mathcal{R}_{i,j}(\mathbf{z}, \mathcal{T}) = \|\mathcal{D}_{i,j}^{\mathbf{z}} - \mathcal{D}_{i,j}^{\mathcal{T}}\|^2 \quad (5)$$

This regularization provides an incentive to the model to obtain the Euclidean metric properties of the target space.

Gaussian. Here, we model the fact that perceptual ratings are subjective assessments. Therefore, we consider that each perceptual rating between instruments i and j is drawn from a univariate Gaussian $d_{i,j} \sim \mathcal{N}(\mu_{i,j}, \sigma_{i,j})$. As we can see, we define a different distribution for each *pair* of instruments. When evaluating the regularization, we draw a different distance at each iteration for all pairs

$$\{\mathcal{D}_{i,j}^{\mathcal{T}}\}_{it} \sim \mathcal{N}(\mu_{i,j}, \sigma_{i,j})$$

Hence, this regularization models the uncertainty present in the set of perceptual ratings.

3.2 Prior regularization

In the VAE objective, we observe that the prior $p(\mathbf{z})$ already carries information on the organization of the latent space. Therefore, we can inject the desired topology of the latent space inside that term. Here, we propose to introduce a class-based prior

$$p(\mathbf{z}_i) = \mathcal{N}(\mu_{\mathcal{T}}(C_i), \sigma_{\mathcal{T}}(C_i))$$

where C_i is the class of element i . Therefore, this prior pushes the VAE to find a configuration of the samples in latent space so that they follow the same distribution as their class in our target timbre space. The computation of class means $\mu_{\mathcal{T}}(C_i)$ and covariances $\sigma_{\mathcal{T}}(C_i)$ based on the perceptual ratings is detailed in Section 4.1.

4. EXPERIMENTS

4.1 Datasets

Timbre studies. We rely on perceptual dissimilarity ratings collected across five independent timbre studies [5, 9, 12, 14, 16], detailed globally in [3, 15]. As discussed earlier (Section 2.2), even though all studies follow the same protocol, there are some discrepancies in the instruments, number of participants and rating scales.

Hence, we normalize the dissimilarity ratings so that all studies map to a common scale from 0 to 1. Then, we compute the maximal set of instruments for which we had pairwise ratings for all pairs by counting co-occurrences in studies. This leads to a set of 11 instruments (Piano, Cello, Violin, Flute, Clarinet, Trombone, Horn, Oboe, Saxophone, Trumpet, Tuba). Finally, we extract the set of ratings that corresponds to our selected instruments, amounting to a total of 11845 pairwise ratings. Based on this set of ratings, we compute an MDS space to obtain the positions in target space of each instrument (which also corresponds to the mean $\mu_{\mathcal{T}}$) and to ensure the consistency of our normalized perceptual space. For all pairs of instruments, we also fit a Gaussian distribution to the pairwise dissimilarity ratings in order to obtain the mean $\mu_{i,j}$ and variance $\sigma_{i,j}$ of that pair for the Gaussian regularization. We derive the global variance $\sigma_{\mathcal{T}}$ for each instrument, by taking the mean of their pairwise variances. Results of this analysis are displayed in Figure 2. Even though ratings come from different studies, the resulting space appears very coherent with clusters of families and the distances between individual instruments correlated to previous perceptual studies.

Audio datasets. In order to learn the distribution of instrumental audio, we rely on the Studio On Line (SOL) database [1]. We selected 2,200 samples to represent the 11 instruments for which we extracted perceptual ratings. These represent the whole tessitura and dynamics available (to remove effects from the pitch and loudness). All recordings were resampled to a sampling rate of 22050Hz. For each audio sample, we compute the Non-Stationary Gabor Transform (NSGT) mapped on a Constant-Q scale of 24 bins per octave. We only keep the magnitude of the NSGT to train our models. Then, we perform a corpus-wide normalization to preserve the relative intensities of

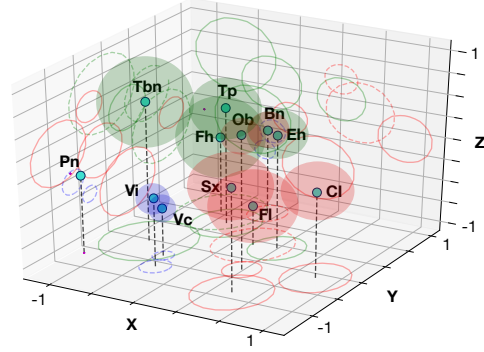


Figure 2. Multi-dimensional scaling (MDS) applied to the combined normalized set of perceptual dissimilarity ratings (strings in blue, brasses in green and winds in red).

the samples and extract a single temporal frame to represent the given audio sample. Finally, the dataset is randomly split between a training (90%) and test (10%) set.

4.2 Models

In order to evaluate our proposal, we rely on a very simple VAE architecture to show its efficiency. The encoder is defined as a 3-layers feed-forward network with ReLU activations and 3000 units per layer. The last layer maps to a latent space of 64 dimensions. The decoder is defined with the same architecture, mapping back to the dimensionality of the input. For learning the model, we use a value of β , which is linearly increased from 0 to 2 during the first 100 epochs (*warmup* procedure [18]). In order to train the model, we rely on the ADAM optimizer [10] with an initial learning rate of 0.00001, and a Xavier weight initialization [18]. In a first stage, we train the model without perceptual regularization ($\alpha = 0$) for 5000 epochs. Then, we introduce the perceptual regularization ($\alpha = 1$) and train for another 1000 epochs. This allows the model to first focus on the quality of the reconstruction with its own unsupervised regularization, and then to converge towards a solution with perceptual space properties. This leads to a training time of one hour on a NVIDIA Titan X GPU.

5. RESULTS

5.1 Latent spaces properties

In order to visualize the latent spaces, we apply a Principal Component Analysis (PCA) to obtain a 3d representation. Using a PCA ensures that the representation is a linear transform that preserves the distances inside the original space. This also provides an exploitable control space for audio synthesis. Results are displayed in Figure 3.

As we can see, the VAE without regularization is already able to dissociate instrumental distributions, while providing almost perfect reconstruction of audio samples from the low-dimensional space. This confirms that VAEs can provide interesting latent spaces for analysis and synthesis. However, the relationships between instruments are

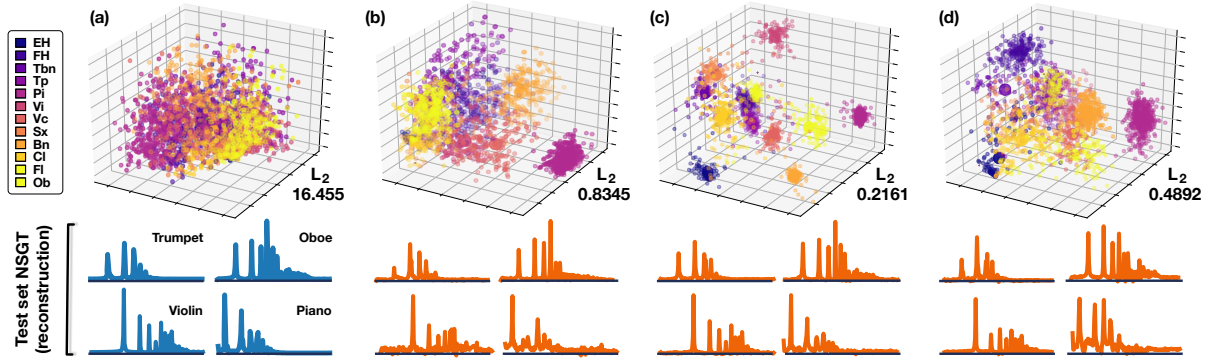


Figure 3. Comparing the latent spaces for the VAE unregularized (a) or with Prior (b), Euclidean (c) and Gaussian (d) regularization. The mean L_2 differences between latent and timbre spaces distances is indicated on each graph. We show under each space the reconstruction of NSGT distributions from the test set directly from these spaces.

entirely different from perceptual ratings. Furthermore, the large variance of the distributions seem to indicate that the model rather tries to spread the information across the latent space to help the reconstruction.

In the case of all regularizations (b-d), we can clearly see the enhancement on the dissociation of instrumental distributions. Furthermore, the overall distances between instruments match well the distances based on perceptual ratings (Figure 2). This similarity is particularly striking for the L_2 regularization (c), which provides the lowest overall differences to our combined timbre space. This might come from the fact that MDS spaces have an Euclidean metric topology. However, this might also indicate an effect of *over-regularization*, which might impact generalization. For all regularized latent spaces, the instrumental distributions are shuffled around the space in order to comply with the reconstruction objective. However, the pairwise distances reflecting perceptual relations are well matched as indicated by their respective L_2 differences to the timbre space. Finally, by looking at the reconstructions of the NSGT distributions from the test set, we can see that enforcing the perceptual topology to the latent spaces does not impact the quality of audio reconstruction (this evaluation is quantified in Section 5.3). However, we note an occasional addition of low-amplitude noise, which might indicate that the model focuses on optimizing the partials rather than the low-amplitude tail of the distribution.

5.2 Discriminative capabilities

We evaluate the discriminative capabilities of the latent spaces through a classification task. We use a very low-capacity classifier composed of a single-layer network of 512 ReLU units with batch normalization and softmax regression. The low-capacity classifier ensures that the latent space needs to be well organized to obtain a good accuracy. In order to evaluate the impact of our proposal, we also compare these results to a simple PCA with softmax regression and an Auto-Encoder (AE) with the same capacity as the VAE. Results are presented in Table 1.

We can see that all models perform an excellent classifi-

Method	Family	Instrument	Pitch	Dynam.
PCA	0.790	0.697	0.167	0.527
AE	0.973	0.957	0.936	0.597
VAE	0.978	0.993	0.963	0.941
Prior	0.975	0.991	0.993	0.936
Euclidean	0.972	0.990	0.990	0.943
Gaussian	0.982	0.991	0.989	0.948

Table 1. Discriminative capabilities in classifying *family*, *instrument*, *pitch* and *dynamics* of the test set.

Method	$\log p(\mathbf{x})$	$\ \mathbf{x} - \tilde{\mathbf{x}}\ ^2$
PCA	-	2.2570
AE	-1.2008	1.6223
VAE	-2.3443	0.1593
Prior	-2.7143	0.1883
Euclidean	17.8960	0.1223
Gaussian	0.2894	0.1749

Table 2. Generative capabilities evaluated on the log likelihood and reconstruction error over the test set.

cation of instrumental properties. However, a very interesting observation comes from the vanilla VAE providing the best accuracy on *instrument* classification, even though we regularized other models with distances highly relevant to these categories. This might underline the fact that perceptual information could blur discrimination of highly similar instruments (such as violin and violoncello). Interestingly, the symmetric results on *pitch* and *dynamics* categories might indicate that regularized model are pushed to focus on timbre properties. Therefore, they need to more clearly separate the variations coming from pitch and loudness to understand the variability of timbre.

5.3 Generative capabilities

We quantify generative capabilities by evaluating reconstructions from the latent space, through the log likelihood and mean difference between original and reconstructed audio on the test set. The results are presented in Table 2.

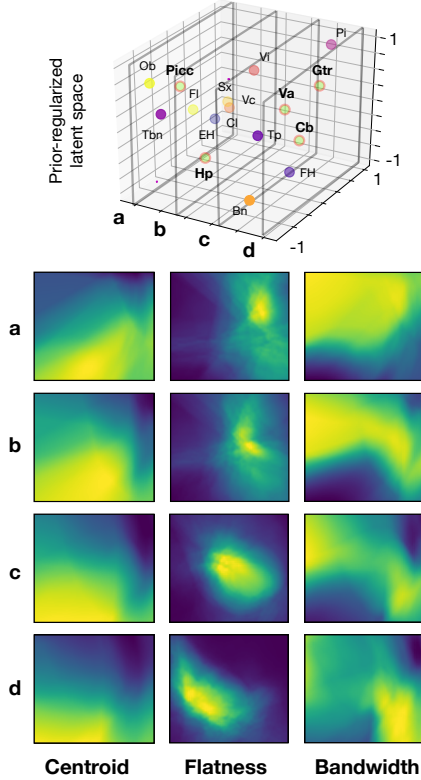


Figure 4. (Top) We encode instruments that were not part of timbre studies to show the *out-of-domain* capabilities of latent spaces. (Bottom) *Topology of descriptors*. We define 4 projection planes equally spaced across the x axis. We sample points at these positions on a 50x50 grid and reconstruct their audio distribution to compute their spectral *centroid*, *flatness* and *bandwidth*.

Overall, the regularizations do not impact the reconstruction quality of the model. Furthermore, we can now sample directly from the spaces to obtain novel sounds that remain perceptually relevant, which allows us to turn our spaces into generative timbre synthesizers. However, as previously hypothesized, the L_2 regularization seems to have a too strong effect on the latent space, disrupting the generalization of the model. We provide generated audio clips representing paths between different instruments in the latent space on the supporting repository for subjective evaluation of the latent space audio synthesis.

5.4 Perceptual inference

The encoder of our perceptually-regularized spaces is able to analyze new instruments that were not part of the original timbre studies. Hence, we could hope that it is able to predict perceptual relationships between new instruments, to feed further timbre studies. To evaluate this, we extracted instruments outside of our perceptual set (Contrabass, Guitar, Harp, Piccolo, Viola) and encode these samples in the latent space to study the *out-of-domain* generalization capabilities of our model. Results are presented in

Figure 4 (Top, only the centroid of distributions are shown for clarity). Here, the Piccolo and Viola seem to group in a coherent way with their families. However, the Guitar and Harp do not provide such straightforward relationships. Therefore, perceptual inference from these spaces would require more extensive perception experiments.

5.5 The topology of audio descriptors

We analyze the behavior of signal descriptors across the latent space in order to study their topology. As the space is continuous, we do so by sampling uniformly the PCA space and then using the decoder to generate all audio samples on this grid. Then, we compute the audio descriptors of these samples. In order to provide a visualization here, we select equally-distant planes across the x dimension (at positions $\{-.75, -.25, .25, .75\}$) in Figure 4 for the spectral *flatness*, *centroid* and *bandwidth*. Videos of continuous traversals of the latent space for different descriptors are available on the supporting repository.

Audio descriptors seem to be organized in a non-linear way across our spaces. However, they still exhibit both locally smooth evolution and an overall logical organization. This shows that our model is able to organize audio variations. A very interesting observation comes from the topology of the centroid. Indeed, all perceptual studies underline its linear correlation to timbre perception, which is partly confirmed by our model (see Figure 4). This confirms the perceptual relevance of these latent spaces. However, this also shows that the relation between centroid and timbre perception might not be entirely linear.

6. CONCLUSION

We have shown that VAEs can learn a latent space allowing for high-level audio analysis and synthesis directly from these spaces. We proposed different methods for regularizing these spaces to follow the metric properties of timbre spaces. These regularized models provide a control space from which the generation of perceptually relevant audio content is straightforward. By analyzing the behavior of audio descriptors across the latent space, we have shown that, while following a non-linear evolution, they still exhibit some locally smooth properties. Future works on these spaces include perceptual experiments to confirm their perceptual topology and also to thrive on the smoothness of audio descriptors to develop a descriptor-based synthesizer.

7. ACKNOWLEDGEMENTS

This work was supported by the MAKIMOno project 17-CE38-0015-01 funded by the French ANR and the Canadian Natural Sciences and Engineering Research Council (STPG 507004-17) and the ACTOR Partnership funded by the Canadian Social Sciences and Humanities Research Council (895-2018-1023).

8. REFERENCES

- [1] Guillaume Ballet, Riccardo Borghesi, Peter Hoffmann, and Fabien Lévy. Studio online 3.0: An internet “killer application” for remote access to ircam sounds and processing tools. *Journée d’Informatique Musicale (JIM)*, 1999.
- [2] Christopher M. Bishop and Tom M. Mitchell. Pattern recognition and machine learning. 2014.
- [3] John A. Burgoyne and Stephen McAdams. A meta-analysis of timbre perception using nonlinear extensions to clascal. In *International Symposium on Computer Music Modeling and Retrieval*, pages 181–202. Springer, 2007.
- [4] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders. *arXiv preprint arXiv:1704.01279*, 2017.
- [5] John M Grey. Multidimensional perceptual scaling of musical timbres. *the Journal of the Acoustical Society of America*, 61(5):1270–1277, 1977.
- [6] John M. Grey and John W. Gordon. Perceptual effects of spectral modifications on musical timbres. *The Journal of the Acoustical Society of America*, 63(5):1493–1500, 1978.
- [7] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.
- [8] Wei-Ning Hsu, Yu Zhang, and James Glass. Learning latent representations for speech generation and transformation. *arXiv preprint arXiv:1704.04222*, 2017.
- [9] Paul Iverson and Carol L. Krumhansl. Isolating the dynamic attributes of musical timbre. *The Journal of the Acoustical Society of America*, 94(5):2595–2603, 1993.
- [10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [11] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [12] Carol L. Krumhansl. Why is musical timbre so hard to understand. *Structure and perception of electroacoustic sound and music*, 9:43–53, 1989.
- [13] Jen-Tzung Kuo and Kuan-Ting Chien. Variational recurrent neural networks for speech separation. *INTER-SPEECH 2017*.
- [14] Stephen Lakatos. A common perceptual space for harmonic and percussive timbres. *Perception & psychophysics*, 62(7):1426–1439, 2000.
- [15] Stephen McAdams, Bruno L. Giordano, Patrick Susini, Geoffroy Peeters, and Vincent Rioux. A meta-analysis of acoustic correlates of timbre dimensions. *Journal of the Acoustical Society of America*, 120(5):3275, 2006.
- [16] Stephen McAdams, Suzanne Winsberg, Sophie Donnadieu, Geert De Soete, and Jochen Krimphoff. Perceptual scaling of synthesized musical timbres: Common dimensions, specificities, and latent subject classes. *Psychological research*, 58(3):177–192, 1995.
- [17] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.
- [18] Casper K. Sønderby, Tapani Raiko, Lars Maaløe, Søren K. Sønderby, and Ole Winther. How to train deep variational autoencoders and probabilistic ladder networks. *arXiv preprint arXiv:1602.02282*, 2016.
- [19] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

CONDITIONING DEEP GENERATIVE RAW AUDIO MODELS FOR STRUCTURED AUTOMATIC MUSIC

Rachel Manzelli*

Vijay Thakkar*

Ali Siahkamari

Brian Kulis

*Equal contributions

ECE Department, Boston University

{manzelli, thakkarv, siaa, bkulis}@bu.edu

ABSTRACT

Existing automatic music generation approaches that feature deep learning can be broadly classified into two types: raw audio models and symbolic models. Symbolic models, which train and generate at the note level, are currently the more prevalent approach; these models can capture long-range dependencies of melodic structure, but fail to grasp the nuances and richness of raw audio generations. Raw audio models, such as DeepMind’s WaveNet, train directly on sampled audio waveforms, allowing them to produce realistic-sounding, albeit unstructured music. In this paper, we propose an automatic music generation methodology combining both of these approaches to create structured, realistic-sounding compositions. We consider a Long Short Term Memory network to learn the melodic structure of different styles of music, and then use the unique symbolic generations from this model as a conditioning input to a WaveNet-based raw audio generator, creating a model for automatic, novel music. We then evaluate this approach by showcasing results of this work.

1. INTRODUCTION

The ability of deep neural networks to generate novel musical content has recently become a popular area of research. Many variations of deep neural architectures have generated pop ballads,¹ helped artists write melodies,² and even have been integrated into commercial music generation tools.³

Current music generation methods are largely focused on generating music at the note level, resulting in outputs consisting of symbolic representations of music such as sequences of note numbers or MIDI-like streams of events. These methods, such as those based on Long Short Term Memory networks (LSTMs) and recurrent neural networks (RNNs), are effective at capturing medium-scale effects in music, can produce melodies with constraints such as mood and tempo, and feature fast generation times [14,22].

In order to create sound, these methods often require an intermediate step of interpretation of the output by humans, where the symbolic representation transitions to an audio output in some way.

An alternative is to train on and produce raw audio waveforms directly by adapting speech synthesis models, resulting in a richer palette of potential musical outputs, albeit at a higher computational cost. WaveNet, a model developed at DeepMind primarily targeted towards speech applications, has been applied directly to music; the model is trained to predict the next sample of 8-bit audio (typically sampled at 16 kHz) given the previous samples [25]. Initially, this was shown to produce rich, unique piano music when trained on raw piano samples. Follow-up work has developed faster generation times [16], generated synthetic vocals for music using WaveNet-based architectures [3], and has been used to generate novel sounds and instruments [8]. This approach to music generation, while very new, shows tremendous potential for music generation tools. However, while WaveNet produces more realistic sounds, the model does not handle medium or long-range dependencies such as melody or global structure in music. The music is expressive and novel, yet sounds unpracticed in its lack of musical structure.

Nonetheless, raw audio models show great potential for the future of automatic music. Despite the expressive nature of some advanced symbolic models, those methods require constraints such as mood and tempo to generate corresponding symbolic output [22]. While these constraints can be desirable in some cases, we express interest in generating structured raw audio directly due to the flexibility and versatility that raw audio provides; with no specification, these models are able to learn to generate expression and mood directly from the waveforms they are trained on. We believe that raw audio models are a step towards less guided, unsupervised music generation, since they are unconstrained in this way. With such tools for generating raw audio, one can imagine a number of new applications, such as the ability to edit existing raw audio in various ways.

Thus, we explore the combination of raw audio and symbolic approaches, opening the door to a host of new possibilities for music generation tools. In particular, we train a biaxial Long Short Term Memory network to create novel symbolic melodies, and then treat these melodies as an extra conditioning input to a WaveNet-based model. Consequently, the LSTM model allows us to represent long-range melodic structure in the music,

¹ <http://www.flow-machines.com/>

² <https://www.ampermusic.com/>

³ <https://www.jukedeck.com/>



while the WaveNet-based component interprets and expands upon the generated melodic structure in raw audio form. This serves to both eliminate the intermediate interpretation step of the symbolic representations and provide structure to the output of the raw audio model, while maintaining the aforementioned desirable properties of both models.

We first discuss the tuning of the original unconditioned WaveNet model to produce music of different instruments, styles, and genres. Once we have tuned this model appropriately, we then discuss our extension to the conditioned case, where we add a local conditioning technique to the raw audio model. This method is comparable to using a text-to-speech method within a speech synthesis model. We first generate audio from the conditioned raw audio model using well-known melodies (e.g., a C major scale and the Happy Birthday melody) after training on the MusicNet dataset [24]. We also discuss an application of our technique to editing existing raw audio music by changing some of the underlying notes and re-generating selections of audio. Then, we incorporate the LSTM generations as a unique symbolic component. We demonstrate results of training both the LSTM and our conditioned WaveNet-based model on corresponding training data, as well as showcase and evaluate generations of realistic raw audio melodies by using the output of the LSTM as a unique local conditioning time series to the WaveNet model.

This paper is an extension of an earlier work originally published as a workshop paper [19]. We augment that work-in-progress model in many aspects, including more concrete results, stronger evaluation, and new applications.

2. BACKGROUND

We elaborate on two prevalent deep learning models for music generation, namely raw audio models and symbolic models.

2.1 Raw Audio Models

Initial efforts to generate raw audio involved models used primarily for text generation, such as char-rnn [15] and LSTMs. Raw audio generations from these networks are often noisy and unstructured; they are limited in their capacity to abstract higher level representations of raw audio, mainly due to problems with overfitting [21].

In 2016, DeepMind introduced WaveNet [25], a generative model for general raw audio, designed mainly for speech applications. At a high level, WaveNet is a deep learning architecture that operates directly on a raw audio waveform. In particular, for a waveform modeled by a vector $x = \{x_1, \dots, x_T\}$ (representing speech, music, etc.), the joint probability of the entire waveform is factorized as a product of conditional probabilities, namely

$$p(x) = p(x_1) \prod_{t=2}^T p(x_t | x_1, \dots, x_{t-1}). \quad (1)$$

The waveforms in WaveNet are typically represented as 8-bit audio, meaning that each x_i can take on one of

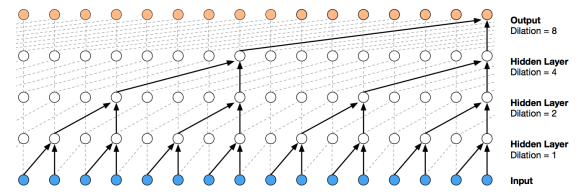


Figure 1: A stack of dilated causal convolutions as used by WaveNet, reproduced from [25].

256 possible values. The WaveNet model uses a deep neural network to model the conditional probabilities $p(x_t | x_1, \dots, x_{t-1})$. The model is trained by predicting values of the waveform at step t and comparing them to the true value x_t , using cross-entropy as a loss function; thus, the problem simply becomes a multi-class classification problem (with 256 classes) for each timestep in the waveform.

The modeling of conditional probabilities in WaveNet utilizes causal convolutions, similar to masked convolutions used in PixelRNN and similar image generation networks [7]. Causal convolutions ensure that the prediction for time step t only depends on the predictions for previous timesteps. Furthermore, the causal convolutions are dilated; these are convolutions where the filter is applied over an area larger than its length by skipping particular input values, as shown in Figure 1. In addition to dilated causal convolutions, each layer features gated activation units and residual connections, as well as skip connections to the final output layers.

2.2 Symbolic Audio Models

Most deep learning approaches for automatic music generation are based on symbolic representations of the music. MIDI (Musical Instrument Digital Interface),⁴ for example, is a ubiquitous standard for file format and protocol specification for symbolic representation and transmission. Other representations that have been utilized include the piano roll representation [13]—inspired by player piano music rolls—text representations (e.g., ABC notation⁵), chord representations (e.g., Chord2Vec [18]), and lead sheet representations. A typical scenario for producing music in such models is to train and generate on the same type of representation; for instance, one may train on a set of MIDI files that encode melodies, and then generate new MIDI melodies from the learned model. These models attempt to capture the aspect of long-range dependency in music.

A traditional approach to learning temporal dependencies in data is to use recurrent neural networks (RNNs). A recurrent neural network receives a timestep of a series x_t along with a hidden state h_t as input. It outputs y_t , the model output at that timestep, and computes h_{t+1} , the hidden state at the next timestep. RNNs take advantage of

⁴ <https://www.midi.org/specifications>

⁵ <http://abcnotation.com>

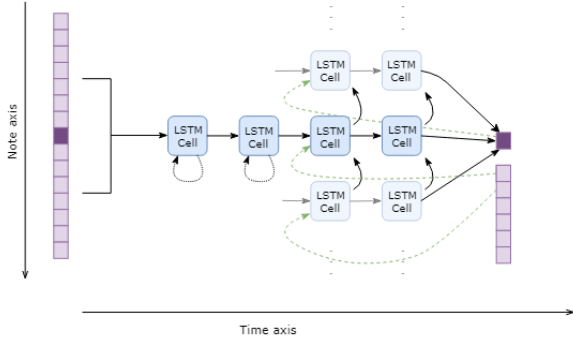


Figure 2: A representation of a biaxial LSTM network. Note that the first two layers have connections across timesteps, while the last two layers have recurrent connections across notes [14].

this hidden state to store some information from the previous timesteps. In practice, vanilla RNNs do not perform well when training sequences have long temporal dependencies due to issues of vanishing/exploding gradients [2]. This is especially true for music, as properties such as key signature and time signature may be constant throughout a composition.

Long Short Term Memory networks are a variant of RNNs that have proven useful in symbolic music generation systems. LSTM networks modify the way memory information is stored in RNNs by introducing another unit to the original RNN network: the cell state, c_t , where the flow of information is controlled by various gates. LSTMs are designed such that the interaction between the cell state and the hidden state prevents the issue of vanishing/exploding gradients [10, 12].

There are numerous existing deep learning symbolic music generation approaches [5], including models that are based on RNNs, many of which use an LSTM as a key component of the model. Some notable examples include DeepBach [11], the CONCERT system [20], the Celtic Melody Generation system [23] and the Biaxial LSTM model [14]. Additionally, some approaches combine RNNs with restricted Boltzmann machines [4, 6, 9, 17].

3. ARCHITECTURE

We first discuss our symbolic method for generating unique melodies, then detail the modifications to the raw audio model for compatibility with these generations. Modifying the architecture involves working with both symbolic and raw audio data in harmony.

3.1 Unique Symbolic Melody Generation with LSTM Networks

Recently, applications of LSTMs specific to music generation, such as the biaxial LSTM, have been implemented and explored. This model utilizes a pair of tied, parallel networks to impose LSTMs both in the temporal dimension and the pitch dimension at each timestep. Each note has its own network instance at each timestep, and

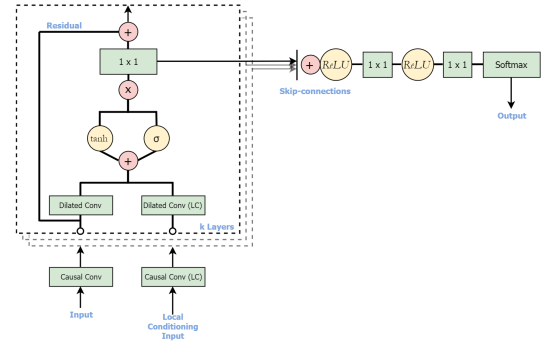


Figure 3: An overview of the model architecture, showing the local conditioning time series as an extra input.

receives input of the MIDI note number, pitchclass, beat, and information on surrounding notes and notes at previous timesteps. This information first passes through two layers with connections across timesteps, and then two layers with connections across notes, detailed in Figure 2. This combination of note dependency and temporal dependency allow the model to not only learn the overall instrumental and temporal structure of the music, but also capture the interdependence of the notes being played at any given timestep [14].

We explore the sequential combination of the symbolic and raw audio models to produce structured raw audio output. We train a biaxial LSTM model on the MIDI files of a particular genre of music as training data, and then feed the MIDI generations from this trained model into the raw audio generator model.

3.2 Local Conditioning with Raw Audio Models

Once a learned symbolic melody is obtained, we treat it as a second time series within our raw audio model (analogous to using a second time series with a desired text to be spoken in the speech domain). In particular, in the WaveNet model, each layer features a gated activation unit. If x is the raw audio input vector, then at each layer k , it passes through the following gated activation unit:

$$z = \tanh(W_{f,k} * x) \odot \sigma(W_{g,k} * x), \quad (2)$$

where $*$ is a convolution operator, \odot is an elementwise multiplication operator, $\sigma(\cdot)$ is the sigmoid function, and the $W_{f,k}$ and $W_{g,k}$ are learnable convolution filters. Following WaveNet's use of local conditioning, we can introduce a second time series y (in this case from the LSTM model, to capture the long-term melody), and instead utilize the following activation, effectively incorporating y as an extra input:

$$z = \tanh(W_{f,k} * x + V_{f,k} * y) \odot \sigma(W_{g,k} * x + V_{g,k} * y), \quad (3)$$

where V are learnable linear projections. By conditioning on an extra time series input, we effectively guide the raw audio generations to require certain characteristics; y influences the output at all timestamps.

Instrument	Minutes	Labels
Piano	1,346	794,532
Violin	874	230,484
Cello	621	99,407
Solo Piano	917	576,471
Solo Violin	30	8,837
Solo Cello	49	10,876

Table 1: Statistics of the MusicNet dataset. [24]

In our modified WaveNet model, the second time series y is the upsampled MIDI embedding of the local conditioning time series. In particular, local conditioning (LC) embeddings are 128-dimensional binary vectors, where ones correspond to note indices that are being played at the current timestep. As with the audio time series, the LC embeddings first go through a layer of causal convolutions to reduce the number of dimensions from 128 to 16, which are then used in the dilation layers as the conditioning samples. This reduces the computational requirement for the dilation layers without reducing the note state information, as most of the embeddings are zero for most timestamps. This process along with the surrounding architecture is shown in Figure 3.

3.3 Hyperparameter Tuning

Table 2 enumerates the hyperparameters used in the WaveNet-based conditioned model to obtain our results. We note that the conditioned model needs only 30 dilation layers as compared to the 50 we had used in the unconditioned network. Training with these parameters gave us comparable results as compared to the unconditioned model in terms of the timbre of instruments and other nuances in generations. This indicates that the decrease in parameters is offset by the extra information provided by the conditioning time series.

4. EMPIRICAL EVALUATION

Example results of generations from our models are posted on our web page.⁶

One of the most challenging tasks in automated music generation is evaluating the resulting music. Any generated piece of music can generally only be subjectively evaluated by human listeners. Here, we qualitatively evaluate our results to the best of our ability, but leave the results on our web page for the reader to subjectively evaluate. We additionally quantify our results by comparing the resulting loss functions of the unconditioned and conditioned raw audio models. Then, we evaluate the structural component by computing the cross-correlation between the spectrogram of the generated raw audio and conditioning input.

4.1 Training Datasets and Loss Analysis

At training time, in addition to raw training audio, we must also incorporate its underlying symbolic melody, perfectly

⁶ <http://people.bu.edu/bkulis/projects/music/index.html>

Hyperparameter	Value
Initial Filter Width	32
Dilation Filter Width	2
Dilation Layers	30
Residual Channels	32
Dilation Channels	32
Skip Channels	512
Initial LC Channels	128
Dilation LC Channels	16
Quantization Channels	128

Table 2: WaveNet hyperparameters used for training of the conditioned network.

aligned with the raw audio at each timestep. The problem of melody extraction in raw audio is still an active area of research; due to a general lack of such annotated music, we have experimented with multiple datasets.

Primarily, we have been exploring use of the recently-released MusicNet database for training [24], as this data features both raw audio as well as melodic annotations. Other metadata is also included, such as the composer of the piece, the instrument with which the composition is played, and each note’s position in the metrical structure of the composition. The music is separated by genre; there are over 900 minutes of solo piano alone, which has proven to be very useful in training on only one instrument. The different genres provide many different options for training. Table 1 shows some other statistics of the MusicNet dataset.

After training with these datasets, we have found that the loss for the unconditioned and conditioned WaveNet models follows our expectation of the conditioned model exhibiting a lower cross-entropy training loss than the unconditioned model. This is due to the additional embedding information provided along with the audio in the conditioned case. Figure 5 shows the loss for two WaveNet models trained on the MusicNet cello dataset over 100,000 iterations, illustrating this decreased loss for the conditioned model.

4.2 Unconditioned Music Generation with WaveNet

We preface the evaluation of our musical results by acknowledging the fact that we first tuned WaveNet for unstructured music generation, as most applications of WaveNet have explored speech applications. Here we worked in the unconditioned case, i.e., no second time series was input to the network. We tuned the model to generate music trained on solo piano inputs (about 50 minutes of the Chopin nocturnes, from the YouTube-8M dataset [1]), as well as 350 songs of various genres of electronic dance music, obtained from No Copyright Sounds⁷.

We found that WaveNet models are capable of producing lengthy, complex musical generations without losing instrumental quality for solo instrumental training data. The network is able to learn short-range dependencies, in-

⁷ <https://www.youtube.com/user/NoCopyrightSounds>



Figure 4: Example MIDI generation from the biaxial LSTM trained on cello music, visualized as sheet music.

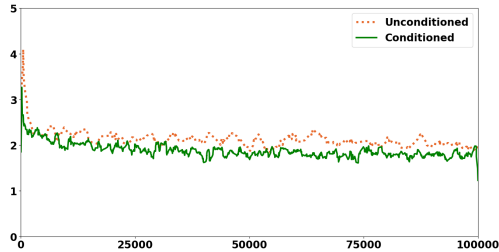


Figure 5: Cross entropy loss for the conditioned (solid green) and unconditioned (dotted orange) WaveNet models over the first 100,000 training iterations, illustrating the lower training loss of the conditioned model.

cluding hammer action and simple chords. Although generations may have a consistent energy, they are unstructured and do not contain any long-range temporal dependencies. Results that showcase these techniques and attributes are available on our webpage.

4.3 Structure in Raw Audio Generations

We evaluate the structuring ability of our conditioned raw audio model for a generation based on how closely it follows the conditioning signal it was given, first using popular existing melodies, then the unique LSTM generations. We use cross-correlation as a quantitative evaluation method. We also acknowledge the applications of our model to edit existing raw audio.

4.3.1 Raw Audio from Existing Melodies

We evaluate our approach first by generating raw audio from popular existing melodies, by giving our conditioned model a second time series input of the Happy Birthday melody and a C major scale. Since we are familiar with these melodies, they are easier to evaluate by ear.

Initial versions of the model evaluated in this way were trained on the MusicNet cello dataset. The generated raw audio follows the conditioning input, the recognizable Happy Birthday melody and C major scale, in a cello timbre. The results of these generations are uploaded on our webpage.

4.3.2 Raw Audio From Unique LSTM Generations

After generating novel melodies from the LSTM, we produced corresponding output from our conditioned model. Since it is difficult to qualitatively evaluate such melodies



(a) Unedited training sample from the MusicNet dataset.



(b) Slightly modified training sample.

Figure 6: MIDI representations of a sample from the MusicNet solo cello dataset, visualized as sheet music; (b) is a slightly modified version of (a), the original training sample. We use these samples to showcase the ability of our model to “edit” raw audio.

by ear due to unfamiliarity with the melody, we are interested in evaluating how accurately the conditioned model follows a novel melody quantitatively. We evaluate our results by computing the cross-correlation between the MIDI sequence and the spectrogram of the generated raw audio as shown in Figure 7. Due to the sparsity of both the spectrogram and the MIDI file in the frequency dimension, we decided to calculate the cross-correlation between one-dimensional representations of the two time series. We chose the frequency of the highest note in the MIDI at each timestep as its one-dimensional representation. In the case of the raw audio, we chose the most active frequency in its spectrogram at each timestep. We acknowledge some weakness in this approach, since some information is lost by reducing the dimensionality of both time series.

Cross-correlation is the “sliding dot product” of two time series — a measure of linear similarity as a function of the displacement of one series relative to the other. In this instance, the cross-correlation between the MIDI sequence and the corresponding raw audio peaks at delay 0 and is equal to 0.3. In order to assure that this correlation is not due to chance, we have additionally calculated the cross-correlation between the generated raw audio and 50 different MIDI sequences in the same dataset. In Figure 7, we can see that the cross-correlation curve stays above the other random correlation curves in the area around delay 0. This shows that the correlation found is not by chance, and the raw audio output follows the conditioning vector appropriately.

This analysis generalizes to any piece generated with our model; we have successfully been able to transform an unstructured model with little long-range dependency to one with generations that exhibit certain characteristics.

4.3.3 Editing Existing Raw Audio

In addition, we explored the possibility of using our approach as a tool similar to a MIDI synthesizer, where we first generate from an existing piece of a symbolic melody,

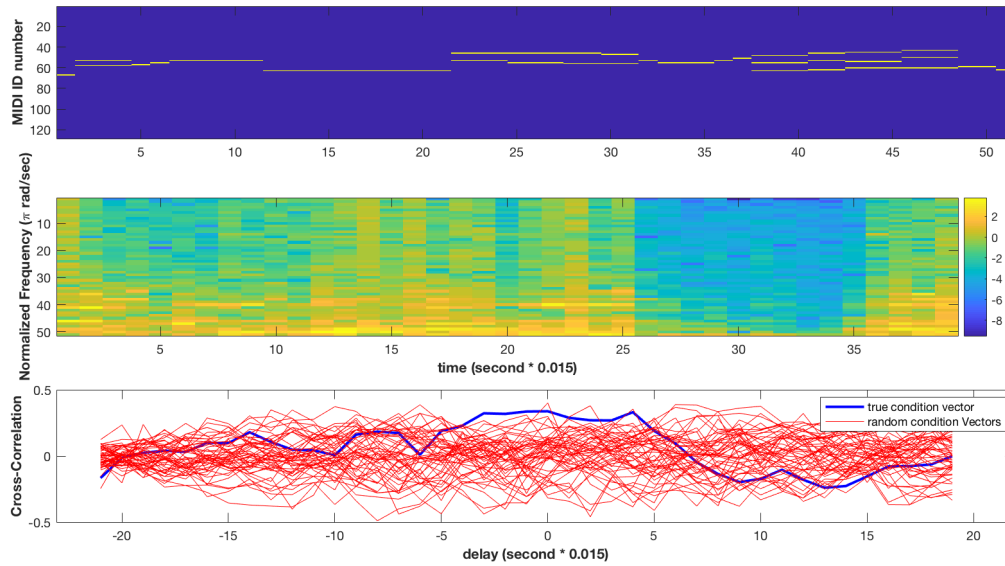


Figure 7: Comparison of the novel LSTM-generated melody (top) and the corresponding raw audio output of the conditioned model represented as a spectrogram (middle). The bottom plot shows the cross-correlation between the frequency of the highest note of the MIDI and the most active frequency of raw audio from the WaveNet-based model, showing strong conditioning from the MIDI on the generated audio.

in this case, from the training data. Then, we generate new audio by making small changes to the MIDI, and evaluate how the edits reflect in the generated audio. We experiment with this with the goal of achieving a higher level of fidelity to the audio itself rather using a synthesizer to replay the MIDI as audio, as that often forgoes the nuances associated with raw audio.

Figure 6(a) and 6(b) respectively show a snippet of the training data taken from the MusicNet cello dataset and the small perturbations made to it, which were used to evaluate this approach. The results posted on our webpage show that the generated raw audio retains similar characteristics between the original and the edited melody, while also incorporating the changes to the MIDI in an expressive way.

5. CONCLUSIONS AND FUTURE WORK

In conclusion, we focus on combining raw and symbolic audio models for the improvement of automatic music generation. Combining two prevalent models allows us to take advantage of both of their features; in the case of raw audio models, this is the realistic sound and feel of the music, and in the case of symbolic models, it is the complexity, structure, and long-range dependency of the generations.

Before continuing to improve our work, we first plan to more thoroughly evaluate our current model using ratings of human listeners. We will use crowdsourced evaluation techniques (specifically, Amazon Mechanical Turk⁸) to compare our outputs with other systems.

A future modification of our approach is to merge the LSTM and WaveNet models to a coupled architecture.

This joint model would eliminate the need to synthesize MIDI files, as well as the need for MIDI labels aligned with raw audio data. In essence, this adjustment would create a true end-to-end automatic music generation model.

Additionally, DeepMind recently updated the WaveNet model to improve generation speed by 1000 times over the previous model, at 16 bits per sample and a sampling rate of 24kHz [26]. We hope to investigate this new model to develop real-time generation of novel, structured music, which has many significant implications.

The potential results of our work could augment and inspire many future applications. The combination of our model with multiple audio domains could be implemented; this could involve the integration of speech audio with music to produce lyrics sung in tune with our realistic melody.

Even without the additional improvements considered above, the architecture proposed in this paper allows for a modular approach to automated music generation. Multiple different instances of our conditioned model can be trained on different genres of music, and generate based on a single local conditioning series in parallel. As a result, the same melody can be reproduced in different genres or instruments, strung together to create effects such as a quartet or a band. The key application here is that this type of synchronized effect can be achieved without awareness of the other networks, avoiding model interdependence.

6. ACKNOWLEDGEMENT

We would like to acknowledge that this research was supported in part by NSF CAREER Award 1559558.

⁸ <https://www.mturk.com/mturk/>

7. REFERENCES

- [1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. YouTube-8M: A large-scale video classification benchmark. *CoRR*, abs/1609.08675, 2016.
- [2] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
- [3] M. Blaauw and J. Bonada. A neural parametric singing synthesizer. *ArXiv preprint 1704.03809*, 2017.
- [4] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 26 Jun–1 Jul 2012.
- [5] J. Briot, G. Hadjeres, and F. Pachet. Deep learning techniques for music generation—a survey. *ArXiv preprint 1709.01620*, 2017.
- [6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv preprint 1412.3555*, 2014.
- [7] A. Van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756, New York, New York, USA, 20–22 Jun 2016.
- [8] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan. Neural audio synthesis of musical notes with WaveNet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1068–1077, International Convention Centre, Sydney, Australia, 06–11 Aug 2017.
- [9] K. Goel, R. Vohra, and JK Sahoo. Polyphonic music generation by modeling temporal dependencies using a rnn-dbn. In *International Conference on Artificial Neural Networks*, pages 217–224. Springer, 2014.
- [10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [11] G. Hadjeres, F. Pachet, and F. Nielsen. DeepBach: a steerable model for Bach chorales generation. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1362–1371, International Convention Centre, Sydney, Australia, 06–11 Aug 2017.
- [12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [13] A. Huang and R. Wu. Deep learning for music. *ArXiv preprint 1606.04930*, 2016.
- [14] D. D. Johnson. Generating polyphonic music using tied parallel networks. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 128–143. Springer, 2017.
- [15] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078, 2015.
- [16] T. Le Paine, P. Khorrami, S. Chang, Y. Zhang, P. Ramachandran, M. A. Hasegawa-Johnson, and T. S. Huang. Fast waveNet generation algorithm. *ArXiv preprint 1611.09482*, 2016.
- [17] Q. Lyu, J. Zhu Z. Wu, and H. Meng. Modelling high-dimensional sequences with LSTM-RTRBM: Application to polyphonic music generation. In *Proc. International Artificial Intelligence Conference (AAAI)*, 2015.
- [18] S. Madjiheurem, L. Qu, and C. Walder. Chord2Vec: Learning musical chord embeddings. In *Proceedings of the Constructive Machine Learning Workshop at 30th Conference on Neural Information Processing Systems*, Barcelona, Spain, 2016.
- [19] R. Manzelli, V. Thakkar, A. Siahkamari, and B. Kulis. An end to end model for automatic music generation: Combining deep raw and symbolic audio networks. In *Proceedings of the Musical Metacreation Workshop at 9th International Conference on Computational Creativity*, Salamanca, Spain, 2018.
- [20] M. C. Mozer. Neural network composition by prediction: Exploring the benefits of psychophysical constraints and multiscale processing. *Connection Science*, 6(2–3):247–280, 1994.
- [21] A. Nayeibi and M. Vitelli. Gruv: Algorithmic music generation using recurrent neural networks. *Course CS224D: Deep Learning for Natural Language Processing (Stanford)*, 2015.
- [22] I. Simon and S. Oore. Performance RNN: Generating music with expressive timing and dynamics, 2017. <https://magenta.tensorflow.org/performance-rnn>.
- [23] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova. Music transcription modelling and composition using deep learning. *ArXiv preprint 1604.08723*, 2016.
- [24] J. Thickstun, Z. Harchaoui, and S. M Kakade. Learning features of music from scratch. In *International Conference on Learning Representations (ICLR)*, 2017.
- [25] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A generative model for raw audio. *ArXiv preprint 1609.03499*, 2016.

- [26] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis. Parallel wavenet: Fast high-fidelity speech synthesis. *CoRR*, abs/1711.10433, 2017.

CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS WITH BINARY NEURONS FOR POLYPHONIC MUSIC GENERATION

Hao-Wen Dong and Yi-Hsuan Yang

Research Center for IT innovation, Academia Sinica, Taipei, Taiwan

{salu133445,yang}@citi.sinica.edu.tw

ABSTRACT

It has been shown recently that deep convolutional generative adversarial networks (GANs) can learn to generate music in the form of piano-rolls, which represent music by binary-valued time-pitch matrices. However, existing models can only generate real-valued piano-rolls and require further post-processing, such as hard thresholding (HT) or Bernoulli sampling (BS), to obtain the final binary-valued results. In this paper, we study whether we can have a convolutional GAN model that directly creates binary-valued piano-rolls by using binary neurons. Specifically, we propose to append to the generator an additional refiner network, which uses binary neurons at the output layer. The whole network is trained in two stages. Firstly, the generator and the discriminator are pretrained. Then, the refiner network is trained along with the discriminator to learn to binarize the real-valued piano-rolls the pretrained generator creates. Experimental results show that using binary neurons instead of HT or BS indeed leads to better results in a number of objective measures. Moreover, deterministic binary neurons perform better than stochastic ones in both objective measures and a subjective test. The source code, training data and audio examples of the generated results can be found at <https://salu133445.github.io/bmusegan/>.

1. INTRODUCTION

Recent years have seen increasing research on symbolic-domain music generation and composition using deep neural networks [7]. Notable progress has been made to generate monophonic melodies [25,27], lead sheets (i.e., melody and chords) [8, 11, 26], or four-part chorales [14]. To add something new to the table and to increase the polyphony and the number of instruments of the generated music, we attempt to generate piano-rolls in this paper, a music representation that is more general (e.g., comparing to lead-sheets) yet less studied in recent work on music generation. As Figure 1 shows, we can consider an M -track piano-roll

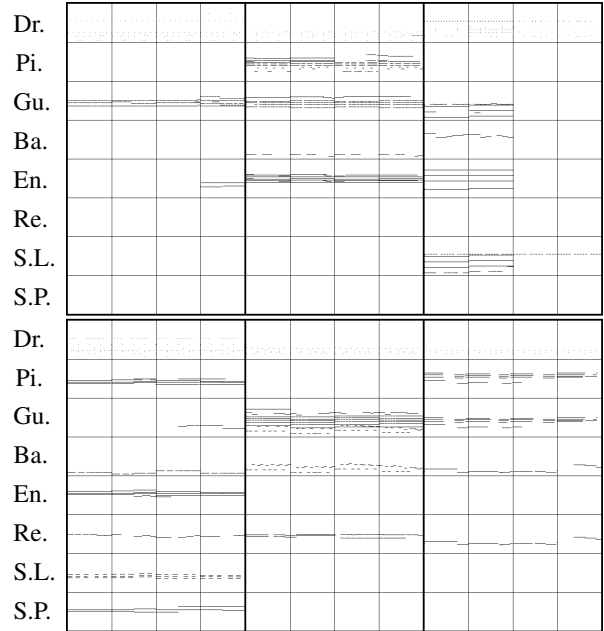


Figure 1. Six examples of eight-track piano-roll of four-bar long (each block represents a bar) seen in our training data. The vertical and horizontal axes represent note pitch and time, respectively. The eight tracks are *Drums*, *Piano*, *Guitar*, *Bass*, *Ensemble*, *Reed*, *Synth Lead* and *Synth Pad*.

as a collection of M binary time-pitch matrices indicating the presence of pitches per time step for each track.

Generating piano-rolls is challenging because of the large number of possible active notes per time step and the involvement of multiple instruments. Unlike a melody or a chord progression, which can be viewed as a sequence of note/chord events and be modeled by a recurrent neural network (RNN) [21,24], the musical texture in a piano-roll is much more complicated (see Figure 1). While RNNs are good at learning the temporal dependency of music, convolutional neural networks (CNNs) are usually considered better at learning local patterns [18].

For this reason, in our previous work [10], we used a convolutional generative adversarial network (GAN) [12] to learn to generate piano-rolls of five tracks. We showed that the model generates music that exhibit drum patterns and plausible note events. However, musically the generated result is still far from satisfying to human ears, scoring around 3 on average on a five-level Likert scale in overall



© Hao-Wen Dong and Yi-Hsuan Yang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: Hao-Wen Dong and Yi-Hsuan Yang. "Convolutional Generative Adversarial Networks with Binary Neurons for Polyphonic Music Generation", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

quality in a user study [10].¹

There are several ways to improve upon this prior work. The major topic we are interested in is the introduction of the *binary neurons* (BNs) [1, 4] to the model. We note that conventional CNN designs, also the one adopted in our previous work [10], can only generate real-valued predictions and require further postprocessing *at test time* to obtain the final binary-valued piano-rolls.² This can be done by either applying a *hard threshold* (HT) on the real-valued predictions to binarize them (which was done in [10]), or by treating the real-valued predictions as probabilities and performing *Bernoulli sampling* (BS).

However, we note that such naïve methods for binarizing a piano-roll can easily lead to *overly-fragmented notes*. For HT, this happens when the original real-valued piano-roll has many entries with values close to the threshold. For BS, even an entry with low probability can take the value 1, due to the stochastic nature of probabilistic sampling.

The use of BNs can mitigate the aforementioned issue, since the binarization is part of the training process. Moreover, it has two potential benefits:

- In [10], binarization of the output of the generator G in GAN is done only at test time not at training time (see Section 2.1 for a brief introduction of GAN). This makes it easy for the discriminator D in GAN to distinguish between the generated piano-rolls (which are real-valued in this case) and the real piano-rolls (which are binary). With BNs, the binarization is done at training time as well, so D can focus on extracting musically relevant features.
- Due to BNs, the input to the discriminator D in GAN at training time is binary instead of real-valued. This effectively reduces the model space from \mathbb{R}^N to 2^N , where N is the product of the number of time steps and the number of possible pitches. Training D may be easier as the model space is substantially smaller, as Figure 2 illustrates.

Specifically, we propose to append to the end of G a *refiner network* R that uses either deterministic BNs (DBNs) or stochastic BNs (SBNs) at the output layer. In this way, G makes real-valued predictions and R binarizes them. We train the whole network in two stages: in the first stage we pretrain G and D and then fix G ; in the second stage, we train R and fine-tune D . We use residual blocks [16] in R to make this two-stage training feasible (see Section 3.3).

As minor contributions, we use a new shared/private design of G and D that cannot be found in [10]. Moreover, we add to D two streams of layers that provide onset/offset and chroma information (see Sections 3.2 and 3.4).

The proposed model is able to directly generate binary-valued piano-rolls at test time. Our analysis shows that the

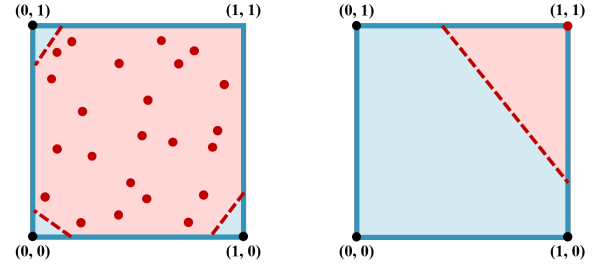


Figure 2. An illustration of the decision boundaries (red dashed lines) that the discriminator D has to learn when the generator G outputs (left) real values and (right) binary values. The decision boundaries divide the space into the *real* class (in blue) and the *fake* class (in red). The black and red dots represent the real data and the fake ones generated by the generator, respectively. We can see that the decision boundaries are easier to learn when the generator outputs binary values rather than real values.

generated results of our model with DBNs features fewer overly-fragmented notes as compared with the result of using HT or BS. Experimental results also show the effectiveness of the proposed two-stage training strategy compared to either a joint or an end-to-end training strategy.

2. BACKGROUND

2.1 Generative Adversarial Networks

A generative adversarial network (GAN) [12] has two core components: a *generator* G and a *discriminator* D . The former takes as input a random vector \mathbf{z} sampled from a prior distribution $p_{\mathbf{z}}$ and generates a fake sample $G(\mathbf{z})$. D takes as input either real data \mathbf{x} or fake data generated by G . During training time, D learns to distinguish the fake samples from the real ones, whereas G learns to fool D .

An alternative form called WGAN was later proposed with the intuition to estimate the Wasserstein distance between the real and the model distributions by a deep neural network and use it as a critic for the generator [2]. The objective function for WGAN can be formulated as:

$$\min_G \max_D \mathbb{E}_{\mathbf{x} \sim p_d} [D(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p_z} [D(G(\mathbf{z}))], \quad (1)$$

where p_d denotes the real data distribution. In order to enforce Lipschitz constraints on the discriminator, which is required in the training of WGAN, Gulrajani *et al.* [13] proposed to add to the objective function of D a *gradient penalty* (GP) term: $\mathbb{E}_{\tilde{\mathbf{x}} \sim p_{\tilde{\mathbf{x}}}} [(\|\nabla_{\tilde{\mathbf{x}}} D(\tilde{\mathbf{x}})\| - 1)^2]$, where $p_{\tilde{\mathbf{x}}}$ is defined as sampling uniformly along straight lines between pairs of points sampled from p_d and the model distribution p_g . Empirically they found it stabilizes the training and alleviates the mode collapse issue, compared to the weight clipping strategy used in the original WGAN. Hence, we employ WGAN-GP [13] as our generative framework.

2.2 Stochastic and Deterministic Binary Neurons

Binary neurons (BNs) are neurons that output binary-valued predictions. In this work, we consider two types of

¹ Another related work on generating piano-rolls, as presented by Boulanger-Lewandowski *et al.* [6], replaced the output layer of an RNN with conditional restricted Boltzmann machines (RBMs) to model high-dimensional sequences and applied the model to generate piano-rolls sequentially (i.e. one time step after another).

² Such binarization is typically not needed for an RNN or an RBM in polyphonic music generation, since an RNN is usually used to predict pre-defined note events [22] and an RBM is often used with binary visible and hidden units and sampled by Gibbs sampling [6, 20].

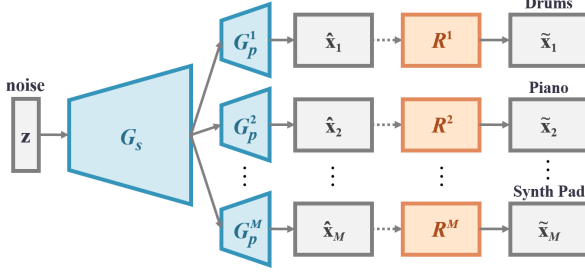


Figure 3. The generator and the refiner. The generator (G_s and several G_p^i collectively) produces real-valued predictions. The refiner network (several R^i) refines the outputs of the generator into binary ones.



Figure 4. The refiner network. The tensor size remains the same throughout the network.

BNs: deterministic binary neurons (DBNs) and stochastic binary neurons (SBNs). DBNs act like neurons with *hard thresholding* functions as their activation functions. We define the output of a DBN for a real-valued input x as:

$$DBN(x) = u(\sigma(x) - 0.5), \quad (2)$$

where $u(\cdot)$ denotes the unit step function and $\sigma(\cdot)$ is the logistic sigmoid function. SBNs, in contrast, binarize an input x according to a probability, defined as:

$$SBN(x) = u(\sigma(x) - v), \quad v \sim U[0, 1], \quad (3)$$

where $U[0, 1]$ denotes a uniform distribution.

2.3 Straight-through Estimator

Computing the exact gradients for either DBNs or SBNs, however, is intractable. For SBNs, it requires the computation of the average loss over all possible binary samplings of all the SBNs, which is exponential in the total number of SBNs. For DBNs, the threshold function in Eq. (2) is non-differentiable. Therefore, the flow of backpropagation used to train parameters of the network would be blocked.

A few solutions have been proposed to address this issue [1, 4]. One strategy is to replace the non-differentiable functions, which are used in the forward pass, by differentiable functions (usually called the *estimators*) in the backward pass. An example is the *straight-through* (ST) estimator proposed by Hinton [17]. In the backward pass, ST simply treats BNs as identity functions and ignores their gradients. A variant of the ST estimator is the *sigmoid-adjusted ST estimator* [9], which multiplies the gradients in the backward pass by the derivative of the sigmoid function. Such estimators were originally proposed as regularizers [17] and later found promising for conditional computation [4]. We use the sigmoid-adjusted ST estimator in training neural networks with BNs and found it empirically works well for our generation task as well.

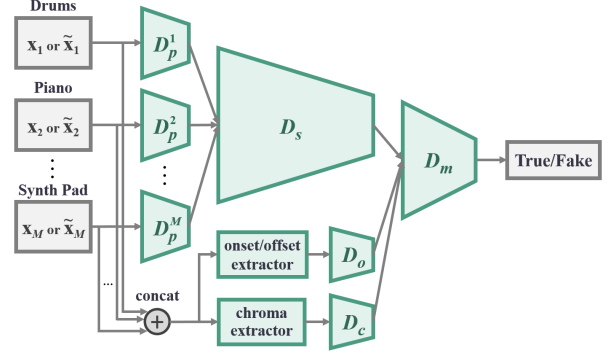


Figure 5. The discriminator. It consists of three streams: the main stream (D_m , D_s and several D_p^i ; the upper half), the onset/offset stream (D_o) and the chroma stream (D_c).

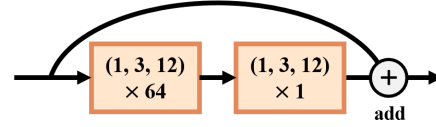


Figure 6. Residual unit used in the refiner network. The values denote the kernel size and the number of the output channels of the two convolutional layers.

3. PROPOSED MODEL

3.1 Data Representation

Following [10], we use the *multi-track piano-roll* representation. A multi-track piano-roll is defined as a set of piano-rolls for different tracks (or instruments). Each piano-roll is a binary-valued score-like matrix, where its vertical and horizontal axes represent note pitch and time, respectively. The values indicate the presence of notes over different time steps. For the temporal axis, we discard the tempo information and therefore every beat has the same length regardless of tempo.

3.2 Generator

As Figure 3 shows, the generator G consists of a “s”hared network G_s followed by M “p”rivate network G_p^i , $i = 1, \dots, M$, one for each track. The shared generator G_s first produces a high-level representation of the output musical segments that is shared by all the tracks. Each private generator G_p^i then turns such abstraction into the final piano-roll output for the corresponding track. The intuition is that different tracks have their own musical properties (e.g., textures, common-used patterns), while jointly they follow a common, high-level musical idea. The design is different from [10] in that the latter does not include a shared G_s in early layers.

3.3 Refiner

The refiner R is composed of M private networks R^i , $i = 1, \dots, M$, again one for each track. The refiner aims to refine the real-valued outputs of the generator, $\hat{x} = G(z)$, into binary ones, \tilde{x} , rather than learning a new mapping

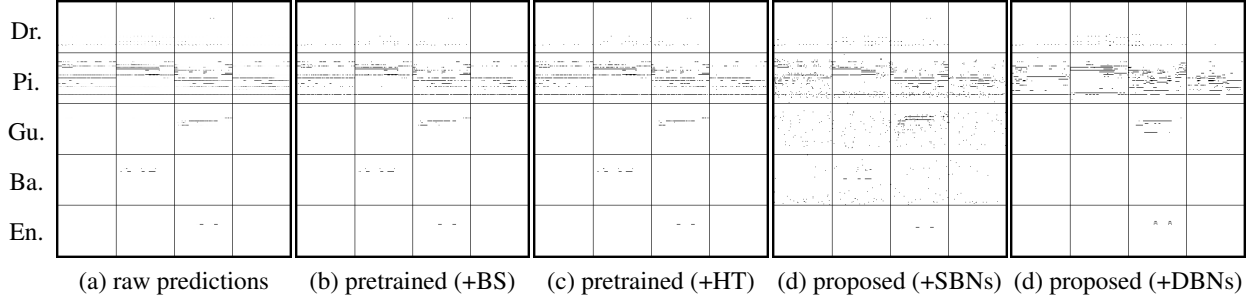


Figure 7. Comparison of binarization strategies. (a): the probabilistic, real-valued (raw) predictions of the pretrained G . (b), (c): the results of applying post-processing algorithms directly to the raw predictions in (a). (d), (e): the results of the proposed models, using an additional refiner R to binarize the real-valued predictions of G . Empty tracks are not shown. (We note that in (d), few noises (33 pixels) occur in the *Reed* and *Synth Lead* tracks.)

from $G(\mathbf{z})$ to the data space. Hence, we draw inspiration from *residual learning* and propose to construct the refiner with a number of *residual units* [16], as shown in Figure 4. The output layer (i.e. the final layer) of the refiner is made up of either DBNs or SBNs.

3.4 Discriminator

Similar to the generator, the discriminator D consists of M private network D_p^i , $i = 1, \dots, M$, one for each track, followed by a shared network D_s , as shown in Figure 5. Each private network D_p^i first extracts low-level features from the corresponding track of the input piano-roll. Their outputs are concatenated and sent to the shared network D_s to extract higher-level abstraction shared by all the tracks. The design differs from [10] in that only one (shared) discriminator was used in [10] to evaluate all the tracks collectively. We intend to evaluate such a new shared/private design in Section 4.5.

As a minor contribution, to help the discriminator extract musically-relevant features, we propose to add to the discriminator two more streams, shown in the lower half of Figure 5. In the first *onset/offset stream*, the differences between adjacent elements in the piano-roll along the time axis are first computed, and then the resulting matrix is summed along the pitch axis, which is finally fed to D_o .

In the second *chroma stream*, the piano-roll is viewed as a sequence of one-beat-long frames. A chroma vector is then computed for each frame and jointly form a matrix, which is then be fed to D_c . Note that all the operations involved in computing the chroma and onset/offset features are differentiable, and thereby we can still train the whole network by backpropagation.

Finally, the features extracted from the three streams are concatenated and fed to D_m to make the final prediction.

3.5 Training

We propose to train the model in a **two-stage** manner: G and D are pretrained in the first stage; R is then trained along with D (fixing G) in the second stage. Other training strategies are discussed and compared in Section 4.4.

4. ANALYSIS OF THE GENERATED RESULTS

4.1 Training Data & Implementation Details

The Lakh Pianoroll Dataset (LPD) [10]³ contains 174,154 multi-track piano-rolls derived from the MIDI files in the Lakh MIDI Dataset (LMD) [23].⁴ In this paper, we use a cleansed subset (*LPD-cleansed*) as the training data, which contains 21,425 multi-track piano-rolls that are in 4/4 time and have been matched to distinct entries in Million Song Dataset (MSD) [5]. To make the training data cleaner, we consider only songs with an *alternative* tag. We randomly pick six four-bar phrases from each song, which leads to the final training set of 13,746 phrases from 2,291 songs.

We set the temporal resolution to 24 time steps per beat to cover common temporal patterns such as triplets and 32th notes. An additional one-time-step-long pause is added between two consecutive (i.e. without a pause) notes of the same pitch to distinguish them from one single note. The note pitch has 84 possibilities, from C1 to B7.

We categorize all instruments into drums and sixteen instrument families according to the specification of General MIDI Level 1.⁵ We discard the less popular instrument families in LPD and use the following eight tracks: *Drums*, *Piano*, *Guitar*, *Bass*, *Ensemble*, *Reed*, *Synth Lead* and *Synth Pad*. Hence, the size of the target output tensor is 4 (bar) \times 96 (time step) \times 84 (pitch) \times 8 (track).

Both G and D are implemented as CNNs. The length of the input random vector is 128. R consists of two residual units [16] shown in Figure 6. Following [13], we use the Adam optimizer [19] and only apply batch normalization to G and R . We apply the *slope annealing trick* [9] to networks with BNs, where the slope of the sigmoid function in the sigmoid-adjusted ST estimator is multiplied by 1.1 after each epoch. The batch size is 16 except for the first stage in the two-stage training setting, where the batch size is 32. For more details, we refer readers to the online appendix, which can be found on the project website.⁶

³<https://salu133445.github.io/lakh-pianoroll-dataset/>

⁴<http://colinraffel.com/projects/lmd/>

⁵<https://www.midi.org/specifications/item/gm-level-1-sound-set>

⁶<https://salu133445.github.io/bmusegan/>

	training data	pretrained		proposed		joint		end-to-end		ablated-I		ablated-II	
		BS	HT	SBNs	DBNs	SBNs	DBNs	SBNs	DBNs	BS	HT	BS	HT
QN	0.88	0.67	0.72	0.42	0.78	0.18	0.55	0.67	0.28	0.61	0.64	0.35	0.37
PP	0.48	0.20	0.22	0.26	0.45	0.19	0.19	0.16	0.29	0.19	0.20	0.14	0.14
TD	0.96	0.98	1.00	0.99	0.87	0.95	1.00	1.40	1.10	1.00	1.00	1.30	1.40

(Underlined and bold font indicate respectively the top and top-three entries with values closest to those shown in the ‘training data’ column.)

Table 1. Evaluation results for different models. Values closer to those reported in the ‘training data’ column are better.

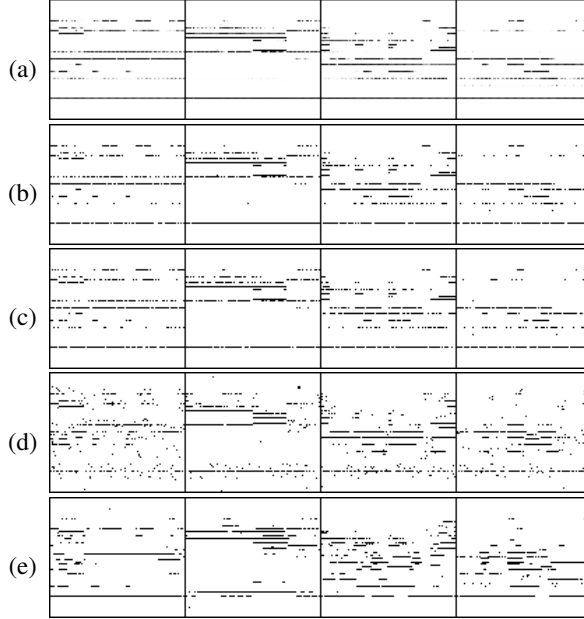


Figure 8. Closeup of the piano track in Figure 7.

4.2 Objective Evaluation Metrics

We generate 800 samples for each model and use the following metrics proposed in [10] for evaluation. We consider a model better if the average metric values of the generated samples are closer to those computed from the training data.

- **Qualified note rate (QN)** computes the ratio of the number of the qualified notes (notes no shorter than three time steps, i.e., a 32th note) to the total number of notes. Low QN implies overly-fragmented music.
- **Polyphonicity (PP)** is defined as the ratio of the number of time steps where more than two pitches are played to the total number of time steps.
- **Tonal distance (TD)** measures the distance between the chroma features (one for each beat) of a pair of tracks in the tonal space proposed in [15]. In what follows, we only report the **TD** between the piano and the guitar, for they are the two most used tracks.

4.3 Comparison of Binarization Strategies

We compare the proposed model with two common test-time binarization strategies: *Bernoulli sampling* (BS) and *hard thresholding* (HT). Some qualitative results are pro-

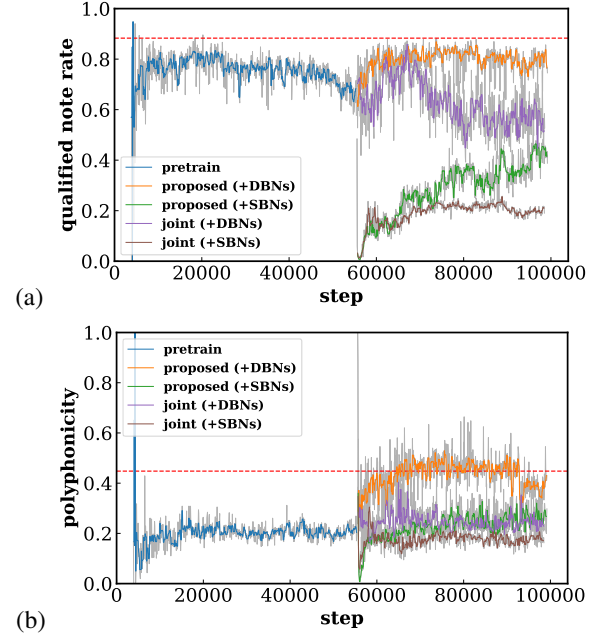


Figure 9. (a) Qualified note rate (QN) and (b) polyphonicity (PP) as a function of training steps for different models. The dashed lines indicate the average QN and PP of the training data, respectively. (Best viewed in color.)

vided in Figures 7 and 8. Moreover, we present in Table 1 a quantitative comparison among them.

Both qualitative and quantitative results show that the two test-time binarization strategies can lead to overly-fragmented piano-rolls (see the ‘pretrained’ ones). The proposed model with DBNs is able to generate piano-rolls with a relatively small number of overly-fragmented notes (a **QN** of 0.78; see Table 1) and to better capture the statistical properties of the training data in terms of **PP**. However, the proposed model with SBNs produces a number of random-noise-like artifacts in the generated piano-rolls, as can be seen in Figure 8(d), leading to a low **QN** of 0.42. We attribute to the stochastic nature of SBNs. Moreover, we can also see from Figure 9 that only the proposed model with DBNs keeps improving after the second-stage training starts in terms of **QN** and **PP**.

4.4 Comparison of Training Strategies

We consider two alternative training strategies:

- **joint:** pretrain G and D in the first stage, and then

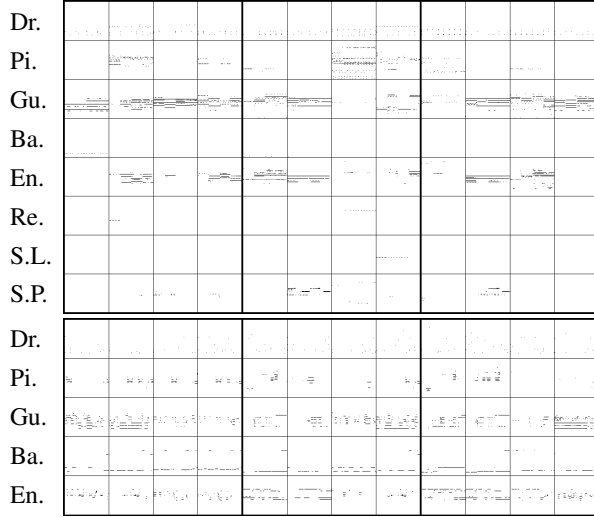


Figure 10. Example generated piano-rolls of the end-to-end models with (top) DBNs and (bottom) SBNs. Empty tracks are not shown.

train G and R (like viewing R as part of G) jointly with D in the second stage.

- **end-to-end:** train G , R and D jointly in one stage.

As shown in Table 1, the models with DBNs trained using the *joint* and *end-to-end* training strategies receive lower scores as compared to the *two-stage* training strategy in terms of **QN** and **PP**. We can also see from Figure 9(a) that the model with DBNs trained using the *joint* training strategy starts to degenerate in terms of **QN** at about 10,000 steps after the second-stage training begins.

Figure 10 shows some qualitative results for the *end-to-end* models. It seems that the models learn the proper pitch ranges for different tracks. We also see some chord-like patterns in the generated piano-rolls. From Table 1 and Figure 10, in the end-to-end training setting SBNs are not inferior to DBNs, unlike the case in the two-stage training. Although the generated results appear preliminary, to our best knowledge this represents the first attempt to generate such high dimensional data with BNs from scratch.

4.5 Effects of the Shared/private and Multi-stream Design of the Discriminator

We compare the proposed model with two ablated versions: the **ablated-I** model, which removes the onset/offset and chroma streams, and the **ablated-II** model, which uses only a shared discriminator without the shared/private and multi-stream design (i.e., the one adopted in [10]).⁷ Note that the comparison is done by applying either BS or HT (not BNs) to the first-stage pretrained models.

As shown in Table 1, the proposed model (see “pre-trained”) outperforms the two ablated versions in all three metrics. A lower **QN** for the proposed model as compared to the ablated-I model suggests that the onset/offset stream can alleviate the overly-fragmented note problem. Lower

⁷ The number of parameters for the proposed, ablated-I and ablated-II models is 3.7M, 3.4M and 4.6M, respectively.

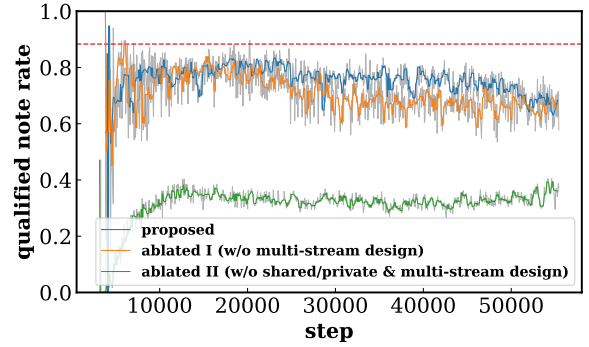


Figure 11. Qualified note rate (QN) as a function of training steps for different models. The dashed line indicates the average QN of the training data. (Best viewed in color.)

	with SBNs	with DBNs
completeness*	0.19	0.81
harmonicity	0.44	0.56
rhythmicity	0.56	0.44
overall rating	0.16	0.84

*We asked, “Are there many overly-fragmented notes?”

Table 2. Result of a user study, averaged over 20 subjects.

TD for the proposed and ablated-I models as compared to the ablated-II model indicates that the shared/private design better capture the intertrack harmonicity. Figure 11 also shows that the proposed and ablated-I models learn faster and better than the ablated-II model in terms of **QN**.

4.6 User Study

Finally, we conduct a user study involving 20 participants recruited from the Internet. In each trial, each subject is asked to compare two pieces of four-bar music generated from scratch by the proposed model using SBNs and DBNs, and vote for the better one in four measures. There are five trials in total per subject. We report in Table 2 the ratio of votes each model receives. The results show a preference to DBNs for the proposed model.

5. DISCUSSION AND CONCLUSION

We have presented a novel convolutional GAN-based model for generating binary-valued piano-rolls by using binary neurons at the output layer of the generator. We trained the model on an eight-track piano-roll dataset. Analysis showed that the generated results of our model with deterministic binary neurons features fewer overly-fragmented notes as compared with existing methods. Though the generated results appear preliminary and lack musicality, we showed the potential of adopting binary neurons in a music generation system.

In future work, we plan to further explore the end-to-end models and add recurrent layers to the temporal model. It might also be interesting to use BNs for music transcription [3], where the desired outputs are also binary-valued.

6. REFERENCES

- [1] Binary stochastic neurons in tensorflow, 2016. Blog post on R2RT blog. [Online] <https://r2rt.com/binary-stochastic-neurons-in-tensorflow.html>.
- [2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proc. ICML*, 2017.
- [3] Emmanouil Benetos, Simon Dixon, Dimitrios Gianneoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.
- [4] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- [5] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In *Proc. ISMIR*, 2011.
- [6] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proc. ICML*, 2012.
- [7] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation: A survey. *arXiv preprint arXiv:1709.01620*, 2017.
- [8] Hang Chu, Raquel Urtasun, and Sanja Fidler. Song from PI: A musically plausible network for pop music generation. In *Proc. ICLR, Workshop Track*, 2017.
- [9] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. In *Proc. ICLR*, 2017.
- [10] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. MuseGAN: Symbolic-domain music generation and accompaniment with multi-track sequential generative adversarial networks. In *Proc. AAAI*, 2018.
- [11] Douglas Eck and Jürgen Schmidhuber. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In *Proc. IEEE Workshop on Neural Networks for Signal Processing*, 2002.
- [12] Ian J. Goodfellow et al. Generative adversarial nets. In *Proc. NIPS*, 2014.
- [13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. In *Proc. NIPS*, 2017.
- [14] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. DeepBach: A steerable model for Bach chorales generation. In *Proc. ICML*, 2017.
- [15] Christopher Harte, Mark Sandler, and Martin Gasser. Detecting harmonic change in musical audio. In *Proc. ACM MM Workshop on Audio and Music Computing Multimedia*, 2006.
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proc. ECCV*, 2016.
- [17] Geoffrey Hinton. Neural networks for machine learning—using noise as a regularizer (lecture 9c), 2012. Coursera, video lectures. [Online] <https://www.coursera.org/lecture/neural-networks/using-noise-as-a-regularizer-7-min-wbw7b>.
- [18] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. In *Proc. ISMIR*, 2017.
- [19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [20] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. Imposing higher-level structure in polyphonic music generation using convolutional restricted Boltzmann machines and constraints. *Journal of Creative Music Systems*, 3(1), 2018.
- [21] Hyungui Lim, Seungyeon Rhyu, and Kyogu Lee. Chord generation from symbolic melody using BLSTM networks. In *Proc. ISMIR*, 2017.
- [22] Olof Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. In *NIPS Workshop on Constructive Machine Learning Workshop*, 2016.
- [23] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016.
- [24] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *Proc. ICML*, 2018.
- [25] Bob L. Sturm, João Felipe Santos, Oded Ben-Tal, and Iryna Korshunova. Music transcription modelling and composition using deep learning. In *Proc. CSMS*, 2016.
- [26] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In *Proc. ISMIR*, 2017.
- [27] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. In *Proc. AAAI*, 2017.

COVER SONG SYNTHESIS BY ANALOGY

Christopher J. Tralie

Duke University Department of Mathematics
ctralie@alumni.princeton.edu

ABSTRACT

In this work, we pose and address the following “cover song analogies” problem: given a song A by artist 1 and a cover song A' of this song by artist 2, and given a different song B by artist 1, synthesize a song B' which is a cover of B in the style of artist 2. Normally, such a polyphonic style transfer problem would be quite challenging, but we show how the cover songs example constrains the problem, making it easier to solve. First, we extract the longest common beat-synchronous subsequence between A and A' , and we time stretch the corresponding beat intervals in A' so that they align with A . We then derive a version of joint 2D convolutional NMF, which we apply to the constant-Q spectrograms of the synchronized segments to learn a translation dictionary of sound templates from A to A' . Finally, we apply the learned templates as filters to the song B , and we mash up the translated filtered components into the synthesized song B' using audio mosaicing. We showcase our algorithm on several examples, including a synthesized cover version of Michael Jackson’s “Bad” by Alien Ant Farm, learned from the latter’s “Smooth Criminal” cover.

1. INTRODUCTION

The rock group Alien Ant Farm has a famous cover of Michael Jackson’s “Smooth Criminal” which is faithful to but stylistically unique from the original song. However, to our knowledge, they never released a cover of any other Michael Jackson songs. What if we instead wanted to know how they would have covered Michael Jackson’s “Bad”? That is, we seek a song which is identifiable as MJ’s “Bad,” but which also sounds as if it’s in Alien Ant Farm’s *style*, including timbral characteristics, relative tempo, and instrument types.

In general, multimedia style transfer is a challenging task in computer aided creativity applications. When an example of the stylistic transformation is available, as in the “Smooth Criminal” example above, this problem can be phrased in the language of analogies; given an object A and a differently stylized version of this object A' , and given an object B in the style of A , synthesize an ob-

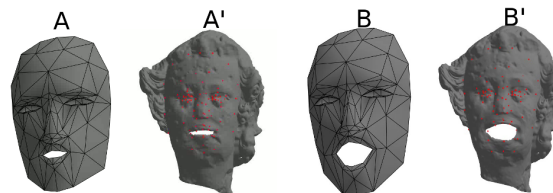


Figure 1. A demonstration of the “shape analogies” technique in [22]. In the language of our work, the statue head with a neutral expression (A') is a “cover” of the low resolution mesh A with a neutral expression, and this is used to synthesize the surprised statue “cover” face B' from a low resolution surprised face mesh B .

ject B' which has the properties of B but the style of A' . One of the earliest works using this vocabulary is the “image analogies” work [12], which showed it was possible to transfer both linear filters (e.g. blurring, embossing) and nonlinear filters (e.g. watercolors) in the same simple framework. More recent work with convolutional networks has shown even better results for images [10]. There has also been some work on “shape analogies” for 3D meshes [22], in which nonrigid deformations between triangle meshes A and A' are used to induce a corresponding deformation B' from an object B , which can be used for motion transfer (Figure 1).

In the audio realm, most style transfer works are based on mashing up sounds from examples in a target style using “audio mosaicing,” usually after manually specifying some desired path through a space of sound grains [16]. A more automated audio mosaicing technique, known as “audio analogies” [19], uses correspondences between a MIDI score and audio to drive concatenated synthesis, which leads to impressive results on monophonic audio, such as stylized synthesis of jazz recordings of a trumpet. More recently, this has evolved into the audio to musical audio setting with audio “mosaicing,” in which the timbre of an audio source is transferred onto a target by means of a modified NMF algorithm [7], such as bees buzzing The Beatles’ “Let It Be.” A slightly closer step to the polyphonic (multi source) musical audio to musical audio case has been shown to work for drum audio cross-synthesis with the aid of a musical score [5], and some very recent initial work has extended this to the general musical audio to musical audio case [9] using 2D nonnegative matrix factorization, though this still remains open. Finally, there is some recent work on converting polyphonic audio of guitar songs to musical scores of varying difficulties so users



© Christopher J. Tralie. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Christopher J. Tralie. “Cover Song Synthesis by Analogy”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

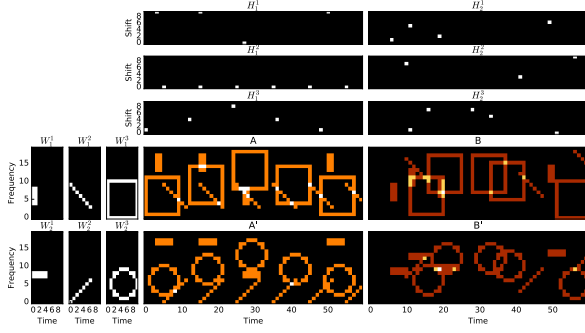


Figure 2. An ideal cartoon example of our joint 2DNMF and filtering process, where $M = 20$, $N = 60$, $T = 10$, $F = 10$, and $K = 3$. In this example, vertical time-frequency blocks are “covered” by horizontal blocks, diagonal lines with negative slopes are covered by diagonal lines with positive slopes, and squares are covered by circles. When presented with a new song B , our goal is to synthesize a song B' whose CQT is shown in the lower right green box.

can play their own covers [1].

In this work, we constrain the polyphonic musical audio to musical audio style transfer problem by using *cover song* pairs, or songs which are the same but in a different style, and which act as our ground truth A and A' examples in the analogies framework. Since small scale automatic cover song identification has matured in recent years [4, 17, 18, 23], we can accurately synchronize A and A' (Section 2.1), even if they are in very different styles. Once they are synchronized, the problem becomes more straightforward, as we can blindly factorize A and A' into different instruments which are in correspondence, turning the problem into a series of monophonic style transfer problems. To do this, we perform NMF2D factorization of A and A' (Section 2.2). We then filter B by the learned NMF templates and mash up audio grains to create B' , using the aforementioned “musaicing” techniques [7] (Section 2.3). We demonstrate our techniques on snippets of A , A' , and B which are about 20 seconds long, and we show qualitatively how the instruments of A' transfer onto the music of B in the final result B' (Section 3).

2. ALGORITHM DETAILS

In this section, we will describe the steps of our algorithm in more detail.¹

2.1 Cover Song Alignment And Synchronization

As with the original image analogies algorithm [12], we find it helpful if A and A' are in direct correspondence at the sample level. Since cover song pairs generally differ in tempo, we need to align them first. To accomplish this, we draw upon the state of the art “early fusion” cover song alignment technique presented by the authors of [23]. Briefly, we extract beat onsets for A and A' using either

¹ Note that all audio is mono and sampled at 22050hz.

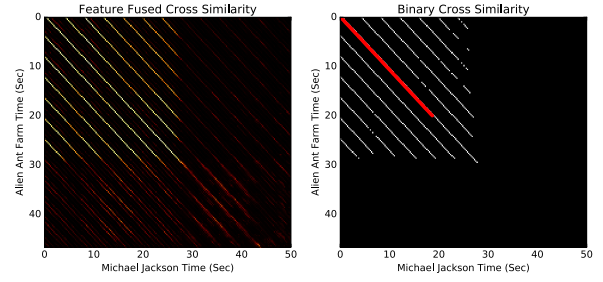


Figure 3. An example feature fused cross-similarity matrix D for the first 80 beats of Michael Jackson’s “Smooth Criminal,” compared to the cover version by Alien Ant Farm. We threshold the matrix and extract 20 seconds of the longest common subsequence, as measured by Smith Waterman. The alignment path is shown in red.

a simple dynamic programming beat tracker [8] or slower but more accurate RNN + Bayesian beat trackers [13], depending on the complexity of the audio. We then compute beat-synchronous sliding window HPCP and MFCC features, and we fuse them using similarity network fusion [25, 26]. The result is a $M \times N$ cross-similarity matrix D , where M is the number of beats in A and N is the number of beats in A' , and D_{ij} is directly proportional to the similarity between beat i of A and beat j in A' . Please refer to [23] for more details.

Once we have the matrix D , we can then extract an alignment between A and A' by performing Smith Waterman [20] on a binary thresholded version of D , as in [23]. We make one crucial modification, however. To allow for more permissive alignments with missing beats for identification purposes, the original cover songs algorithm creates a binary thresholded version of D using 10% mutual binary nearest neighbors. On the other hand, in this application, we seek shorter snippets from each song which are as well aligned as possible. Therefore, we create a stricter binary thresholded version B , where $B_{ij} = 1$ only if it is in the top $3\sqrt{MN}$ distances over all MN distances in D . This means that many rows of B_{ij} will be all zeros, but we will hone in on the best matching segments. Figure 3 shows such a thresholding of the cross-similarity matrix for two versions of the song “Smooth Criminal,” which is an example we will use throughout this section. Once B has been computed, we compute a X -length alignment path P by back-tracing through the Smith Waterman alignment matrix, as shown in Figure 3.

Let the beat onset times for A in the path P be t_1, t_2, \dots, t_X and the beat times for A' be s_1, s_2, \dots, s_X . We use the rubberband library [3] to time stretch A' beat by beat, so that interval $[s_i, s_{i+1}]$ is stretched by a factor $(t_{i+1} - t_i)/(s_{i+1} - s_i)$. The result is a snippet of A' which is the same length as the corresponding snippet in A . Henceforth, we will abuse notation and refer to these snippets as A and A' . We also extract a smaller snippet from B of the same length for reasons of memory efficiency, which we will henceforth refer to as B .

2.2 NMF2D for Joint Blind Factorization / Filtering

Once we have synchronized the snippets A and A' , we blindly factorize and filter them into K corresponding tracks A_1, A_2, \dots, A_K and A'_1, A'_2, \dots, A'_K . The goal is for each track to contain a different instrument. For instance, A_1 may contain an acoustic guitar, which is covered by an electric guitar contained in A'_1 , and A_2 may contain drums which are covered by a drum machine in A'_2 . This will make it possible to reduce the synthesis problem to K independent monophonic analogy problems in Section 2.3.

To accomplish this, the main tool we use is 2D convolutional nonnegative matrix factorization (2DNMF) [15]. We apply this algorithm to the magnitude of the constant-Q transforms (CQTs) [2] \mathbf{C}_A , $\mathbf{C}_{A'}$ and \mathbf{C}_B of A , A' , and B , respectively. Intuitively, we seek K time-frequency templates in A and A' which represent small, characteristic snippets of the K instruments we believe to be present in the audio. We then approximate the magnitude constant-Q transform of A and A' by convolving these templates in both time and frequency. The constant-Q transform is necessary in this framework, since a pitch shift can be approximated by a *linear* shift of all CQT bins by the same amount. Frequency shifts can then be represented as convolutions in the vertical direction, which is not possible with the ordinary STFT. Though it is more complicated, 2DNMF is a more compact representation than 1D convolutional NMF (in time only), in which it is necessary to store a different template for each pitch shift of each instrument. We note that pitch shifts in real instruments are more complicated than shifting all frequency bins by the same perceptual amount [14], but the basic version of 2DNMF is fine for our purposes.

More concretely, define a K -component, F -frequency, T -time lag 2D convolutional NMF decomposition for a matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ as follows

$$\mathbf{X} \approx \mathbf{\Lambda}_{\mathbf{W}, \mathbf{H}} = \sum_{\tau=1}^T \sum_{\phi=1}^F \mathbf{W}^\tau \mathbf{H}^\phi \quad (1)$$

where $\mathbf{W}^\tau \in \mathbb{R}^{M \times K}$ and $\mathbf{H}^\phi \in \mathbb{R}^{K \times N}$ store a τ -shifted time template and ϕ -frequency shifted coefficients, respectively. By \downarrow_ϕ , we mean down shift the rows of A by ϕ , so that row i of $\downarrow_\phi A$ is row $i - \phi$ of A , and the first ϕ rows of $\downarrow_\phi A$ are all zeros. And \leftarrow_τ means left-shift A , so that column j of $\leftarrow_\tau A$ is column $j - \tau$ of A , and the first τ columns of $\leftarrow_\tau A$ are all zeros.

In our problem, we define an extension of 2DNMF to *jointly* factorize the 2 songs, A and A' , which each have M CQT coefficients. In particular, given matrices $\mathbf{C}_A, \mathbf{C}_{A'} \in \mathbb{C}^{M \times N_1}$ representing the complex CQT frames in each song over time, we seek $\mathbf{W}_1^\tau, \mathbf{W}_2^\tau \in \mathbb{R}^{M \times K}$ and $\mathbf{H}_1^\phi \in \mathbb{R}^{K \times N_1}$ that minimize the sum of the Kullback-Leibler divergences between the magnitude CQT coefficients and the convolutions:

$$D(|\mathbf{C}_A| \parallel \mathbf{\Lambda}_{\mathbf{W}_1, \mathbf{H}_1}) + D(|\mathbf{C}_{A'}| \parallel \mathbf{\Lambda}_{\mathbf{W}_2, \mathbf{H}_1}) \quad (2)$$

where the Kullback-Leibler divergence $D(X \parallel Y)$ is defined as

$$D(\mathbf{X} \parallel \mathbf{Y}) = \sum_i \sum_j \mathbf{X}_{i,j} \log \frac{\mathbf{X}_{i,j}}{\mathbf{Y}_{i,j}} - \mathbf{X}_{i,j} + \mathbf{Y}_{i,j} \quad (3)$$

That is, we *share* \mathbf{H}_1 between the factorizations of $|\mathbf{C}_A|$ and $|\mathbf{C}_{A'}|$ so that we can discover shared structure between the covers. Following similar computations to those of ordinary 2DNMF [15], it can be shown that Equation 2 is non-decreasing under the alternating update rules:

$$\mathbf{W}_1^\tau \leftarrow \mathbf{W}_1^\tau \odot \frac{\sum_{\phi=1}^F \left(\frac{\uparrow_\phi |\mathbf{C}_A|}{\mathbf{\Lambda}_{\mathbf{W}_1, \mathbf{H}_1}} \right) \mathbf{H}_1^\phi}{\sum_{\phi=1}^F \mathbf{1} \cdot \mathbf{H}_1^\phi} \quad (4)$$

$$\mathbf{W}_2^\tau \leftarrow \mathbf{W}_2^\tau \odot \frac{\sum_{\phi=1}^F \left(\frac{\uparrow_\phi |\mathbf{C}_{A'}|}{\mathbf{\Lambda}_{\mathbf{W}_2, \mathbf{H}_1}} \right) \mathbf{H}_1^\phi}{\sum_{\phi=1}^F \mathbf{1} \cdot \mathbf{H}_1^\phi} \quad (5)$$

$$\mathbf{H}_1^\phi \leftarrow \mathbf{H}_1^\phi \odot \left(\frac{\sum_{\tau=1}^T \mathbf{W}_1^\tau \left(\frac{\leftarrow_\tau |\mathbf{C}_A|}{\mathbf{\Lambda}_{\mathbf{W}_1, \mathbf{H}_1}} \right) + \mathbf{W}_2^\tau \left(\frac{\leftarrow_\tau |\mathbf{C}_{A'}|}{\mathbf{\Lambda}_{\mathbf{W}_2, \mathbf{H}_1}} \right)}{\sum_{\tau=1}^T \mathbf{W}_1^\tau \mathbf{1} + \mathbf{W}_2^\tau \mathbf{1}} \right) \quad (6)$$

where $\mathbf{1}$ is a column vector of all 1s of appropriate dimension. We need an invertible CQT to go back to audio templates, so we use the non-stationary Gabor Transform (NSGT) implementation of the CQT [24] to compute \mathbf{C}_A , $\mathbf{C}_{A'}$, and \mathbf{C}_B . We use 24 bins per octave between 50hz and 11.7kHz, for a total of 189 CQT bins. We also use $F = 14$ in most of our examples, allowing 7 halfstep shifts, and we use $T = 20$ on temporally downsampled CQTs to cover a timespan of 130 milliseconds. Finally, we iterate through Equations 4, 5, and 6 in sequence 300 times.

Note that a naive implementation of the above equations can be very computationally intensive. To ameliorate this, we implemented GPU versions of Equations 1, 4, 5, and 6. Equation 1 in particular is well-suited for a parallel implementation, as the shifted convolutional blocks overlap each other heavily and can be carefully offloaded into shared memory to exploit this². In practice, we witnessed a 30x speedup of our GPU implementation over our CPU implementation for 20 second audio clips for A , A' , and B .

Figure 2 shows a synthetic example with an exact solution, and Figure 4 shows a local min which is the result of running Equations 4, 5, and 6 on real audio from the ‘‘Smooth Criminal’’ example. It is evident from \mathbf{H}_1 that the first component is percussive (activations at regular intervals in \mathbf{H}_1^1 , and no pitch shifts), while the second component corresponds to the guitar melody (\mathbf{H}_1^2 appears like a ‘‘musical score’’ of sorts). Furthermore, \mathbf{W}_1^1 and \mathbf{W}_2^1

² In the interest of space, we omit more details of our GPU-based NMFD in this paper, but a documented implementation can be found at <https://github.com/ctralie/CoverSongSynthesis/>, and we plan to release more details in a companion paper later.

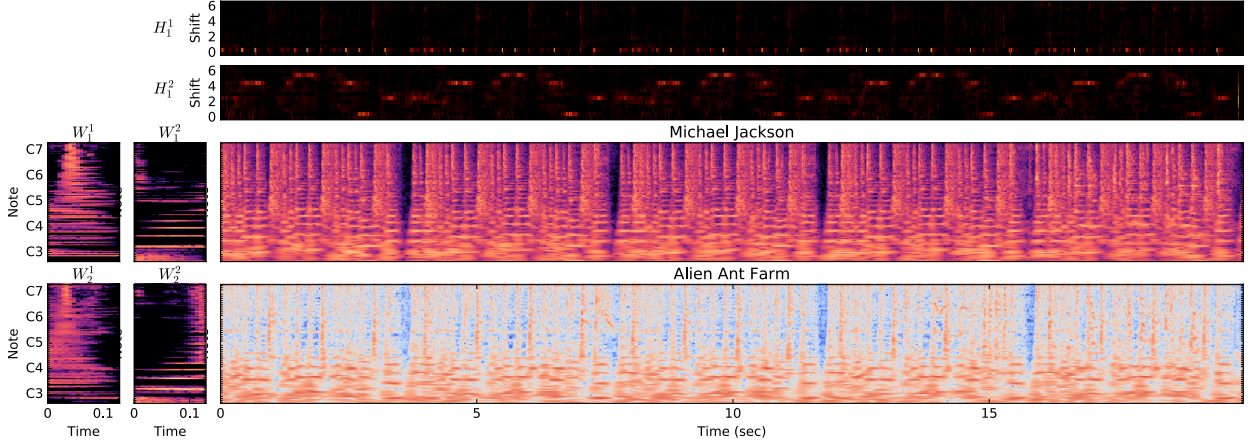


Figure 4. Joint 2DNMF on the magnitude CQTs of “Smooth Criminal” by Michael Jackson and Alien Ant Farm, with $F = 14$ (7 halfsteps at 24 bins per octave), $T = 20$ (130ms in time), and $K = 2$ components. In this case, W_1^1 and W_2^1 hold percussive components, and W_1^2 and W_2^2 hold guitar components.

have broadband frequency content consistent with percussive events, while W_1^2 and W_2^2 have visible harmonics consistent with vibrating strings. Note that we generally use more than $K = 2$, which allows finer granularity than harmonic/percussive, but even for $K = 2$ in this example, we observe qualitatively better separation than off-the-shelf harmonic/percussive separation algorithms [6].

Once we have W_1 , W_2 , and H_1 , we can recover the audio templates A_1, A_2, \dots, A_K and A'_1, A'_2, \dots, A'_K by using the components of W_1 and W_2 as filters. First, define $\Lambda_{W,H,k}$ as

$$\Lambda_{W,H,k} = \sum_{\tau=1}^T \sum_{\phi=1}^F \mathbf{W}^k \downarrow_{\tau}^{\phi} \mathbf{H}_k \rightarrow_{\tau}^{\phi} \quad (7)$$

where \mathbf{W}^k is the k^{th} column of \mathbf{W} and \mathbf{H}_k is the k^{th} row of \mathbf{H} . Now, define the filtered CQTs by using soft masks derived from W_1 and H_1 :

$$\mathbf{C}_{A_k} = \mathbf{C}_A \odot \left(\frac{\Lambda_{W_1,H_1,k}^p}{\sum_{m=1}^K \Lambda_{W_1,H_1,m}^p} \right) \quad (8)$$

$$\mathbf{C}_{A'_k} = \mathbf{C}_{A'} \odot \left(\frac{\Lambda_{W_2,H_1,k}^p}{\sum_{m=1}^K \Lambda_{W_2,H_1,m}^p} \right) \quad (9)$$

where p is some positive integer applied element-wise (we choose $p = 2$), and the above multiplications and divisions are also applied element-wise. It is now possible to invert the CQTs to uncover the audio templates, using the inverse NSGT [24]. Thus, if the separation was good, we are left with K independent monophonic pairs of sounds between A and A' .

In addition to inverting the sounds after these masks are applied, we can also listen to the components of W_1 and W_2 themselves to gain insight into how they are behaving as filters. Since W_1 and W_2 are magnitude only, we apply the Griffin Lim algorithm [11] to perform phase retrieval, and then we invert them as before to obtain 130 millisecond sounds for each k .

2.3 Mosaicing And Mixing

We now describe how to use the corresponding audio templates we learned in Section 2.2 to perform style transfer on a new piece of audio, B .

2.3.1 Separating Tracks in B

First, we compute the CQT of B , $\mathbf{C}_B \in \mathbb{C}^{M \times N_2}$. We then represent its magnitude using W_1 as a basis, so that we filter B into the same set of instruments into which A was separated. That is, we solve for \mathbf{H}_2 so that $|\mathbf{C}_B| \approx \Lambda_{W_1,H_2}$. This can be performed with ordinary 2DNMF, holding W_1 fixed; that is, repeating the following update until convergence

$$\mathbf{H}_2^{\phi} \leftarrow \mathbf{H}_2^{\phi} \odot \left(\frac{\sum_{\tau=1}^T \mathbf{W}_1^{\tau} \downarrow_{\tau}^{\phi} \left(\frac{|\mathbf{C}_B|}{\Lambda_{W_1,H_2}} \right)}{\sum_{\tau=1}^T \mathbf{W}_1^{\tau} \downarrow_{\tau}^{\phi} \mathbf{1}} \right) \quad (10)$$

As with A and A' , we can now filter B into a set of audio tracks B_1, B_2, \dots, B_K by first computing filtered CQTs as follows

$$\mathbf{C}_{B_k} = \mathbf{C}_B \odot \left(\frac{\Lambda_{W_1,H_2,k}^p}{\sum_{m=1}^K \Lambda_{W_1,H_2,m}^p} \right) \quad (11)$$

and then inverting them.

2.3.2 Constructing B' Track by Track

At this point, we could use \mathbf{H}_2 and let our cover song CQT magnitudes $|\mathbf{C}_{B'}| = \Lambda_{W_2,H_2}$, followed by Griffin Lim to recover the phase. However, we have found that the resulting sounds are too “blurry,” as they lack all but rearranged low rank detail from A' . Instead, we choose to draw sound grains from the inverted, filtered tracks from A' , which contain all of the detail of the original song. For this, we first reconstruct each track of B using audio grains from the corresponding tracks in A , and then we

replace each track with the corresponding track in B . To accomplish this, we apply the audio musaicing technique of Driedger [7] to each track in B , using source audio A .

For computational and memory reasons, we now abandon the CQT and switch to the STFT with hop size $h = 256$ and window size $w = 2048$. More specifically, let N_1 be the number of STFT frames in A and A' and let N_2 be the number of STFT frames in B . For each A_k , we create an STFT matrix \mathbf{S}_{A_k} which holds the STFT of A_k concatenated to pitch shifted versions of A_k , so that pitches beyond what were in A can be represented, but by the same instruments. We use ± 6 halfsteps, so $\mathbf{S}_{A_k} \in \mathbb{C}^{w \times 13N_1}$. We do the same for A'_k to create $\mathbf{S}_{A'_k} \in \mathbb{C}^{w \times 13N_1}$. Finally, we compute the STFT of B_k without any shifts: $\mathbf{S}_{B_k} \in \mathbb{C}^{w \times N_2}$.

Now, we apply Driedger's technique, using $|\mathbf{S}_{A_k}|$ as a spectral dictionary to reconstruct $|\mathbf{S}_{B_k}|$ (\mathbf{S}_{A_k} is analogous to the buzzing bees in [7]). That is, we seek an $\mathbf{H}_k \in \mathbb{R}^{13N_1 \times N_2}$ so that

$$|\mathbf{S}_{B_k}| \approx |\mathbf{S}_{A_k}| \mathbf{H} \quad (12)$$

For completeness, we briefly re-state the iterations in Driedger's technique [7]. For L iterations total, at the ℓ^{th} iteration, compute the following 4 updates in order. First, we restrict the number of repeated activations by filtering in a maximum horizontal neighborhood in time

$$\mathbf{R}_{\mathbf{km}}^{(\ell)} = \left\{ \begin{array}{ll} \mathbf{H}_{\mathbf{km}}^{(\ell)} & \text{if } \mathbf{H}_{\mathbf{km}}^{(\ell)} = \mu_{\mathbf{km}}^{r,(\ell)} \\ \mathbf{H}_{\mathbf{km}}^{(\ell)} (1 - \frac{n+1}{N}) & \text{otherwise} \end{array} \right\} \quad (13)$$

where $\mu_{\mathbf{km}}^{r,(\ell)}$ holds the maximum in a neighborhood $\mathbf{H}_{\mathbf{k}, \mathbf{m}-r:\mathbf{m}+r}$ for some parameter r (we choose $r = 3$ in our examples). Next, we restrict the number of simultaneous activations by shrinking all of the values in each column that are less than the top p values in that column:

$$\mathbf{P}_{\mathbf{km}}^{(\ell)} = \left\{ \begin{array}{ll} \mathbf{R}_{\mathbf{km}}^{(\ell)} & \text{if } \mathbf{R}_{\mathbf{km}}^{(\ell)} \geq \mathbf{MP}^{(n)} \\ \mathbf{R}_{\mathbf{km}}^{(\ell)} (1 - \frac{n+1}{N}) & \text{otherwise} \end{array} \right\} \quad (14)$$

where $\mathbf{MP}^{(\ell)}$ is a row vector holding the p^{th} largest value of each column of $\mathbf{R}^{(\ell)}$ (we choose $p = 10$ in our examples). After this, we promote time-continuous structures by convolving along all of the diagonals of \mathbf{H} :

$$\mathbf{C}_{\mathbf{km}}^{(\ell)} = \sum_{i=-c}^c \mathbf{P}_{(\mathbf{k}+\mathbf{i}),(\mathbf{m}+\mathbf{i})}^{(\ell)} \quad (15)$$

We choose $c = 3$ in our examples. Finally, we perform the ordinary KL-based NMF update:

$$\mathbf{H}^{(\ell+1)} \leftarrow \mathbf{H}^{(\ell)} \odot \frac{|\mathbf{S}_{A_k}|^T |\mathbf{S}_{B_k}|}{|\mathbf{S}_{A_k}| \mathbf{C}^{(\ell)}} \quad (16)$$

We perform 100 such iterations ($L = 100$). Once we have our final \mathbf{H} , we can use this to create B'_k as follows:

$$\mathbf{S}_{B'_k} = \mathbf{S}_{A'_k} \mathbf{H} \quad (17)$$

In other words, we use the learned activations to create \mathbf{S}_{B_k} using \mathbf{S}_{A_k} , but we instead use these activations with the dictionary from $\mathbf{S}_{A'_k}$. *This is the key step in the style transfer*, and it is done for the same reason that \mathbf{H}_1 is shared between A and A' . Figure 5 shows an example for the guitar track on Michael Jackson's "Bad," translating to Alien Ant Farm using A and A' templates from Michael Jackson's and Alien Ant Farms' versions of "Smooth Criminal," respectively. It is visually apparent that $|\mathbf{S}_{A_k}| \mathbf{H} \approx \mathbf{S}_{B_k}$, it is also apparent that $\mathbf{S}_{B'_k} = \mathbf{S}_{A'_k} \mathbf{H}$ is similar to \mathbf{S}_{B_k} , except it has more energy in the higher frequencies. This is consistent with the fact that Alien Ant Farm uses a more distorted electric guitar, which has more broadband energy.

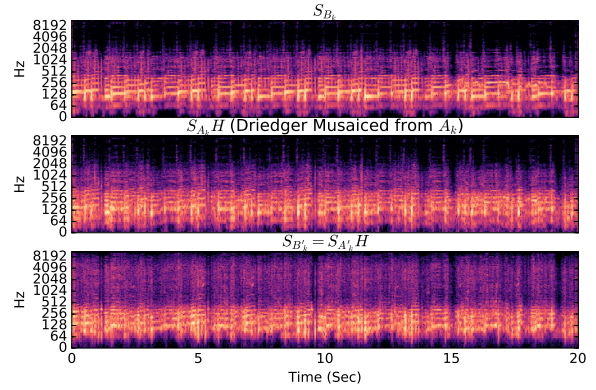


Figure 5. Driedger's technique [7] using audio grains from pitch shifted versions of A_k (the k^{th} track in Michael Jackson's "Smooth Criminal") to create B_k (the k^{th} track in Michael Jackson's "Bad"), and using the activations to create B'_k (the k^{th} synthesized track in Alien Ant Farm's "Bad").

To create our final B' , we simply add all of the STFTs $\mathbf{S}_{B'_k}$ together for each k , and we perform and inverse STFT to go back to audio.

2.4 A Note About Tempos

The algorithm we have described so far assumes that the tempos t_A , $t_{A'}$, and t_B of A , A' , and B , respectively are similar. This is certainly not true in more interesting covers. Section 2.1 took care of the disparity between A and A' during the synchronization. However, we also need to perform a tempo scaling on B by a factor of t_A/t_B before running our algorithm. Once we have computed B' , whose tempo is initially t_A , we scale its tempo back by $(t_B/t_A) \cdot (t_{A'}/t_A)$. For instance, suppose that $t_A = 60$, $t_{A'} = 80$, and $t_B = 120$. Then the final tempo of B' will be $60 \times (120/60) \times (80/60) = 160$ bpm.

3. EXPERIMENTAL EXAMPLES

We now qualitatively explore our technique on several examples³. In all of our examples, we use $K = 3$ sources.

³ Please listen to our results at <http://www.covers1000.net/analogsies.html>

This is a hyperparameter that should be chosen with care in general, since we would like each component to correspond to a single instrument or group of related instruments. However, our initial examples are simple enough that we expect basically a harmonic source, a percussive source, and one other source or sub-separation between either harmonic and percussive.

First, we follow through on the example we have been exploring and we synthesize Alien Ant Farm’s “Bad” from Michael Jackson’s “Bad” (B), using “Smooth Criminal” as an example. The translation of the guitar from synth to electric is clearly audible in the final result. Furthermore, a track which was exclusively drums in A included some extra screams in A' that Alien Ant Farm performed as embellishments. These embellishments transferred over to B' in the “Bad” example, further reinforcing Alien Ant Farm’s style. Note that these screams would not have been preserved had we simply used inverted the CQT Λ_{W_2, H_2} , but they are present in one of the filtered tracks and available as audio grains during mosaicing for that track.

In addition to the “Bad” example, we also synthesize Alien Ant Farm’s version of “Wanna Be Startin Something,” using “Smooth Criminal” as an example for A and A' once again. In this example, Alien Ant Farm’s screams occur consistently with the fast drum beat every measure.

Finally, we explore an example with a more extreme tempo shift between A and A' ($t_{A'} < t_A$). We use Marilyn Manson’s cover of “Sweet Dreams” by the Eurythmics to synthesize a Marilyn Manson cover of “Who’s That Girl” by the Eurythmics. We found that in this particular example, we obtained better results when we performed 2DNMF on $|C_A|$ by itself first, and then we performed the optimizations in Equation 5 and Equation 6, holding W_1 fixed. This is a minor tweak that can be left to the discretion of the user at runtime.

Overall, our technique works well for instrument translation in these three examples. However, we did notice that the vocals did not carry over at all in any of them. This is to be expected, since singing voice separation often assumes that the instruments are low rank and the voice is high rank [21], and our filters and final mosaicing are both derived from low rank NMF models.

4. DISCUSSION / FUTURE DIRECTIONS

In this work, we demonstrated a proof of concept, fully automated end to end system which can synthesize a cover song snippet of a song B given an example cover A and A' , where A is by the same artist as B , and B' should sound like B but in the style of A' . We showed some promising initial results on a few examples, which is, to our knowledge, one of the first steps in the challenging direction of automatic polyphonic audio mosaicing.

Our technique does have some limitations, however. Since we use W_1 for both A and B , we are limited to examples in which A and B have similar instruments. It would be interesting to explore how far one could push this technique with different instruments between A and B , which happens quite often even within a corpus by the

same artist.

We have also noticed that in addition to singing voice, other “high rank” instruments, such as the fiddle, cannot be properly translated. We believe that that translating such instruments and voices would be an interesting and challenging future direction of research, and it would likely need a completely different approach to the one we presented here.

Finally, out of the three main steps of our pipeline, synchronization (Section 2.1), blind joint factorization/source separation (Section 2.2), and filtering/mosaicing (Section 2.3), the weakest step by far is the blind source separation. The single channel source separation problem is still far from solved in general even without the complication of cover songs, so that will likely remain the weakest step for some time. If one has access to the unmixed studio tracks for A , A' , and B , though, that step can be entirely circumvented; the algorithm would remain the same, and one would expect higher quality results. Unfortunately, such tracks are difficult to find in general for those who do not work in a music studio, which is why blind source separation also remains an important problem in its own right.

5. ACKNOWLEDGEMENTS

Christopher Tralie was partially supported by an NSF big data grant DKA-1447491 and an NSF Research Training Grant NSF-DMS 1045133. We also thank Brian McFee for helpful discussions about invertible CQTs.

6. REFERENCES

- [1] Shunya Ariga, Satoru Fukayama, and Masataka Goto. Song2guitar: A difficulty-aware arrangement system for generating guitar solo covers from polyphonic audio of popular music. In *18th International Society for Music Information Retrieval (ISMIR)*, 2017.
- [2] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.
- [3] C Cannam. Rubber band library. *Software released under GNU General Public License (version 1.8. 1)*, 2012.
- [4] Ning Chen, Wei Li, and Haidong Xiao. Fusing similarity functions for cover song identification. *Multimedia Tools and Applications*, pages 1–24, 2017.
- [5] Christian Dittmar and Meinard Müller. Reverse Engineering the Amen Break – Score-informed Separation and Restoration applied to Drum Recordings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(9):1531–1543, 2016.
- [6] Jonathan Driedger, Meinard Müller, and Sascha Disch. Extending harmonic-percussive separation of audio signals. In *ISMIR*, pages 611–616, 2014.

- [7] Jonathan Driedger, Thomas Prätzlich, and Meinard Müller. Let it bee-towards nmf-inspired audio mosaicing. In *ISMIR*, pages 350–356, 2015.
- [8] Daniel PW Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.
- [9] Hadrien Foroughmand and Geoffroy Peeters. Multi-source musaicing using non-negative matrix factor 2-d deconvolution. In *18th International Society for Music Information Retrieval (ISMIR), Late Breaking Session*, 2017.
- [10] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.
- [11] Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.
- [12] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.
- [13] Florian Krebs, Sebastian Böck, and Gerhard Widmer. An efficient state-space model for joint tempo and meter tracking. In *ISMIR*, pages 72–78, 2015.
- [14] Tomohiko Nakamura and Hirokazu Kameoka. Shifted and convolutive source-filter non-negative matrix factorization for monaural audio source separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 489–493. IEEE, 2016.
- [15] Mikkel N Schmidt and Morten Mørup. Nonnegative matrix factor 2-d deconvolution for blind single channel source separation. In *International Conference on Independent Component Analysis and Signal Separation*, pages 700–707. Springer, 2006.
- [16] Diemo Schwarz, Roland Cahen, and Sam Britton. Principles and applications of interactive corpus-based concatenative synthesis. In *Journées d’Informatique Musicale (JIM)*, pages 1–1, 2008.
- [17] Joan Serra, Xavier Serra, and Ralph G Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11(9):093017, 2009.
- [18] Diego F Silva, Chin-Chin M Yeh, Gustavo Enrique de Almeida Prado Alves Batista, Eamonn Keogh, et al. Simple: assessing music similarity using subsequences joins. In *International Society for Music Information Retrieval Conference, XVII*. International Society for Music Information Retrieval-ISMIR, 2016.
- [19] Ian Simon, Sumit Basu, David Salesin, and Maneesh Agrawala. Audio analogies: Creating new music from an existing performance by concatenative synthesis. In *ICMC*. Citeseer, 2005.
- [20] Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.
- [21] Pablo Sprechmann, Alexander M Bronstein, and Guillermo Sapiro. Real-time online singing voice separation from monaural recordings using robust low-rank modeling. In *ISMIR*, pages 67–72, 2012.
- [22] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 399–405. ACM, 2004.
- [23] Christopher J Tralie. Mfcc and hpcp fusion for robust cover song identification. In *18th International Society for Music Information Retrieval (ISMIR)*, 2017.
- [24] Gino Angelo Velasco, Nicki Holighaus, Monika Dörfler, and Thomas Grill. Constructing an invertible constant-q transform with non-stationary gabor frames. *Proceedings of DAFX11, Paris*, pages 93–99, 2011.
- [25] Bo Wang, Jiayan Jiang, Wei Wang, Zhi-Hua Zhou, and Zhuowen Tu. Unsupervised metric fusion by cross diffusion. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2997–3004. IEEE, 2012.
- [26] Bo Wang, Aziz M Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains, and Anna Goldenberg. Similarity network fusion for aggregating data types on a genomic scale. *Nature methods*, 11(3):333–337, 2014.

PART-INVARIANT MODEL FOR MUSIC GENERATION AND HARMONIZATION

Yujia Yan^b, Ethan Lustig^a, Joseph VanderStel^a, Zhiyao Duan^b

Electrical and Computer Engineering^b and Eastman School of Music^a, University of Rochester
{yujia.yan, j.vanderstel, zhiyao.duan}@rochester.edu
ethan.s.lustig@gmail.com

ABSTRACT

Automatic music generation has been gaining more attention in recent years. Existing approaches, however, are mostly ad hoc to specific rhythmic structures or instrumentation layouts, and lack music-theoretic rigor in their evaluations. In this paper, we present a neural language (music) model that tries to model symbolic multi-part music. Our model is part-invariant, i.e., it can process/generate any part (voice) of a music score consisting of an arbitrary number of parts, using a single trained model. For better incorporating structural information of pitch spaces, we use a structured embedding matrix to encode multiple aspects of a pitch into a vector representation. The generation is performed by Gibbs Sampling. Meanwhile, our model directly generates note spellings to make outputs human-readable. We performed objective (grading) and subjective (listening) evaluations by recruiting music theorists to compare the outputs of our algorithm with those of music students on the task of bassline harmonization (a traditional pedagogical task). Our experiment shows that errors of our algorithm and students are differently distributed, and the range of ratings for generated pieces overlaps with students' to varying extents for our three provided basslines. This experiment suggests some future research directions.

1. INTRODUCTION

In recent years, there has been a growing interest in automatic music composition. Automatic music composition is a challenging problem, and it remains an open research topic regardless of many overblown statements in the press since the early days of artificial intelligence.

Apart from purely rule-based models that are difficult to craft, log-linear models, e.g., Hidden Markov Models (HMM), Conditional Random Fields (CRF), and Probabilistic Context-Free Grammars (PCFG) form a set of traditional methods for sequence modeling involving discrete

variables (e.g., [13] [19] [18] [20]). When used for modeling music, they typically model each aspect of music (e.g., melody, harmony, durations) separately, or condition one variable on a small set of other variables (e.g., [1]). This is because, in music, when multiple aspects join together, the number of resulting combinations is prohibitively large, and the dataset is too small for learning every combination. Moreover, over-sized probability tables make inference extremely slow. Neural network based approaches solve this problem by expressing functions with a general high capacity approximator at the cost of higher computational requirements (relative to the small factorized model, but not always), less interpretability and fewer theoretic guarantees.

In [9] and [14], multi-layered LSTMs are used to model Bach's four-part chorales. For generation, the former uses Gibbs sampling and the later uses greedy search. In [10], a neural autoregressive distribution estimator is used to model the same Bach Chorales dataset, and for generation, authors compare Gibbs sampling, block Gibbs sampling, and ancestral sampling. In [22] and [6], Generative Adversarial Networks (GAN) are used to model and generate music pieces in their MIDI piano-roll form, and for generation, GAN based models sample the result directly without the need for an iterative sampling procedure.

However, most existing models, during training, adapt to specific music structures of the corpus being modeled. As our first attempt to extend the expressiveness of a music language model, we wonder if there is some invariance that can be exploited to obtain better generality. It is commonly believed that Bach wrote his chorale harmonizations by firstly writing out basslines for given melodies and then filling in inner voices (Alto, Tenor) [15]. Also, rules for each part (voice) share much in common, for example, a single part tends to move in the reverse direction after a leap. This motivated our idea of treating parts as the basic entity to model.

In this paper, we propose a part-invariant model for multi-part music¹. Our generation framework follows the *Markov Blanket* formalism used in DeepBach [9]. Our model is a part-based model. As a basic consideration of counterpoint, each part should be in a good shape by itself, and when multiple parts are put together, the resulting



© Yujia Yan^b, Ethan Lustig^a, Joseph VanderStel^a, Zhiyao Duan^b. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Yujia Yan^b, Ethan Lustig^a, Joseph VanderStel^a, Zhiyao Duan^b. "Part-invariant Model for Music Generation and Harmonization", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ Supplementary materials and some generation examples can be found at <http://www.ece.rochester.edu/projects/air/projects/model0.html>

aggregated sonority should be good. By *part-invariance*, we mean that the structure of our model explicitly captures the relationship among notes within every single part, and we share this structure with all parts of the score. A separate structure aggregates the information of how different parts would look like when joined together. As a result, our model is capable of expressing/processing music scores with any number of parts using a single trained model.

2. MULTI-PART MUSIC

In this work, we focus on music containing multiple monophonic parts (voices). For example, most of Bach chorales were written in the SATB format (Soprano, Alto, Tenor, and Bass), with each part containing a monophonic stream of notes. It is a traditional pedagogical practice to teach fundamental concepts of music theory by having students analyze and compose (i.e. “part write”) this kind of music. When analyzing or composing music, we often separate a musical score into streams of notes [2], consciously or unconsciously. This part-separated form of music scores is easier to analyze and manipulate algorithmically, and many symbolic music analysis tasks use this separation as one of their preprocessing steps [7]. There are some existing approaches to perform part (voice) segmentation; see [7, 8] for more details. Therefore, our proposed technique focuses on encoding a part-segmented representation assuming the segmentation is known.

2.1 Representation

In traditional western music notation, durations of notes are derived by uniformly dividing a duration of a unit length recursively. Notes start and end on a subdivided position. It is thus reasonable to represent a music score as events on a grid, with each grid point representing a time frame. This process is commonly known as *quantization*. This practice can be seen in many works, e.g., [1, 9, 14, 22]. In this work, we keep the quantization step size fixed throughout the piece.

We encode two aspects of a music score: *pitch* and *metrical structure*. We make the following requirements for this representation:

1. This representation is able to encode a minimal set of musical notational elements, from which the reconstructed music score is human-readable.
2. Values at the same beat position under different quantization step sizes are the same.

Existing works make use of MIDI pitch numbers for encoding pitch. However, MIDI pitch numbers discard one element that is important for context determination: note spelling. In the proposed representation, pitch is represented by a tuple (*diatonic note number*, *accidental*), where *diatonic note number* is the index of a note name with accidental removed (imagine the indices for white keys on a piano keyboard), and *accidental* has a range of $[-2, 2]$, that is, up to 2 flats and 2 sharps. For representing

a whole note event, similar to [9, 17], we use a special continuation symbol, which is -1 in the diatonic note number field. For positions of rest notes, we artificially set their diatonic note number to 0. Accidentals are undefined in these two cases, therefore zeros can be filled in.

We encode the metrical structure into three simultaneous sequences sharing the same time resolution as the pitch frames: 1) *Bar Line* is a binary sequence encoding measure boundaries. A value of 1 is assigned to the frame at the first beat of a measure, and 0 is assigned elsewhere. 2) *Beat Level* encodes a frame’s beat (sub-)division level in the metric hierarchy within a measure. Frames at the highest beat division level are assigned a value of 0; frames at the next level are assigned -1 , etc. 3) *Accent Level* encodes the relative strength of beat positions of frames within a measure, with 0 representing the highest strength and -1 representing the second highest strength, etc. For example, for a classical 4/4 time signature, the frame at the first beat of a measure is assigned 0, the frame at the third beat is assigned -1 , etc.

The first two encoding sequences work together to make it possible to reconstruct bar lines and the time signature. The third sequence further encodes metrical accents that are indicative of different music styles, and regular/irregular metrical structures.

Bar Line	1	0	0	0	0	0	0	0
Beat Level	0	-1	0	-1	0	-1	0	-1
Accent Level	0	-3	-2	-3	-1	-3	-2	-3

3. THE PART-INVARIANT MODEL

3.1 Model Architecture

Following the general practice of language models, our model predicts one symbol at a position given its (musical) context, that is,

$$P(x_{t,k} | context_{t,k}),$$

where t is the time frame index, k is the part index, and $x_{t,k}$ is the pitch representation at position (t, k) . We further assume $context_{t,k}$ to be able to separate $x_{t,k}$ from influences of all other variables (*Markov Blanket* assumption). This *Markov Blanket* formalism is also used in [9].

For obtaining a vector summarizing the context for part k and frame t , after masking the symbol at the position (t, k) as a special *UNK* symbol, we first use a part-wise summarizer, which is a single-layered bidirectional RNN², to produce a *part-wise context vector* for each part. Then all part-wise context vectors are aggregated by one of reduction operations, e.g., *max*, *min*, *sum*, along the axis of part indexes, to produce an *aggregated context vector*. We also summarize the metrical structure (bar line, beat level, and accent level) with another single-layered bidirectional RNN to produce a *metrical context vector*. Finally, for time frame t , the part-wise context vector for part k , the

² *Bidirectional* here means that the output is a concatenation of outputs for the same time step from two RNNs with opposite directions.

aggregated context vector, and the metrical context vector are concatenated and fed into a feed-forward network with a *softmax* output layer to obtain the final prediction $P(x_{t,k} | context_{t,k})$.

Our model is illustrated in Figure 1. Inputs to the part-wise context summarizer are vector embeddings described in Section 3.1.1. Inputs to the metrical context summarizer are raw metrical sequences, and the RNN structure used in our experiment is described in Section 3.1.2.

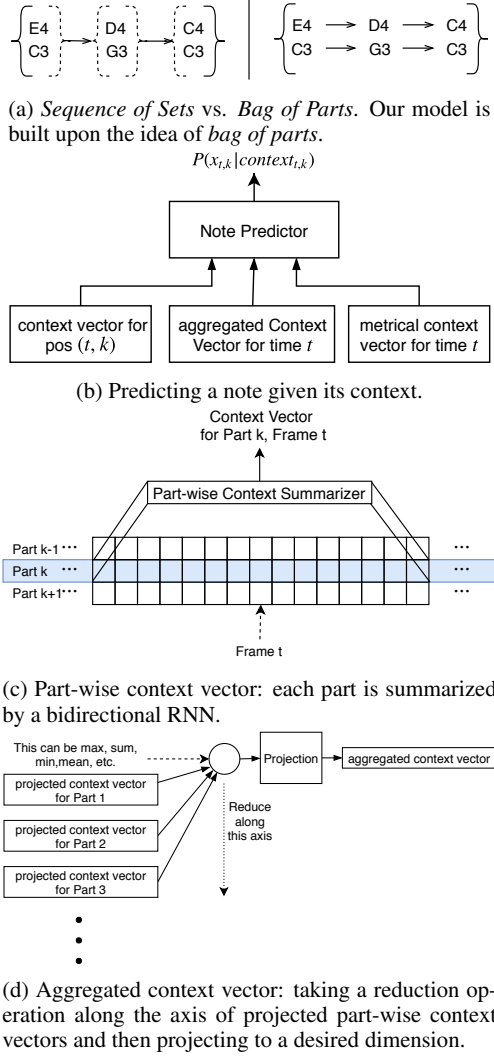


Figure 1: Model Architecture.

3.1.1 Structured Pitch Vector Embedding

An embedding layer, which is usually the first layer of neural networks for modeling discrete symbols, learns a vector representation for each symbol. For embedding pitches, if each pitch is treated as a separate symbol, some general relationships that are already known (e.g., octave equivalence, intervals) will be lost. Therefore, we propose to use a factorized vector embedding representation (i.e., multiple terms in Eq. (1)) for each pitch for better generality.

For readers not familiar with embedding layers, one can treat V_k 's below as lookup tables, each of which creates

one entry (vector) for every possible value it takes.

The final vector embedding $V(p)$ is the sum of a series of embedding vectors, with each encoding a different “aspect” of a pitch.

$$V(p) = V_1(\text{diatonicPitchClass}(d)) + V_2(d) + V_3(p) + V_4(\text{MIDI}(p)) + V_5(\text{chromaticPitchClass}(\text{MIDI}(p))), \quad (1)$$

where $p = (d, acc)$ is the pitch tuple defined in Section 2, with d being the diatonic note number and acc being the accidental; $\text{MIDI}(\cdot)$ is the MIDI pitch number; $\text{diatonicPitchClass}(\cdot)$ and $\text{chromaticPitchClass}(\cdot)$ wrap numbers according to octave equivalence; V is the final vector embedding; V_1, V_2, V_3, V_4, V_5 are vector embeddings for different aspects. These vector embeddings are jointly learned during training.

3.1.2 Stack Augmented Multiplicative Gated Recurrent Unit

The temporal dependency can be long for a representation using fine quantized time frames. In this work, instead of using standard LSTMs, we use a stack augmented multiplicative Gated Recurrent Unit as the RNN block. The GRU part implements the short-term memory. We choose the stack mechanism [11] for the long-term memory because of its resemblance to the pushdown automata, which has more expressive power than a finite state machine and is able to recognize context-free languages, which are often used to model some elements in music.

We make the following convention for our notation: unbolded lowercase letters, e.g. a , denote scalars; bolded lowercase letters, e.g. \mathbf{x}, \mathbf{h} , denote vectors; bolded uppercase letters, e.g. \mathbf{W}, \mathbf{S} , denote matrices.

The original GRU, as introduced in [4], transforms the input sequence of $\langle \mathbf{x}_t \rangle$ into a sequence of $\langle \mathbf{h}_t \rangle$, where t is the time step index:

$$\begin{aligned} \mathbf{r}_t &= \sigma(\mathbf{W}_r[\mathbf{x}_t; \mathbf{h}_{t-1}] + \mathbf{b}_r), \\ \mathbf{u}_t &= \sigma(\mathbf{W}_u[\mathbf{x}_t; \mathbf{h}_{t-1}] + \mathbf{b}_u), \\ \mathbf{c}_t &= \tanh(\mathbf{W}_c[\mathbf{x}_t; \mathbf{r}_t \odot \mathbf{h}_{t-1}] + \mathbf{b}_c), \\ \mathbf{h}_t &= \mathbf{u}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{u}_t) \odot \mathbf{c}_t, \end{aligned} \quad (2)$$

where \mathbf{r}_t is the reset gate, \mathbf{u}_t is the update gate, \mathbf{c}_t is the update candidate, $\sigma(x) = \frac{1}{1+e^{-x}}$ is the Sigmoid function. \mathbf{W} 's and \mathbf{b} 's are all trainable parameters, representing weights and biases respectively, \odot here represents element-wise multiplication, $[\mathbf{x}_t; \mathbf{h}_{t-1}]$ concatenates vectors into a longer column vector.

Multiplicative integration [21] adds quadratic terms into RNN update equations in order to improve the expressive power. In our implementation, we replace the equation for the update candidate with

$$\begin{aligned} \mathbf{c}_{t,x} &= \mathbf{W}_{cx} \mathbf{x}_t, \\ \mathbf{c}_{t,r1} &= \mathbf{W}_{cr1}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_{cr1}, \\ \mathbf{c}_{t,r2} &= \mathbf{W}_{cr2}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_{cr2}, \\ \mathbf{c}_t &= \tanh(\mathbf{c}_{t,x} \odot (\mathbf{c}_{t,r1} + 1) + \mathbf{c}_{t,r2}). \end{aligned} \quad (3)$$

A stack-based external memory for RNN is introduced in [11], which is reported to be able to learn some sequences that are not learnable by a traditional RNN, e.g., LSTM.

We denote the stack as a matrix \mathbf{S}_t , with dimensions of $N \times M$, where N is the length of one entry in the stack, and M is the capacity of the stack. In our implementation, a stack augmented memory performs the following procedure at each time step:

1. Fetch \mathbf{v}_t from the stack by positional attention, which is a linear combination of columns in \mathbf{S}_t with weights \mathbf{k}_t :

$$\begin{aligned} \mathbf{k}_t &= \text{softmax}(\mathbf{W}_{\text{read}}[\mathbf{x}_t; \mathbf{h}_{t-1}] + \mathbf{b}_{\text{read}}), \\ \mathbf{v}_t &= \mathbf{S}_t \mathbf{k}_t; \end{aligned} \quad (4)$$

where $\text{softmax}(\mathbf{x}) = \frac{\exp(\mathbf{x})}{1^T \exp(\mathbf{x})}$;

2. Augment the input with the fetched value:

$$\tilde{\mathbf{x}}_t = [\mathbf{x}_t; \mathbf{v}_t], \quad (5)$$

and then run one step RNN with input $\tilde{\mathbf{x}}_t$, which produces \mathbf{h}_t ;

3. Generate the input to the stack:

$$\mathbf{z}_t = \tanh(\mathbf{W}_z[\tilde{\mathbf{x}}_t; \mathbf{h}_t] + \mathbf{b}_z); \quad (6)$$

4. Make decisions on how to update the stack:

$$\begin{bmatrix} a_{t,\text{no-op}} \\ a_{t,\text{push}} \\ a_{t,\text{pop}} \end{bmatrix} = \text{softmax}(\mathbf{W}_a[\tilde{\mathbf{x}}_t; \mathbf{h}_t] + \mathbf{b}_a), \quad (7)$$

where a 's are probabilities that sum up to 1 representing the probability of stack operations (no operation, push, pop);

5. Update the stack by expectation of operations:

$$\begin{aligned} \mathbf{S}_{t,\text{pushed}} &= [\mathbf{z}_t, \text{first}_k(\mathbf{S}_{t-1})], \\ \mathbf{S}_{t,\text{popped}} &= [\text{last}_k(\mathbf{S}_{t-1}), \mathbf{0}], \\ \mathbf{S}_t &= a_{t,\text{no-op}} \mathbf{S}_{t-1} \\ &\quad + a_{t,\text{push}} \mathbf{S}_{t,\text{pushed}} \\ &\quad + a_{t,\text{pop}} \mathbf{S}_{t,\text{popped}}, \end{aligned} \quad (8)$$

where $\text{first}_k(\cdot)$ extracts the first k columns, and $\text{last}_k(\cdot)$ extracts the last k columns. Here $k = M - 1$. Operator $[\cdot]$ concatenates vectors/matrices horizontally.

3.1.3 Context Aggregation: Obtaining Part-Invariance

As mentioned above, the aggregated context vector is obtained by reduction operations on projected part-wise context vectors, and is then projected to the desired dimension.

$$\mathbf{C}_t^{\text{aggregated}} = \mathbf{W}_{\text{proj2}} \left(\bigoplus_{k=1}^K \mathbf{W}_{\text{proj1}} \mathbf{C}_{t,k}^{\text{part}} \right), \quad (9)$$

where $\mathbf{C}_t^{\text{aggregated}}$ and $\mathbf{C}_{t,k}^{\text{part}}$ are aggregated context vector and partwise context vector respectively, $\bigoplus_{k=1}^K$ denotes a

reduction operator over k from 1 to K , where K is the number of parts, $\mathbf{W}_{\text{proj1}}$ and $\mathbf{W}_{\text{proj2}}$ are projection matrices for transforming the context vector into a higher dimension and back in order to improve the expressiveness of this reduction operation. In our experiment, we use max reduction. The proof of the universal approximation property for approximating a continuous set function when max reduction is used can be found in [3].

The reduction operation applied here produces a *continuous bag of parts* (bag means (multi-)set). This terminology draws similarity to *continuous bag of words* (CBOW, [16]), which averages all vector embeddings for all words (mean reduction) within a window to obtain the vector representation for this context. For comparison, existing works conceptually make use of *sequence-of-set* paradigm for context modeling (see Figure 1a), therefore the context model is confined to learning sequential relationships between sets. Our conceptual paradigm is on a different direction. We built a model for processing monophonic parts and a model for putting them into a bag. One important feature for doing this is that it allows learning shared properties of parts. Also, the ordering of parts, which is redundant for a context encoder, is discarded and only the content information of all parts is aggregated. As a result, it reduces the model complexity required.

3.2 Sampling and Generation

After training the Markov blanket model for approximating the probability of a note conditioned on its context, $P(x_{t,k} | \text{context}_{t,k})$, the process of generation is performed by Gibbs sampling with an annealing schedule. This procedure is almost the same as the one used in [9].

Firstly, we initialize notes $x_{t,k}$ of all positions in the empty parts randomly. Then we iterate:

1. Randomly or deterministically select the next position (t, k) that is not fixed³ to sample;
2. Sample new $x_{t,k}$, according to

$$\tilde{P}(\cdot | \text{context}_{t,k}) \propto (P(\cdot | \text{context}_{t,k}))^{1/T}, \quad (10)$$

i.e., the annealed distribution with temperature $T > 0$;

For vanilla Gibbs sampling, $T \equiv 1$. However, as pointed out in [9], conditional distributions outputted are likely to be incompatible and there is no guarantee that the Gibbs sampler will converge to the desired joint distribution.

In Gibbs sampling with an annealing schedule, the temperature starts from a high value and gradually decreases to a low value. By incorporating this annealing scheme, the algorithm can escape from initial bad values much easier at the beginning, and the average likelihood for the selected new samples increases as the temperature decreases. For illustration, in the limiting case that $T \rightarrow 0$, the algorithm

³ Fixed positions are used as conditions. For example, if the task of melody harmonization, the melody part is fixed.

greedily selects new samples that maximizes the local likelihood.

Since in our model parts are orderless, the generated result does not ensure all parts in their usual notated staves. Also, the imperfection of Gibbs sampler often makes the configuration get stuck in a region where voice crossing occurs even if parts in the training set rarely cross. In the experiment, we enforce one constraint during sampling as a workaround: in each time frame, the pitch of each part cannot go above/below the part that is immediately above/below it, i.e., no voice crossing is allowed. This constraint is achieved by limiting the range of candidates to sample. How to design a better sampling procedure is left for future investigation.

4. EXPERIMENT

4.1 Training

4.1.1 Dataset

We trained our model on the Bach Chorale dataset included in Music21 [5]. We chose this dataset to perform our experiment for the following reasons: firstly, it is publicly available; secondly, it matches the objective evaluation methods we designed⁴; thirdly, there is no need to perform voice separation in this dataset. Different parts are separately encoded in the file format.

We performed data augmentation by transposition with a range such that the transposed piece is within the lowest pitch minus 5 semitones to a highest pitch plus 5 semitones for the whole dataset. Enharmonic spellings are resolved by selecting the one that creates the minimum number of accidentals for the entire transposed piece.

4.1.2 Model Specification

In our experiment, we use a quantization step size of a sixteenth note. Embedding layers have a dimension of 200. We use single-layered RNNs as the partwise context summarizer and metrical sequence summarizer. All RNNs have a hidden state size of 200, stack vector length 200, stack size 24. The intermediate dimension for the part aggregating layer is 1000. The final predictor is a feed-forward neural network with 3 layers, each of which contains 400 hidden units. The final softmax layer has a dimension of 400, each corresponding to a specific pitch tuple (diatonic note number, accidental). We use cross entropy as the loss function. Curriculum learning is used in our training: we started from a small half-window width of 8 and gradually doubled the half-window width to a maximum of 128. During training, pitches within the context window are randomly set to a rest with probability 0.1. All layers except for the RNN layers use a dropout rate of 0.1.

In our evaluation, we use a half-window width of 64 for generation. We use a simple linear cooling schedule to decrease the temperature from an initial value of 1.2 to

0.25. The total number of iterations is selected such that every position is sampled 40 times.

Music scores are reconstructed by directly using accidentals, diatonic note numbers and the original encoded metrical sequences. Key signatures and clefs are automatically determined by Music21's [5] built-in functions.

4.2 Evaluation

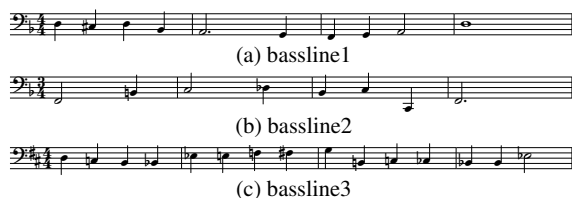


Figure 2: Basslines used in our evaluation.

To perform evaluation, we compared our algorithm's harmonizations of basslines with harmonizations of those same basslines completed by music students. We used three basslines which vary in difficulty, ranging from diatonic (bassline 1) to moderately chromatic (bassline 2) to highly chromatic (bassline 3).⁵ For each bassline, our algorithm generated 30 outputs, for a total of 30*3 outputs. As a side note, 4 bars is the usual length for a harmonization exercise. This length is different from lengths of pieces in the training set.

We recruited 33 second-semester sophomore music majors, offering them extra credit for harmonizing each bassline. We gave each student a .xml file containing the three basslines, with three blank upper staves. We instructed students to harmonize each of the basslines in four-part, SATB chorale style, following the usual rules of voice-leading and harmony. We used valid responses from 27 students (those not empty and returned timely) in the following evaluation tasks.

We recruited two teams for evaluation: graders and listeners. The graders were three music theory PhD students. They were given the 57 valid outputs (57 * 3 in total) in .pdf format; we created a grading rubric⁶. A deduction less than 0 was computed by each grader for each output. The lower the value, the greater the number of errors. One graded example can be found in Figure 3.



Figure 3: Example annotation from one of our graders.

⁴ The objective evaluation follows rules used in textbook part writing, which are greatly influenced by Bach Chorales, however, these rules are not strictly followed by Bach himself.

⁵ Basslines 1 and 2 were taken from Exercises 10.3C and 21.2, respectively, from [12]. In Bassline 2, the B was originally a Bb in [12], but we changed it to increase chromaticism. Bassline 3 was created by us, and intended to represent highly modulatory chromatic harmony.

⁶ The rubric is typical of traditional music theory textbooks and classes. For the detailed rubric, see the supplementary website.

While the grading method was fairly objective (correlations between error values from the three graders were .85, .88, and .92), we also wanted subjective ratings. In addition, we recruited a listening team of another three music theory PhD students. We gave them the same 57 * 3 outputs in .mp3 format, synthesized with a software piano synthesizer and tempo 93, and these instructions:

For each output, answer the following four questions:

1. *As you listen, how much are you enjoying this solution (on a scale of 1 to 4, where 1 = not enjoying at all and 4 = greatly enjoying)?*
2. *As you listen, how confident are you that this solution is by a computer vs. a sophomore (on a scale of 1 to 4, where 1 = probably a computer and 4 = probably a sophomore)?*
3. *As you listen, to what extent does this solution conform to textbook/common-practice voice-leading and harmony (on a scale of 1 to 4, where 1 = not very idiomatic and 4 = quite idiomatic)?*
4. *Please share any other comments or thoughts (for example, why does it sound like it's a computer vs. a sophomore?)*

To summarize, for each output we had 3 gradings (1 value * 3 graders) and 9 subjective ratings (3 ratings*3 listeners) plus additional open-ended comments.

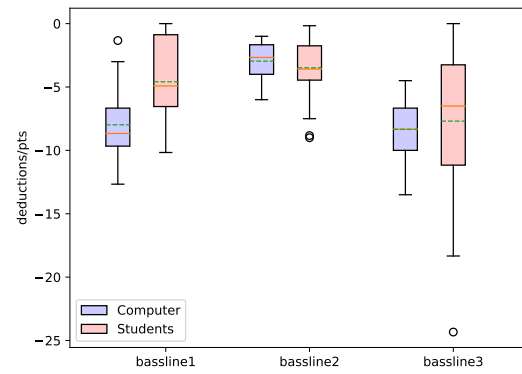
To minimize bias, graders only received .pdf outputs; listeners only received .mp3 outputs. The outputs were presented to the graders and listeners in random order. Both teams were blind to the output source (computer or student), and were allowed to take as much time as they needed to make their assessments.

Our experiment result is summarized in Figure 4. Our experiment shows that gradings and listening ratings for our algorithm and students overlap to different extents (our algorithm performs best on the second bassline). For the listening test, our algorithm consistently performs a bit worse than average second-year second-semester music majors.

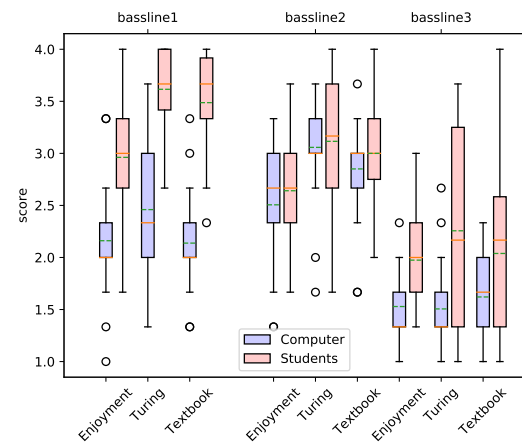
The comments from the listener who contributed most of the open-ended comments (question 4) suggest that the presence of *tonality* was one of the main factors in their Turing judgements. This listener attributed harmonizations that feature small stylistic errors (e.g., oddly repeated notes, parallel voice leading, etc.) to both human and computer, but those harmonizations that sounded resolutely tonal were only attributed to humans. Another listener seemed to ground their judgments on a different feature: “A lot of the ones I think are computer-generated do cadences super well.” Indeed, for the most part the computer did generate well-formed cadences.

By examining the detailed responses from our graders, we have the following rudimentary observations:

1. Errors of our algorithm and students are differently distributed.
2. Parallel octave/fifth (Error 3) is one frequent error produced by our algorithm, more often than students. This type of error is also observed in generation examples shown in [9].



(a) Results for the objective grading test.



(b) Results for the subjective listening test.

Figure 4: Objective and subjective comparisons between our algorithm’s and music students’ harmonization on the three basslines.

3. Our algorithm produces more non-stylistic progressions (Error 6 and Error 7). In our algorithm, it is observed that, smooth/melodic voice leading may sometimes suppress the requirement of the vertical sonority.
4. Students are much more likely to exceed the octave range limit between nearby upper voices (Error 9)

From our experiment result, it is revealed that the proposed algorithm cannot learn what is bad/incorrect just by watching correct examples. Therefore, there is a need to train with negative examples. Our experiment provides useful data for future development.

5. CONCLUSION

In this work, we proposed a part-invariant model for music generation and harmonization that operates on multi-part music scores, which are scores containing multiple monophonic parts. We trained our model on Bach Chorales dataset. We performed objective and subjective evaluations by comparing the outputs of our algorithm against the textbook-style part writings of undergraduate music majors. Our experiment result provides insights and data that will be useful for future development.

6. REFERENCES

- [1] Moray Allan and Christopher Williams. Harmonising chorales by probabilistic inference. In *Advances in neural information processing systems*, pages 25–32, 2005.
- [2] Albert S. Bregman. *Auditory scene analysis*. MIT Press, 1996.
- [3] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.
- [4] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.
- [5] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data.
- [6] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *AAAI-18 AAAI Conference on Artificial Intelligence*, 2018.
- [7] Patrick Gray and Razvan C Bunescu. A neural greedy model for voice separation in symbolic music. In *International Society for Music Information Retrieval Conference (ISMIR 2016)*, pages 782–788, 2016.
- [8] Nicolas Guiomard-Kagan, Mathieu Giraud, Richard Groult, and Florence Levé. Comparing voice and stream segmentation algorithms. In *International Society for Music Information Retrieval Conference (ISMIR 2015)*, pages 493–499, 2015.
- [9] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation. In *International Conference on Machine Learning*, pages 1362–1371, 2017.
- [10] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron C. Courville, and Douglas Eck. Counterpoint by convolution. In *ISMIR*, pages 211–218, 2017.
- [11] Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in neural information processing systems*, pages 190–198, 2015.
- [12] Steven G. Laitz. *Writing and analysis workbook to accompany The complete musician: an integrated approach to tonal theory, analysis, and listening, 3rd edition*. Oxford University Press, 2012.
- [13] Victor Lavrenko and Jeremy Pickens. Polyphonic music modeling with random fields. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 120–129. ACM, 2003.
- [14] Feynman T. Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. Automatic stylistic composition of bach chorales with deep lstm. In *ISMIR*, pages 449–456, 2017.
- [15] Robert L Marshall. How JS Bach composed four-part chorales. *The Musical Quarterly*, 56(2):198–220, 1970.
- [16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [17] François Pachet, Alexandre Papadopoulos, and Pierre Roy. Sampling variations of sequences for structured music generation. In *ISMIR*, pages 167–173, 2017.
- [18] Martin Rohrmeier. A generative grammar approach to diatonic harmonic structure. In *Proceedings of the 4th sound and music computing conference*, pages 97–100, 2007.
- [19] Ian Simon, Dan Morris, and Sumit Basu. Mysong: automatic accompaniment generation for vocal melodies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 725–734. ACM, 2008.
- [20] Andries Van Der Merwe and Walter Schulze. Music generation with markov models. *IEEE MultiMedia*, 18(3):78–85, 2011.
- [21] Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan R Salakhutdinov. On multiplicative integration with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 2856–2864, 2016.
- [22] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR2017)*, Suzhou, China, 2017.

EVALUATING LANGUAGE MODELS OF TONAL HARMONY

David R. W. Sears¹

Filip Korzeniowski²

Gerhard Widmer²

¹ College of Visual & Performing Arts, Texas Tech University, Lubbock, USA

² Institute of Computational Perception, Johannes Kepler University, Linz, Austria

david.sears@ttu.edu

ABSTRACT

This study borrows and extends probabilistic language models from natural language processing to discover the syntactic properties of tonal harmony. Language models come in many shapes and sizes, but their central purpose is always the same: to predict the next event in a sequence of letters, words, notes, or chords. However, few studies employing such models have evaluated the most state-of-the-art architectures using a large-scale corpus of Western tonal music, instead preferring to use relatively small datasets containing chord annotations from contemporary genres like jazz, pop, and rock.

Using symbolic representations of prominent instrumental genres from the common-practice period, this study applies a flexible, data-driven encoding scheme to (1) evaluate Finite Context (or n -gram) models and Recurrent Neural Networks (RNNs) in a chord prediction task; (2) compare predictive accuracy from the best-performing models for chord onsets from each of the selected datasets; and (3) explain differences between the two model architectures in a regression analysis. We find that Finite Context models using the Prediction by Partial Match (PPM) algorithm outperform RNNs, particularly for the piano datasets, with the regression model suggesting that RNNs struggle with particularly rare chord types.

1. INTRODUCTION

For over two centuries, scholars have observed that tonal harmony, like language, is characterized by the logical ordering of successive events, what has commonly been called *harmonic syntax*. In Western music of the common-practice period (1700-1900), pitch events group (or cohere) into discrete, primarily tertian sonorities, and the succession of these sonorities over time produces meaningful syntactic progressions. To characterize the passage from the first two measures of Bach’s “Aus meines Herzens Grunde”, for example, theorists and composers developed a chord typology that specifies both the scale steps on which tertian sonorities are built (*Stufentheorie*), and the

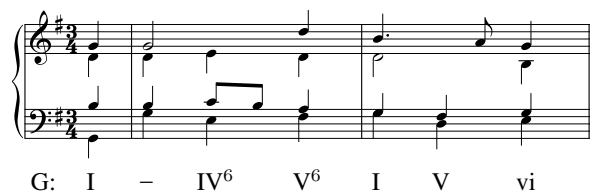


Figure 1. Bach, “Aus meines Herzens Grunde”, mm. 1–2; from the Riemenschneider edition, No. 1. Key and Roman numeral annotations appear below.

functional (i.e., temporal) relations that bind them (*Funktionstheorie*). Shown beneath the staff in Figure 1, this *Roman numeral* system allows the analyst to recognize and describe these relations using a simple lexicon of symbols.

In the presence of such language-like design features, music scholars have increasingly turned to string-based methods from the natural language processing (NLP) community for the purposes of pattern discovery [6], classification [7], similarity estimation [18], and prediction [19]. In sequential prediction tasks, for example, probabilistic language models have been developed to predict the next event in a sequence — whether it consists of letters, words, DNA sequences, or in our case, chords.

Although corpus studies of tonal harmony have become increasingly commonplace in the music research community, applications of language models for chord prediction remain somewhat rare. This is likely because language models take as their starting point a sequence of chords, but the musical surface is often a dense web of chordal and nonchordal tones, making automatic harmonic analysis a tremendous challenge. Indeed, such is the scope of the computational problem that a number of researchers have instead elected to start with a particular chord typology right from the outset (e.g., Roman numerals, figured bass nomenclature, or pop chord symbols), and then identify chord events using either human annotators [3], or rule-based computational classifiers [25]. As a consequence, language models for tonal harmony frequently train on relatively small, heavily curated datasets (< 200, 000 chords) [3], or use data augmentation methods to increase the size of the corpus [15]. And since the majority of these corpora reflect pop, rock, or jazz idioms, vocabulary reduction is a frequent preliminary step to ensure improved model performance, with the researcher typically including specific chord types (e.g., major, minor, seventh, etc.), thus ignor-



© Sears, Korzeniowski, Widmer. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Sears, Korzeniowski, Widmer. “Evaluating language models of tonal harmony”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

ing properties of tonal harmony relating to inversion [15] or chordal extension [11].

Given the state of the annotation bottleneck, we propose a complementary method for the implementation and evaluation of language models for chord prediction. Rather than assume a particular chord typology a priori and train our models on the *chord classes* found therein, we will instead propose a data-driven method for the construction of harmonic corpora using *chord onsets* derived from the musical surface. It is our hope that such a bottom-up approach to chord prediction could provide a springboard for the implementation of chord class models in future studies [2], the central purpose of which is to use predictive methods to reduce the musical surface to a sequence of syntactic progressions by discovering a small vocabulary of chord types.

We begin in Section 2 by describing the datasets used in the present research and then present the tonal encoding scheme that reduces the combinatoric explosion of potential chord types to a vocabulary consisting of roughly two hundred types for each scale-degree in the lowest instrumental part. Next, Section 3 describes the two most state-of-the-art architectures employed in the NLP community: Finite Context (or n -gram) models and Recurrent Neural Networks (RNNs). Section 4 presents the experiments, which (1) evaluate the two aforementioned model architectures in a chord prediction task; (2) compare predictive accuracy from the best-performing models for each dataset; (3) attempt to explain the differences between the two models using a regression analysis. We conclude in Section 5 by considering limitations of the present approach, and offering avenues for future research.

2. CORPUS

This section presents the datasets used in the present research and then describes the chord representation scheme that permits model comparison across datasets.

2.1 Datasets

Shown in Table 1, this study includes nine datasets of Western tonal music (1710–1910) featuring symbolic representations of the notated score (e.g., metric position, rhythmic duration, pitch, etc.). The Chopin dataset consists of 155 works for piano that were encoded in MusicXML format [10]. The Assorted symphonies dataset consists of symphonic movements by Beethoven, Berlioz, Bruckner, and Mahler that were encoded in MATCH format [26]. All other datasets were downloaded from the KernScores database in MIDI format.¹ In total, the composite corpus includes the complete catalogues for Beethoven’s string quartets and piano sonatas, Joplin’s rags, and Chopin’s piano works, and consists of over 1,000 compositions containing more than 1 million chord tokens.

¹ <http://kern.ccarh.org/>.

<i>Composer</i>	<i>Genre</i>	N_{pieces}	N_{tokens}	N_{types}
Bach	Chorale	370	35,237	786
Haydn	Quartet	210	159,579	1472
Mozart	Quartet	82	78,201	1289
Beethoven	Quartet	70*	132,896	1699
Mozart	Piano	51	92,279	833
Beethoven	Piano	102*	176,370	1332
Chopin	Piano	155*	147,827	1790
Joplin	Piano	47*	43,848	854
Assorted	Symphony	29	147,549	2420
<i>Total</i>		1116	1,013,786	2590

Note. * denotes the complete catalogue.

Table 1. Datasets and descriptive statistics for the corpus.

2.2 Chord Representation Scheme

To derive chord progressions from symbolic corpora using data-driven methods, music analysis software frameworks typically perform a *full expansion* of the symbolic encoding, which duplicates overlapping note events at every unique onset time. Shown in Figure 2, expansion identifies 9 unique onset times in the first two measures of Bach’s chorale harmonization, “Aus meines Herzens Grunde.”

Previous studies have represented each chord according to the simultaneous relations between its note-event members (e.g., vertical intervals) [23], the sequential relations between its chord-event neighbors (e.g., melodic intervals) [6], or some combination of the two [22]. For the purposes of this study, we have adopted a chord typology that models every possible combination of note events in the corpus. The encoding scheme consists of an ordered tuple (S, I) for each chord onset in the sequence, where S is a set of up to three intervals above the bass in semitones modulo the octave (i.e., 12), resulting in 13^3 (or 2197) possible combinations;² and I is the chromatic scale degree (again modulo the octave) of the bass, where 0 represents the tonic, 7 the dominant, and so on.

Because this encoding scheme makes no distinction between chord tones and non-chord tones, the syntactic domain of chord types is still very large. To reduce the domain to a more reasonable number, we have excluded pitch class repetitions in S (i.e., voice doublings), and we have allowed permutations. Following [22], the assumption here is that the precise location and repeated appearance of a given interval are inconsequential to the identity of the chord. By allowing permutations, the major triads $\langle 4, 7, 0 \rangle$ and $\langle 7, 4, 0 \rangle$ therefore reduce to $\langle 4, 7, \perp \rangle$. Similarly, by eliminating repetitions, the chords $\langle 4, 4, 10 \rangle$ and $\langle 4, 10, 10 \rangle$ reduce to $\langle 4, 10, \perp \rangle$. This procedure restricts the domain to 233 unique chord types in S (i.e., when I is undefined).

To determine the underlying tonal context of each chord onset, we employ the key-finding algorithm in [1], which tends to outperform other distributional methods (with an

² The value of each vertical interval is either undefined (denoted by \perp), or represents one of twelve possible interval classes, where 0 denotes a perfect unison or octave, 7 denotes a perfect fifth, and so on.

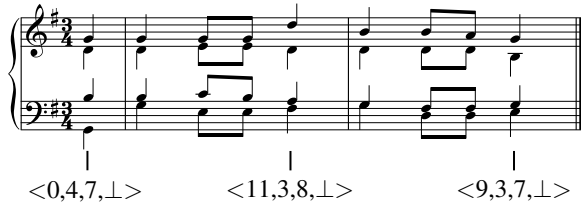


Figure 2. Full expansion of Bach, “Aus meines Herzens Grunde”, mm. 1–2. Three chord onsets are shown with the tonal encoding scheme described in Section 2.2 for illustrative purposes.

accuracy of around 90% for both major and minor keys). Since the movements in this dataset typically feature modulations, we compute the Pearson correlation between the distributional weights in the selected key-finding algorithm and the pitch-class distribution identified in a moving window of 16 quarter-note beats and centered around each chord onset in the sequence. The algorithm interprets the passage in Figure 2 in G major, for example, so the bass note of the first harmony is 0 (i.e., the tonic).

3. LANGUAGE MODELS

The goal of language models is to estimate the probability of event e_i given a preceding sequence of events e_1 to e_{i-1} , notated here as e_1^{i-1} . In principle, these models predict e_i by acquiring knowledge through unsupervised statistical learning of a training corpus, with the model architecture determining how this learning process takes place. For this study we examine the two most common and best-performing language models in the NLP community: (1) Markovian finite-context (or n -gram) models using the PPM algorithm, and (2) recurrent neural networks (RNNs) using both long short-term memory (LSTM) layers and gated recurrent units (GRUs).

3.1 Finite Context Models

Context models estimate the probability of each event in a sequence by stipulating a global order bound (or deterministic context) such that $p(e_i)$ depends only on the previous $n - 1$ events, or $p(e_i | e_{(i-n)+1}^{i-1})$. For this reason, context models are also sometimes called n -gram models, since the sequence $e_{(i-n)+1}^{i-1}$ is an n -gram consisting of a context $e_{(i-n)+1}^{i-1}$, and a single-event prediction e_i . These models first acquire the frequency counts for a collection of sequences from a training set, and then apply these counts to estimate the probability distribution governing the identity of e_i in a test sequence using maximum likelihood (ML) estimation.

Unfortunately, the number of potential n -grams decreases dramatically as the value of n increases, so high-order models often suffer from the *zero-frequency problem*, in which n -grams encountered in the test set do not appear in the training set [27]. The most common solution to this problem has been the *Prediction by Partial Match* (PPM) algorithm, which adjusts the ML estimate for e_i by combining (or *smoothing*) predictions generated at higher

orders with less sparsely estimated predictions from lower orders [5]. Specifically, PPM assigns some portion of the probability mass to accommodate predictions that do not appear in the training set using an *escape method*. The best-performing smoothing method is called *mixtures* (or *interpolated smoothing*), which computes a weighted combination of higher order and lower order models for every event in the sequence.

3.1.1 Model Selection

To implement this model architecture, we apply the variable-order Markov model (called *IDyOM*) developed in [19].³ The model accommodates many possible configurations based on the selected global order bound, escape method, and training type. Rather than select a global order bound, researchers typically prefer an extension to PPM called PPM*, which uses simple heuristics to determine the optimal high-order context length for e_i , and which has been shown to outperform the traditional PPM scheme in several prediction tasks (e.g., [21]), so we apply that extension here. Regarding the escape method, recent studies have demonstrated the potential of *method C* to minimize model uncertainty in melodic and harmonic prediction tasks [12, 21], so we also employ that method here.

To improve model performance, Finite Context models often separately estimate and then combine two subordinate models trained on differed subsets of the corpus: a *long-term* model (LTM+), which is trained on the entire corpus; and a *short-term* (or *cache*) model (STM), which is initially empty for each individual composition and then is trained incrementally (e.g., [8]). As a result, the LTM+ reflects inter-opus statistics from a large corpus of compositions, whereas the STM only reflects intra-opus statistics, some of which may be specific to that composition. Finally, the model implemented here also includes a model that combines the LTM+ and STM models using a weighted geometric mean (BOTH+) [20]. Thus, we report the LTM+, STM, and BOTH+ models for the analyses that follow.⁴

3.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are powerful models designed for sequential modelling tasks. RNNs transform an input sequence \mathbf{x}_1^N to an output sequence \mathbf{o}_1^N through a non-linear projection into a hidden layer \mathbf{h}_1^N , parameterised by weight matrices \mathbf{W}_{hx} , \mathbf{W}_{hh} and \mathbf{W}_{oh} :

$$\mathbf{h}_i = \sigma_h (\mathbf{W}_{hx} \mathbf{x}_i + \mathbf{W}_{hh} \mathbf{h}_{i-1}) \quad (1)$$

$$\mathbf{o}_i = \sigma_o (\mathbf{W}_{oh} \mathbf{h}_i), \quad (2)$$

where σ_h and σ_o are the activation functions for the hidden layer (e.g. the sigmoid function), and the output layer

³ The model is available for download: <http://code.soundsoftware.ac.uk/projects/idyom-project>

⁴ The models featuring the + symbol represent both the statistics from the training set and the statistics from that portion of the test set that has already been predicted.

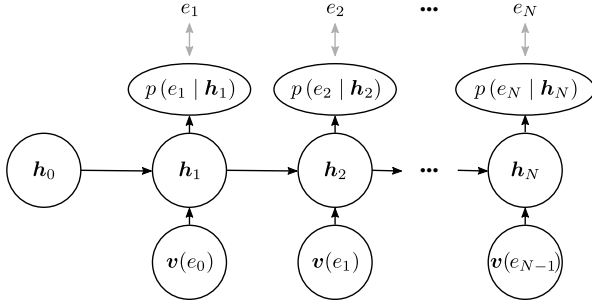


Figure 3. The basic architecture for an RNN-based language model. This model can easily accommodate more recurrent hidden layers or include additional skip-connections between the input and each hidden layer or the output. The first input, e_0 , is a dummy symbol without an associated chord.

(e.g. the softmax), respectively. We excluded bias terms for simplicity.

RNNs have become popular models for natural language processing due to their superior performance compared to Finite Context models [17]. Here, the input at each time step i is a (learnable) vector representation of the preceding symbol, $v(e_{i-1})$. The network’s output $o_i \in \mathbb{R}^{N_{types}}$ is interpreted as the conditional probability over the next symbol, $p(e_i | e_1^{i-1})$. As outlined in Figure 3, this probability depends on *all* preceding symbols through the recurrent connection in the hidden layer.

During training, the categorical cross-entropy between the output o_i and the true chord symbol is minimised by adapting the weight matrices in Eqs. 1 and 2 using stochastic gradient descent and back-propagation through time. However, this training procedure suffers from vanishing and exploding gradients because of the recursive dot product in Eq. 1. The latter problem can be averted by clipping the gradient values; the former, however, is trickier to prevent, and necessitates more complex recurrent structures such as the long short-term memory unit (LSTM) [13] or the gated recurrent unit (GRU) [4]. These units have become standard features of RNN-based language modeling architectures [16].

3.2.1 Model Selection

Selecting good hyper-parameters is crucial for neural networks to perform well. To this end, we performed a number of preliminary experiments to tune the networks. Our final architecture comprises two layers of 128 recurrent units each (either LSTM or GRU), a learnable input embedding of 64 dimensions (i.e. $v(\cdot)$ maps each chord class to a vector in \mathbb{R}^{64}), and skip connections between the input and all other layers.

RNNs are prone to over-fit the training data. We use the network’s performance on held-out data to identify this issue. Since we employ 4-fold cross-validation (see Sec. 4 for details), we hold out one of the three training folds as a validation set. If the results on these data do not improve for 10 epochs, we stop training and select the model with the lowest cross-entropy on the validation data.

We trained the networks for a maximum of 200 epochs, using stochastic gradient descent with a mini-batch size of 4. Each of these 4 data points is a sequence of at most 300 chords. The gradient updates are scaled using the Adam update rule [14] with standard parameters. To prevent exploding gradients, we clip gradient values larger than 1.

4. EXPERIMENTS

4.1 Evaluation

To evaluate performance using a more refined method than one simply based on the accuracy of the model’s prediction, we use a statistic called *corpus cross-entropy*, denoted by H_m .

$$H_m(p_m, e_1^j) = -\frac{1}{j} \sum_{i=1}^j \log_2 p_m(e_i | e_1^{i-1}). \quad (3)$$

H_m represents the average information content for the model probabilities estimated by p_m over all e in the sequence e_1^j . That is, cross-entropy provides an estimate of how uncertain a model is, on average, when predicting a given *sequence* of events [21], regardless of whether the correct symbol for each event was assigned the highest probability in the distribution.

Finally, we employ 4-fold cross-validation stratified by dataset for both model architectures, using cross-entropy as a measure of performance.

4.2 Results

We first compare the average cross-entropy estimates across the entire corpus using Finite Context models and RNNs, and then examine the estimates across datasets for the best performing model configuration from each architecture. We conclude by examining the differences between these models in a regression analysis.

4.2.1 Comparing Models

Table 2 presents the average cross-entropy estimates for each model configuration. For the purposes of statistical inference, we also include the 95% bootstrap confidence interval using the bias-corrected and accelerated percentile method [9]. For the Finite Context models, BOTH+

Model Type	H_m	CI ^a
<i>Finite Context</i>		
LTM+	4.895	4.811–4.978
STM	6.710	6.600–6.820
BOTH+	4.893	4.800–4.966
<i>Recurrent Neural Network</i>		
LSTM	5.583	5.539–5.626
GRU	5.600	5.551–5.645

^a CI refers to the 95% bootstrap confidence interval of H_m using the bias-corrected and accelerated percentile method with 1000 replicates.

Table 2. Model comparison using cross-entropy as an evaluation metric.

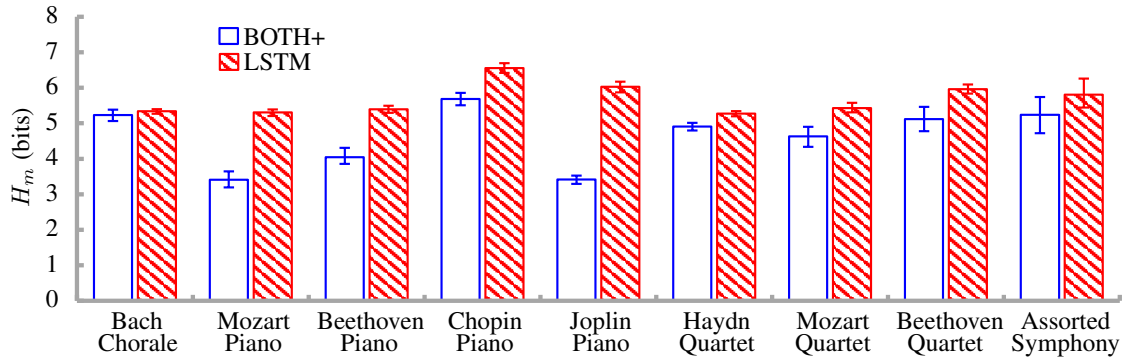


Figure 4. Bar plots of the best-performing model configurations from the Finite Context (BOTH+) and RNN (LSTM) models. Whiskers represent the 95% bootstrap confidence interval of the mean using the bias-corrected and accelerated percentile method with 1000 replicates.

produced the lowest cross-entropy estimates on average, though the difference between BOTH+ and LTM+ was negligible. STM was the worst performing model overall, which is unsurprising given the restrictions placed on the model’s training parameters (i.e., that it only trains on the already-predicted portion of the test set).

Of the RNN models, LSTM slightly outperformed GRU, but again this difference was negligible. What is more, the long-term Finite Context models (BOTH+ and LTM+) significantly outperformed both RNNs. This finding could suggest that context models are better suited to music corpora, since the datasets for melodic and harmonic prediction are generally miniscule relative to those in the NLP community [15]. The encoding scheme for this study also produced a large vocabulary (2590 symbols), so the PPM* algorithm might be useful when the model is forced to predict particularly rare types in the corpus.

4.2.2 Comparing Datasets

To identify the differences between these models for each of the datasets in the corpus, Figure 4 presents the bar plots for the best-performing model configurations from each model architecture: BOTH+ from the Finite Context model, and LSTM from the RNN model. On average, BOTH+ produced the lowest cross-entropy estimates for the piano datasets (Mozart, Beethoven, Joplin), but much higher estimates for the other datasets. This effect was not observed for LSTM, however, with the datasets’ genre — chorale, piano work, quartet, and symphony — apparently playing no role in the model’s overall performance.

The difference between these two model architectures for the Joplin and Mozart piano datasets is particularly striking. Given the degree to which piano works generally consist of fewer homorhythmic textures relative to the other genres in this corpus, it could be the case that the piano datasets feature a larger proportion of rare, monophonic chord types relative to the other datasets. The next section examines this hypothesis using a regression model.

4.2.3 A Regression Model

Given the complexity of the corpus, a number of factors might explain the performance of these models. Thus,

we have included the following five predictors in a multiple linear regression (MLR) model to explain the average cross-entropy estimates for the compositions in the corpus ($N = 1136$):⁵

N_{tokens} Cache (i.e., STM) and RNN-based language models often benefit from datasets that feature longer sequences by exploiting statistical regularities in the portion of the test sequence that was already predicted. Thus, N_{tokens} represents the number of tokens in each sequence. Compositions featuring more tokens should receive lower cross-entropy estimates on average.

N_{types} Language models struggle with data sparsity as n increases (i.e., the zero-frequency problem). One solution is to select corpora for which the vocabulary of possible distinct types is relatively small. Thus, N_{types} represents the number of types in each sequence. Compositions with larger vocabularies should receive higher cross-entropy estimates on average.

Improbable Events that occur with low probability in the zeroth-order distribution are particularly difficult to predict due to the data sparsity problem just mentioned. Thus, *Improbable* represents the proportion of tokens in each sequence that appear in the bottom 10% of types in the zeroth-order probability distribution. Compositions with a large proportion of these particularly rare types should receive higher cross-entropy estimates on average.

Monophonic Chorales feature homorhythmic textures in which each temporal onset includes multiple coincident pitch events. The chord types representing these tokens should be particularly common in this corpus, but some genres might also feature polyphonic textures in which the number of coincident events is potentially quite low (e.g., piano). Thus,

⁵ Four of the 1116 compositions were further subdivided in the selected datasets, producing an additional 20 sequences in the analyses: Beethoven, Quartet No. 6, Op. 18, iv (2); Chopin, Op. 12 (2); Mozart, Piano Sonata No. 6, K. 284, iii (13); Mozart, Piano Sonata No. 11, K. 331, i (7).

Monophonic represents the proportion of tokens in each sequence that consist of only one pitch event. Compositions with a large proportion of these monophonic events should receive higher cross-entropy estimates on average.

Repetition Compared to chord-class corpora, data-driven corpora are far more likely to feature adjacent repetitions of tokens. Thus, *Repetition* represents the proportion of tokens in each sequence that feature adjacent repetitions. Compositions with a large proportion of repetitions should receive lower cross-entropy estimates on average.

Table 3 presents the results of a stepwise regression analysis predicting the average cross-entropy estimates with the aforementioned predictors. R^2 refers to the fit of the model, where a value of 1 indicates that the model accounts for all of the variance in the outcome variable (i.e., a perfectly linear relationship between the predictors and the cross-entropy estimates). The slope of the line measured for each predictor, denoted by β , represents the change in the outcome resulting from a unit change in the predictor.

For the Finite Context model (BOTH+), four of the five predictors explained 53% of the variance in the cross-entropy estimates. As predicted, cross-entropy decreased as the number of tokens increased, suggesting that the model learned from past tokens in the sequence. What is more, cross-entropy increased as the vocabulary increased, as well as when the proportion of monophonic or improbable tokens increased, though the latter two predictors had little effect on the model.

For the RNN model, the effect of these predictors was strikingly different. In this case, cross-entropy increased with the proportion of improbable events. Note that this predictor played only a minor role for the Finite Context model, which suggests PPM* may be responsible for the model’s superior performance. For the remaining predictors, cross-entropy estimates decreased when the proportion of adjacent repeated tokens increased. Like the Finite Context model, the RNN model also struggled when the proportion of monophonic tokens increased, but benefited from longer sequences featuring smaller vocabularies.

5. CONCLUSION

This study examined the potential for language models to predict chords in a large-scale corpus of tonal compositions from the common-practice period. To that end, we developed a flexible chord representation scheme that (1) made minimal a priori assumptions about the chord typology underlying tonal music, and (2) allowed us to create a much larger corpus relative to those based on chord annotations. Our findings demonstrate that Finite Context models outperform RNNs, particularly in piano datasets, which suggests PPM* is responsible for the superior performance, since it assigns a portion of the probability mass to potentially rare, as-yet-unseen types. A regression analysis generally confirmed this hypothesis, with LSTM struggling to predict the improbable types from the piano datasets.

<i>Model</i>	<i>Predictors</i>	β	R^2
BOTH+	N_{tokens}	−2.079	.212
	N_{types}	1.860	.506
	<i>Monophonic</i>	0.233	.506
	<i>Improbable</i>	0.076	.530
LSTM	<i>Improbable</i>	0.463	.318
	<i>Repetition</i>	−0.558	.375
	N_{types}	0.817	.504
	<i>Monophonic</i>	0.452	.568
	N_{tokens}	−0.554	.591

Note. Each predictor appears in the order specified by stepwise selection, with R^2 estimated at each step. However, β presents the standardized betas estimated in the model’s final step.

Table 3. Stepwise regression analysis predicting the average H_m estimated for each composition from the best-performing model configurations with characteristic features of the corpus.

To our knowledge, this is the first language-modeling study to use such a large vocabulary of chord types, though this approach is far more common in the NLP community, where the selected corpus can sometimes contain millions of distinct word types. Our goal in doing so was to bridge the gulf between the most current data-driven methods for melodic and harmonic prediction on the one hand [24], and applications of chord typologies for the creation of corpora using expert analysts on the other [3]. Indeed, despite recent efforts to determine the efficacy of language models for annotated corpora [11, 15], relatively little has been done to develop unsupervised methods for the discovery of tonal harmony in predictive contexts.

One serious limitation of the architectures examined in this study is their unwavering commitment to the surface. Rather than skipping seemingly inconsequential onsets, such as those containing embellishing tones or repetitions, these models predict every onset in their path. As a result, the model configurations examined here attempted to predict tonal (pitch) content rather than tonal harmonic progressions per se. In our view, word class models could provide the necessary bridge between the bottom-up and top-down approaches just described by reducing the vocabulary of surface simultaneities to its most essential harmonies [2]. Along with prediction tasks, these models could then be adapted for sequence generation and automatic harmonic analysis, and in so doing, provide converging evidence that the statistical regularities characterizing a tonal corpus also reflect the *order* in which its constituent harmonies occur.

6. ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement n° 670035).

7. REFERENCES

- [1] J. Albrecht and D. Shanahan. The use of large corpora to train a new type of key-finding algorithm: An improved treatment of the minor mode. *Music Perception*, 31(1):59–67, 2013.
- [2] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based n -gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.
- [3] J. A. Burgoyne, J. Wild, and I. Fujinaga. An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, Miami, USA, 2011.
- [4] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv:1409.1259 [cs, stat]*, September 2014.
- [5] J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, 1984.
- [6] D. Conklin. Representation and discovery of vertical patterns in music. In C. Anagnostopoulou, M. Ferrand, and A. Smaill, editors, *Music and Artificial Intelligence: Lecture Notes in Artificial Intelligence 2445*, volume 2445, pages 32–42. Springer-Verlag, 2002.
- [7] D. Conklin. Multiple viewpoint systems for music classification. *Journal of New Music Research*, 42(1):19–26, 2013.
- [8] D. Conklin and I. H. Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.
- [9] T. J. DiCiccio and B. Efron. Bootstrap confidence intervals. *Statistical Science*, 11(3):189–228, 1996.
- [10] S. Flossmann, W. Goebel, M. Grachten, B. Niedermayer, and G. Widmer. The Magaloff project: An interim report. *Journal of New Music Research*, 39(4):363–377, 2010.
- [11] B. Di Giorgi, S. Dixon, M. Zanoni, and A. Sarti. A data-driven model of tonal chord sequence complexity. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 25(11):2237–2250, 2017.
- [12] T. Hedges and G. A. Wiggins. The prediction of merged attributes with multiple viewpoint systems. *Journal of New Music Research*, 2016.
- [13] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computing*, 9(8):1735–1780, November 1997.
- [14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [15] F. Korzeniowski, D. R. W. Sears, and G. Widmer. A large-scale study of language models for chord prediction. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018.
- [16] G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models. In *Sixth International Conference on Learning Representations (ICLR)*, Vancouver, Canada, April 2018.
- [17] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, Chiba, Japan, 2010.
- [18] D. Müllensiefen and M. Pendzich. Court decisions on music plagiarism and the predictive value of similarity algorithms. *Musicae Scientiae*, Discussion Forum 4B:257–295, 2009.
- [19] M. T. Pearce. *The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition*. Phd thesis, City University, London, 2005.
- [20] M. T. Pearce, D. Conklin, and G. A. Wiggins. *Methods for Combining Statistical Models of Music*, pages 295–312. Springer Verlag, Heidelberg, Germany, 2005.
- [21] M. T. Pearce and G. A. Wiggins. Improved methods for statistical modelling of monophonic music. *Journal of New Music Research*, 33(4):367–385, 2004.
- [22] I. Quinn. Are pitch-class profiles really “key for key”? *Zeitschrift der Gesellschaft der Musiktheorie*, 7:151–163, 2010.
- [23] D. R. W. Sears. *The Classical Cadence as a Closing Schema: Learning, Memory, and Perception*. Phd thesis, McGill University, Montreal, Canada, 2016.
- [24] D. R. W. Sears, M. T. Pearce, W. E. Caplin, and S. McAdams. Simulating melodic and harmonic expectations for tonal cadences using probabilistic models. *Journal of New Music Research*, 47(1):29–52, 2018.
- [25] D. Temperley and D. Sleator. Modeling meter and harmony: A preference-rule approach. *Computer Music Journal*, 23(1):10–27, 1999.
- [26] G. Widmer. Using AI and machine learning to study expressive music performance: Project survey and first report. *AI Communications*, 14(3):149–162, 2001.
- [27] I. H. Witten and T. C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, 1991.

SKELETON PLAYS PIANO: ONLINE GENERATION OF PIANIST BODY MOVEMENTS FROM MIDI PERFORMANCE

Bochen Li¹

Akira Maezawa²

Zhiyao Duan¹

¹ University of Rochester, USA

² Yamaha Corporation, Japan

{bochen.li, zhiyao.duan}@rochester.edu, akira.maezawa@music.yamaha.com

ABSTRACT

Generating expressive body movements of a pianist for a given symbolic sequence of key depressions is important for music interaction, but most existing methods cannot incorporate musical context information and generate movements of body joints that are further away from the fingers such as head and shoulders. This paper addresses such limitations by directly training a deep neural network system to map a MIDI note stream and additional metric structures to a skeleton sequence of a pianist playing a keyboard instrument in an online fashion. Experiments show that (a) incorporation of metric information yields in 4% smaller error, (b) the model is capable of learning the motion behavior of a specific player, and (c) no significant difference between the generated and real human movements is observed by human subjects in 75% of the pieces.

1. INTRODUCTION

Music performance is a multimodal art form. Visual expression is critical for conveying musical expression and ideas to the audience [4,5]. Furthermore, visual expression is critical for communicating musical ideas among musicians in a music ensemble, such as predicting the leader-follower relationship in an ensemble [15].

Despite the importance of body motion in music performance, much work in automatic music performance generation has focused on synthesizing expressive audio data from a corresponding symbolic representation of the music performance (e.g., a MIDI file). We believe that, however, body motion generation is a critical component that opens door to multiple applications. For educational purposes, for example, replicating the visual performance characteristics of well-known musicians can serve as demonstrations for instrument beginners to learn from. Musicologists can apply this framework to analyze the role of gesture and motion in music performance and perception. For entertainment purposes, rendering visual performances along with music audio enables a more immersive music enjoyment experience as in live concerts. For automatic

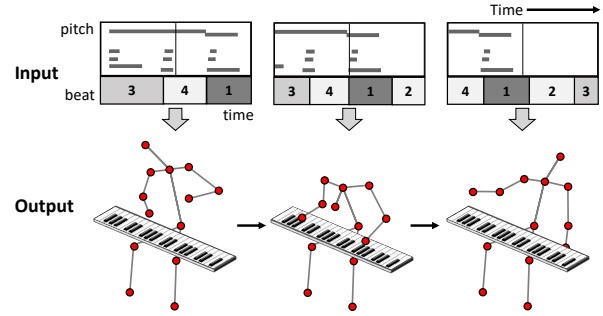


Figure 1. Outline of the proposed system. It generates expressive body movements as skeleton sequences like human playing on a keyboard instrument, given the input of MIDI note stream and metric structure information.

accompaniment systems, appropriate body movements of machine musicians provide visual cues for human musicians to coordinate with, leading to more effective human-computer interaction in music performance settings.

For generating visual music performance, i.e., body position and motion data of a musician, it is important to create an expressive and natural movement of the *whole body* in an online fashion. To consider both expressiveness and naturalness, the challenge is to maintain some common principles in music performance constrained by the musical context being played. Most previous work formulates it as an inverse kinematics problem with physical constraints, where the generated visual performance is limited to hand shapes and finger positions. Unfortunately, this kind of formulation fails to address the two challenges; specifically, (1) it fails to generate the whole body movements that are relevant to music expression, such as the head and body tilt, and (2) it fails to take into account the musical context constraints for generation, which do not contribute to ergonomics.

Therefore, we propose a body movement generation system as outlined in Figure 1. The input is a real-time *MIDI note stream* and a *metric structure*, without any additional indication of phrase structures or expression marks. The MIDI note stream provides the music characteristics and the artistic interpretations, such as note occurrence, speed, and dynamics. The metric structure indicates barlines and beat positions as auxiliary information. Given these the system can automatically generate expressive and natural body movements from any performance data in the



© Bochen Li, Akira Maezawa, Zhiyao Duan. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Bochen Li, Akira Maezawa, Zhiyao Duan. "Skeleton plays piano: online generation of pianist body movements from MIDI performance", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

MIDI format. We design two Convolutional Neural Networks (CNN) to parse the two inputs and then feed the extracted feature representations to a Long Short-Term Memory (LSTM) network to generate proper body movements. The generated body movements are represented as a sequence of positions of the upper body joints¹. The two complementary inputs serve to maintain a correct hand position on the keyboard while conveying musical ideas in the upper body movements. To learn a natural movement, we employ a two-stage training strategy, where the model is trained to learn the joint positions first, then later trained to also learn the body limb lengths.

2. RELATED WORK

There has been work on cross-modal generation, mostly for speech signals tracing back to the 1990s [1], where a person’s lips shown in video frames are warped to match the given phoneme sequence. Given the speech audio, similar work focuses on synthesizing photo-realistic lip movements [14], or landmarks of the whole face [6]. Some other work focuses on the generation of dancers’ body movements [9, 12] and behaviors of animated actors [11].

Similar problem settings for music performances have been rarely studied. When the visual modality is available, the system proposed in [8] explores the correlation between the MIDI score and visual actions, and is able to target the specific player in an ensemble for any given track. Purely from the audio modality, Chen et al. [3] propose to generate images of different instrumentalists in response to different timbres using cross-modal Generative Adversarial Networks (GAN). Regarding the generation of videos, related work generates hand and finger movements of a keyboard player from an MIDI input [17] through inverse kinematics with appropriate constraints. All of the above-mentioned works, however, do not model musicians’ creative body behavior in expressive music performances.

Given the original MIDI score, Widmer et al. [16] propose to predict three expressive dimensions (timing, dynamics, and articulations) on each note event using a Bayesian model trained on a corpus of human interpretations of piano performances. It further gives a comprehensive analysis of computer’s creative ability in generating expressive music performances, and proves that certain aspects of personal styles are identifiable and even learnable from MIDI performances. Regarding to the expressive performance generation in visual modality, Shlizerman et al. [13] propose to generate expressive body skeleton movements and adapt them into textured characters for pianists and violinists. Different from our proposed work, they take the input of audio waveforms rather than MIDI performances. We argue that MIDI data is a more scalable format to carry context information, regardless of recording conditions and piano acoustic characteristics. And most of piano pieces have the sheet music in MIDI format, which can be aligned with a waveform recording.

¹ We do not generate lower body movements as they are often paid less attention by the audience.

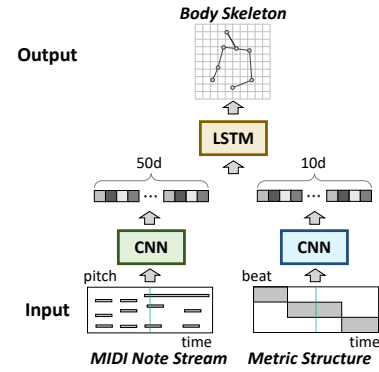


Figure 2. The proposed network structure.

3. METHOD

The goal of our method is to generate a time sequence of body joint coordinates, given a live data stream of note events from the performer’s actions on the keyboard (MIDI note stream), and synchronized metric information. We seek to create the motion at 30 frames-per-second (FPS), a reasonable frame-rate to ensure a perceptually smooth motion. In this section, we introduce the technical details of the proposed method, including the network design and training conditions. We first use two CNN structures to parse the raw input of the MIDI note stream and the metric structure, and feed the extracted feature representations to an LSTM network to generate the body movements, as a sequence of upper-body joint coordinates forming a skeleton. The network structure is shown in Figure 2.

3.1 Feature Extraction by CNN

In contrast to traditional methods, our goal is to model expressive body movements that are associated with the keyboard performance. In this sense, the system should be aware of the general phrases and the metric structure in addition to each individual note event. Instead of designing hand-crafted features, we use CNNs to extract features from the raw input of the MIDI note stream and the metric structure, respectively.

3.1.1 MIDI Note Stream

We convert the MIDI note stream into a series of two-dimensional representations known as the *piano-roll matrix*, and for each of them extract a feature vector ϕ_x as the *piano-roll feature*.

To prepare the piano roll, the MIDI note stream input is sampled at 30 frames-per-second (FPS) to match the target frame rate. This quantizes the time resolution into the unit of 33 ms, as a video frame. Then for each time frame t we define a binary piano-roll matrix $\mathbf{X} \in \mathbb{R}^{128 \times 2\tau}$, where element (m, n) is 1 if there is a key depression action at pitch m (in MIDI note number) and frame $t - \tau + n - 1$, and 0 otherwise. We set $\tau = 30$. The key depression timing is quantized to the closest unit boundary. Note that the sliding window covers both past τ frames and future $\tau - 1$ frames, and the note onset interval in \mathbf{X} captures enough

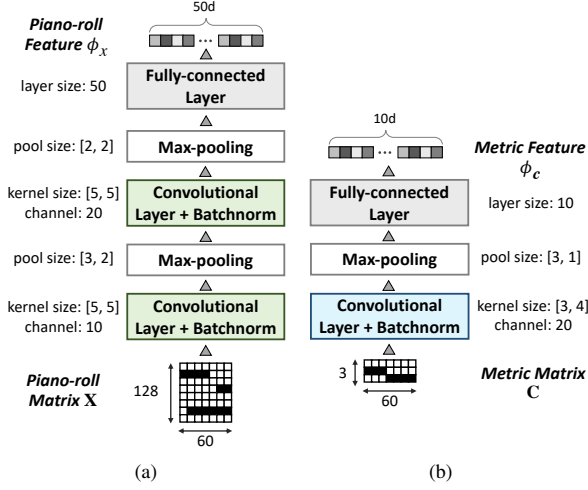


Figure 3. The CNN structures and parameters for feature extraction from the (a) MIDI note stream and (b) metric structure information.

information for motion generation to “schedule” its timing. Looking into the future is necessary for the generation of proper body movements, which is also true for human musicians: to express natural and expressive body movements, a human musician should either look ahead on the sheet music, or be acquainted with it beforehand. Later in Section 3.2 we will introduce in which cases we can avoid the potential delays in real-time applications.

We then use a CNN to extract features from the binary piano-roll matrix \mathbf{X} , as CNNs are capable of capturing local context information. The design of our CNN structure is illustrated in Figure 3.a. The input is the piano-roll matrix \mathbf{X} and the output is a 50-d feature vector ϕ_x as the piano-roll feature. There are two convolutional layers followed by max-pooling layers, and we use leaky rectified linear units (ReLU) for activations. The kernel spans 5 semitones and 5 time steps, assuming that the whole body movement is not sensitive to detailed note occurrence. Overall, it is thought that in addition to generating expressive body movements, the MIDI note stream constrains the hand positions on the keyboard.

3.1.2 Metric Structure

Since the body movements are likely to correlate with the musical beats, we also input the metric structure to the proposed system to obtain another feature vector. This metric structure indexes beats within each measure, which is not encoded in the MIDI note stream. The metric structure can be obtained by aligning the live MIDI note stream with the corresponding symbolic music score with explicitly-annotated beat indices and downbeat positions.

Similar to the MIDI note stream feature, we sample them with the same FPS and window length, and, at each frame t , define the metric information as a binary *metric matrix* $\mathbf{C} \in \mathbb{R}^{M \times 2\tau}$, with $M = 3$. Here, element (m, n) is a one-hot encoding of the metric information at frame $t - \tau + n - 1$, where the three rows correspond to down-

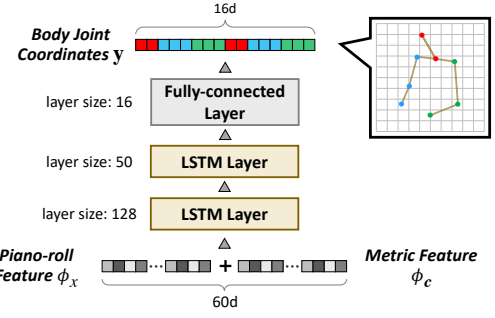


Figure 4. The LSTM network structure for body movement generation.

beats, pick-up beats, and other positions, respectively. We then build another CNN to parse the metric matrix \mathbf{C} and obtain a 10-d output vector ϕ_c as the *metric feature*, as illustrated in Figure 3.b.

3.2 Skeleton Movement Generation by LSTM

To generate the skeleton sequence, we apply the LSTM network, which is capable of preserving the temporal coherence of the output skeleton sequence while learning the pose characteristics associated with the MIDI input. The input to the LSTM is a concatenation of the piano-roll feature ϕ_x and the metric feature ϕ_c , and the output is the normalized coordinates of the body joints \mathbf{y} . Since musical expression of a human pianist is mainly reflected through upper body movements, we model the x - and y - visual coordinates of K joints in the upper body as $\mathbf{y} = \langle y_1, y_2, \dots, y_{2K} \rangle$, where K is 8 in this work, corresponding to nose, neck, both shoulders, both elbows, and both wrists. The first K indices denote the x -coordinates and the remaining denote the y -coordinates. Note that all the coordinate data in \mathbf{y} , for each piece, are shifted such that the average centroid is at the origin, and scaled isotropically such that the average variance along x - and y -axis sums to 1. The network structure is illustrated in Figure 4. It has two LSTM layers, and the output layer is fully-connected to get the 16-d vector approximating \mathbf{y} for the current frame. The output skeleton coordinates are temporally smoothed using a 5-frame moving window. We denote the predicted body joint coordinates, given \mathbf{X} , \mathbf{C} and network parameters θ , as $\hat{\mathbf{y}}(\mathbf{X}, \mathbf{C}|\theta)$.

Since the LSTM is unidirectional, the system is capable of generating motion data in an online manner, with a latency of 30 frames (i.e., 1 second). However, feeding the pre-existing reference music score (after aligned to the live MIDI note stream online) to the system enables an anticipation mechanism like human musicians, which makes it applicable in real-time scenarios without the delay.

3.3 Training Condition

To train the model, we minimize, over θ , the sum of a loss function $J(\mathbf{y}, \mathbf{C}, \mathbf{X}, \theta)$ evaluated over the entire training dataset. The loss function expresses a measure of discrepancy between the predicted body joint coordinates $\hat{\mathbf{y}}$ and

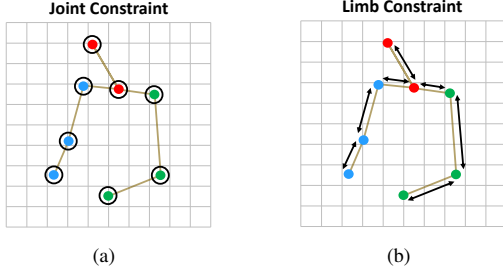


Figure 5. The two constraints applied during training.

the ground-truth coordinates \mathbf{y} .

We use different loss functions during the course of training. In the first 30 epochs, we simply minimize the Manhattan distance between the estimated and the ground-truth body joint coordinates with weight decay:

$$J(\mathbf{y}, \mathbf{C}, \mathbf{X}, \theta) = \sum_k |\hat{y}_k(\mathbf{X}, \mathbf{C}|\theta) - y_k| + \beta \|\theta\|^2, \quad (1)$$

where k is the index for the body joints and $\beta = 10^{-8}$ is a weight parameter. We call this kind of loss the *body joint constraint* (see Figure 5.a). After 30 training epochs, we add another loss to ensure that not only the coordinates are correct but also consistent with the expected limb lengths:

$$J(\mathbf{y}, \mathbf{C}, \mathbf{X}, \theta) = \sum_k |\hat{y}_k(\mathbf{X}, \mathbf{C}|\theta) - y_k| + \sum_{(i,j) \in E} |\hat{z}_{ij}(\mathbf{X}, \mathbf{C}|\theta) - z_{ij}| + \beta \|\theta\|^2, \quad (2)$$

where $z_{ij} = (y_i - y_j) + (y_{K+i} - y_{K+j})$ is the displacement between two joints i and j on a limb (e.g., elbow-wrist), $E = \{(i, j)\}$ is the set of possible limb connections (i, j) of a human body. We call the added term the *body limb constraint* (see Figure 5.b). This is similar to the geometric constraint as described in [10]. There are 7 limb connections in total, given the 8 upper body joints. We then train another 120 epochs using the limb constraint. We use the Adam [7] optimizer, which is a stochastic gradient descent method, to minimize the loss function.

Here we propose to combine the two kinds of constraints in our training epochs. The body limb constraints are important because the loss of joint positions are minimized *independently of each other* in the body joint constraint. Figure 6 demonstrates several generated skeleton samples on the normalized plane, where the limb constraint is not applied in the following 120 epochs. Limb constraint adds dependencies between the loss among different joints, encouraging the model to learn a natural movement that considers the consistency of limb lengths. We only use this constraint at later epochs, however, because the body joint constraint is an easier optimization problem; if we optimize with body limb constraints from the very beginning, the training sometimes fails and remains a state of what seems a local optima, perhaps because the loss function wants to minimize the body joint errors but the gradient must pass through regions where the

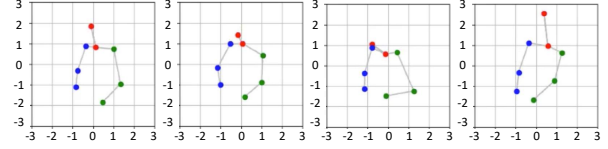


Figure 6. Several generated unnatural skeleton samples without the limb constraint.

limb constraint increases. In this case, the arrangements of the body joints tend to be arbitrary and not ergonomically reasonable.

4. EXPERIMENTS

We perform objective evaluations to measure the accuracy of the generated movements, and subjective evaluations to rate their expressiveness and naturalness.

4.1 Dataset

As there is no existing dataset for the proposed task, we recorded a new audio-visual piano performance dataset with synchronized MIDI stream information on a MIDI keyboard. The dataset contains a total of 74 performance recordings (3 hours and 8 minutes) of 16 different tracks (8 piano duets) played by two pianists, one male and one female. The two players were respectively assigned the primo and the secondo parts of 8 piano duets. Each player then played the 8 tracks multiple times (1-7 times) to render different expressive styles, e.g., normal, exaggerated, etc. At each time the primo and secondo are recorded together to ensure enough visual expressiveness on the players for interactions. The key depression information (pitch, timing, and velocity) is automatically encoded into the MIDI format by the MIDI keyboard. For each recording, the quantized beat number and the downbeat positions were annotated by semi-automatically aligning the MIDI stream and the corresponding MIDI score data. The camera was placed on the left-front side of the player and the perspective was fixed throughout all of the performances. The video frame rate was 30 FPS. The 2D skeleton coordinates were extracted from the video using a method based on OpenPose [2]. The video stream and the MIDI stream of each recording were manually time-shifted to align with the key depression actions. Note that we extract the 2D body skeleton data purely from computer vision techniques instead of capturing 3D data using motion sensors, which makes it possible to use the massive online video recordings of great pianists (e.g., Lang Lang) to train the system.

4.2 Objective Evaluations

We conduct two experiments to assess our method. Since there is no similar previous work to model the players' whole body pose from MIDI input, we set different experimental conditions for the proposed model as baselines and compare them. First, we investigate the effect of incorporating the metric structure information, which is likely to be relevant for expressive motion generation but does not directly affect the players' key depression actions on

the keyboard. Second, we compare the performance of the network when training on a specific player versus training on multiple players. To numerically evaluate the quality of the system output, we use the mean absolute error (MAE) between the generated and the ground-truth skeleton coordinates at each frame.

4.2.1 Effectiveness of the Metric Structure

The system takes as the inputs the MIDI note stream and the metric information. Here we investigate if the latter one can help in the motion generation process, by setting a baseline system that takes the MIDI note stream as the input, ignoring the metric structure by fixing ϕ_c to 0. We evaluate the MAE of the two models, using piece-wise leave-one-out testing over all the 16 tracks.

Results show that adding the metric structure information into the network can decrease the MAE from **0.180** to **0.173**. The unit is in the scale of the normalized plane, where the length of an arm-wrist limb is around 1.2 (see Figure 6). The result is significant because it not only demonstrates that our proposed method can effectively model the metric structure, but also that features that are not indirectly related to physical placement of the hand *does* have an effect on expressive body movements. Although our dataset for evaluation is small, we argue that overfit should not exist since the pieces are quite different.

On the other hand, we also observe that even without the metric structure information, the system output is still reasonable by learning the music context from the MIDI note stream. This setting broadens the use scenarios of the proposed system, such as when the MIDI note stream is from an improvised performance without corresponding metric structure information. Nevertheless, including a reference music score is beneficial for the system not only because it improves the MAE measure, but it also enables an anticipation mechanism to favor real-time generation without potential delays.

4.2.2 Training on A Specific Player

In this experiment, we evaluate the model's performance when fixing the same player for training and testing. Now the experiments are carried out on the two players separately. We first divide the dataset into two subsets, each obtaining the 8 different tracks performed by the two players respectively. On each subset we use the leave-one-out testing for the 8 tracks and calculate the MAE between the generated and ground-truth coordinates of body skeletons. The average of the MAE from the two subsets is **0.170**. Comparing the MAE of 0.173 in Section 4.2.1 and the MAE of 0.170 in this experiment, we see that training on a generic model only on a target player is slightly better than training over different players. This slight improvement may not be statistically significant. The marginal difference also suggests that even when trained on multiple players as in Section 4.2.1, the system is capable of remembering the motion characteristic of each player.

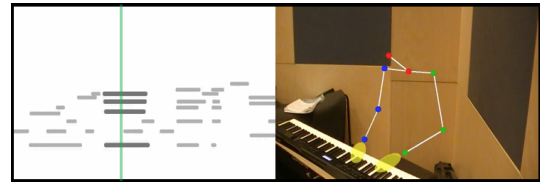


Figure 7. One sample frame of the assembled video for subjective evaluation.

4.3 Subjective Evaluation

Although the objective evaluation using MAE reflects the system's capability of reproducing the players' body movements on a new MIDI performance stream, this measure is still limited. There can be multiple creative ways on body motions to expressively interpret the same music, and the ground-truth body motion is just one possibility. In addition, from MAE we cannot infer the naturalness of the generated body movements, which is even more important than simply learning to reproduce the motion. In this section, we conduct subjective tests to evaluate the quality of the generated body movements, addressing both expressiveness and naturalness. The strategy is to mix the ground-truth body movements with the generated ones and let the testers to tell if each sample is real (ground-truth from human) or fake (generated).

4.3.1 Arrangements

In the subjective evaluation, we mix the two players together and cross-validate on the 16 tracks, as in Section 4.2.1. Here we do not add the metric structure input because positive feedbacks on the generation results purely from the keyboard actions will promise broader use cases of the system, i.e., improvised performance without a reference music score.

From the generated skeleton coordinates, we recover them to the original pixel positions on real video frames using the same scaling factor when normalizing the ground-truth skeleton before training. Then we generate an animation showing body joints as circles and limb connections as straight lines on the background environment image taken by the camera from the same perspective. In the same generated video, we also render a dynamic piano-roll that covers a rolling 5-second segment around the current time frame together with the synthesized audio. For a fair comparison, instead of using the original video recordings of real human performances, we generate human body skeletons by repeating the same process using the ground-truth skeletal data. Figure 7 shows one sample frame of the assembled video as a visualization.

We arrange 16 pairs of the generated and ground-truth skeleton motions on all the 16 tracks, and randomly crop a 10-second excerpt from each one (excluding several chunks containing long silence parts or page turning motions). This results in 32 video excerpts. We shuffle the 32 excerpts before showing them to subjects for evaluation.

We recruit 18 subjects from Yamaha employees, who are in their 20's to 50's, all with rich experience in musical

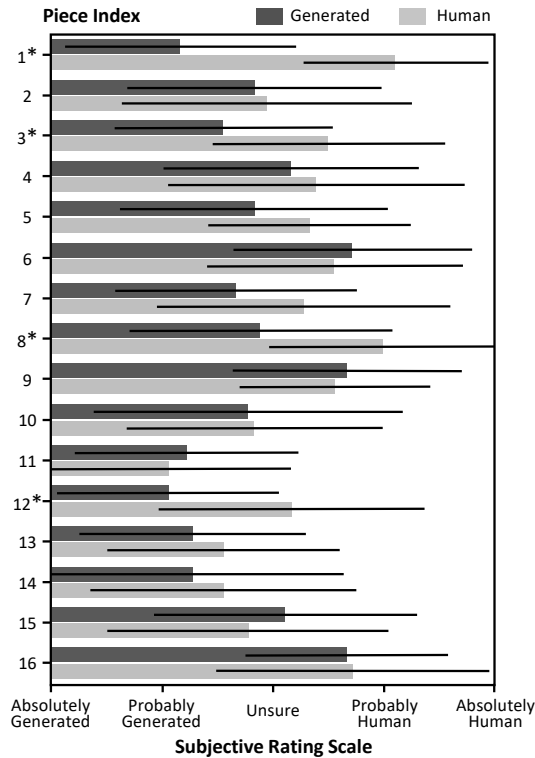


Figure 8. Subjective evaluation on expressiveness and naturalness of the generated and human skeleton performance videos. The tracks with significant different ratings are marked with “*”.

acoustics or music audio signal processing. 17 subjects have instrument performance experiences (15 on keyboard instruments). This guarantees that most of them have a general knowledge of how a human pianist performance may look like based on a given MIDI stream, considering different factors such as hand positions on the keyboard according to pitch height, dominant motions for leading onsets, etc. Based on expressiveness and naturalness they rated the videos on a 5-point scale: absolutely generated (1), probably generated (2), unsure (3), probably human (4), and absolutely human (5).

4.3.2 Results

Figure 8 shows the average subjective ratings as bar plots and their standard deviations as whiskers. A Wilcoxon signed rank test on each piece shows that no significant difference is found in 12 out of the 16 pairs ($p = 0.05$). This suggests that for 3/4 of the observation videos, the generated body movements achieve the same level of expressiveness and naturalness as the real human videos.

In Figure 8, the pieces with significant differences in the subjective ratings between generated and real human videos are marked with “*”. On the 1st piece, we observe an especially significant difference. Further investigation reveals that this piece is in a fast tempo (130 BPM), where the eighth notes are alternatively played by the right and left hand with an agile motion, as shown in Figure 9.a. The generated performance lacks this kind of dexterity. In

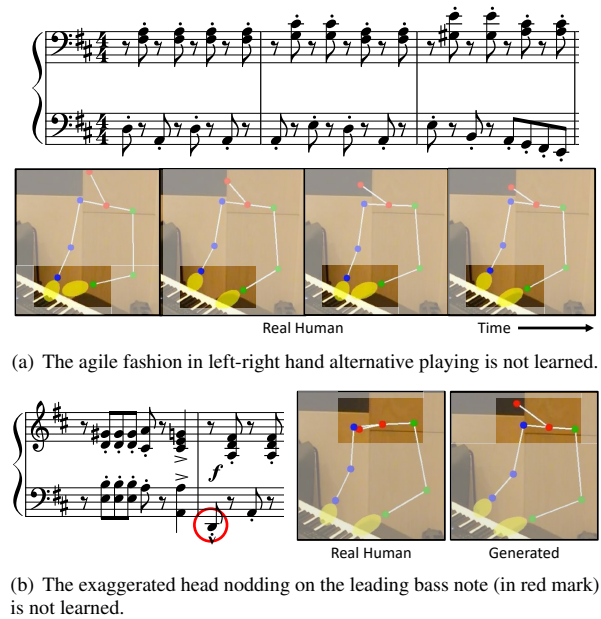


Figure 9. The two typical failure cases.

addition, the physical body motions from the human players are distinct and exaggerated around the phrase boundaries, but the generated ones tend to create more conservative motions. Figure 9.b gives an example, where in the real human’s performance the head moves forward extensively on the leading bass note (marked in red), whereas the generated one does not. Another observed drawback is the improper wrist positioning of a resting hand; a random position is often predicted in these cases. This is because the left/right hand information is not encoded in the MIDI file, and when only one hand is used, the system does not know which hand to use and how to position the other hand. Generally speaking, the generated movements that are rated significantly lower than real human movements tend to be somewhat dull, which might provide the subjects a cue to discriminate between human and generated movements. We present all of the generated videos online².

5. CONCLUSION

In this paper, we proposed a system for generating a skeleton sequence that corresponds to an input MIDI note stream. Thanks to data-driven learning between the MIDI note stream and the skeleton, the system is capable of generating natural playing motions like a human player with no explicit constraints on the physique or fingering, reflecting musical expressions, and attuning the generated motion to a particular performer.

For future work, we will apply more music contextual features to generate richer skeleton movements, and extend our method to the generation of 3D joint coordinates. Generating textured characters based on these skeletons is another future direction.

²<http://www.ece.rochester.edu/projects/air/projects/skeletonpianist.html>

6. ACKNOWLEDGEMENT

This work is partially supported by the National Science Foundation grant 1741472.

7. REFERENCES

- [1] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video rewrite: Driving visual speech with audio. In *Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques*, pages 353–360, 1997.
- [2] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [3] Lele Chen, Sudhanshu Srivastava, Zhiyao Duan, and Chenliang Xu. Deep cross-modal audio-visual generation. In *Proceedings of the ACM International Conference on Multimedia Thematic Workshops*, pages 349–357, 2017.
- [4] Sofia Dahl and Anders Friberg. Visual perception of expressiveness in musicians body movements. *Music Perception: An Interdisciplinary Journal*, 24(5):433–454, 2007.
- [5] Jane W Davidson. Visual perception of performance manner in the movements of solo musicians. *Psychology of Music*, 21(2):103–113, 1993.
- [6] Sefik Emre Eskimez, Ross K Maddox, Chenliang Xu, and Zhiyao Duan. Generating talking face landmarks from speech. In *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation (LVA-ICA)*, 2018.
- [7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–5, 2015.
- [8] Bochen Li, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma. See and listen: Score-informed association of sound tracks to players in chamber music performance videos. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017.
- [9] Zimo Li, Yi Zhou, Shuangjiu Xiao, Chong He, and Hao Li. Auto-conditioned recurrent networks for extended complex human motion synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.
- [10] Guanghan Ning, Zhi Zhang, and Zhiquan He. Knowledge-guided deep fractal neural networks for human pose estimation. *IEEE Transactions on Multimedia*, 20(5):1246–1259, 2017.
- [11] Ken Perlin and Athomas Goldberg. Improv: A system for scripting interactive actors in virtual worlds. In *Proceedings of the ACM Annual Conference on Computer Graphics and Interactive Techniques*, pages 205–216, 1996.
- [12] Ju-Hwan Seo, Jeong-Yean Yang, Jaewoo Kim, and Dong-Soo Kwon. Autonomous humanoid robot dance generation system based on real-time music input. In *Proceedings of the IEEE International Conference on Robot and Human Interactive Communication*, pages 204–209, 2013.
- [13] Eli Shlizerman, Lucio Dery, Hayden Schoen, and Ira Kemelmacher-Shlizerman. Audio to body dynamics. 2017. Available: <https://arxiv.org/pdf/1712.09382.pdf>.
- [14] Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Synthesizing obama: learning lip sync from audio. *ACM Transactions on Graphics (TOG)*, 36(4), 2017.
- [15] Chia-Jung Tsay. The vision heuristic: Judging music ensembles by sight alone. *Organizational Behavior and Human Decision Processes*, 124(1):24–33, 2014.
- [16] Gerhard Widmer, Sebastian Flossmann, and Maarten Grachten. YQX plays chopin. *AI magazine*, 30(3):35–48, 2009.
- [17] Kazuki Yamamoto, Etsuko Ueda, Tsuyoshi Suenaga, Kentaro Takemura, Jun Takamatsu, and Tsukasa Ogasawara. Generating natural hand motion in playing a piano. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3513–3518, 2010.

TOWARDS FULL-PIPELINE HANDWRITTEN OMR WITH MUSICAL SYMBOL DETECTION BY U-NETS

Jan Hajič jr.¹

Matthias Dorfer²

Gerhard Widmer²

Pavel Pecina¹

¹ Institute of Formal and Applied Linguistics, Charles University

² Institute of Computational Perception, Johannes Kepler University

hajicj@ufal.mff.cuni.cz

ABSTRACT

Detecting music notation symbols is the most immediate unsolved subproblem in Optical Music Recognition for musical manuscripts. We show that a U-Net architecture for semantic segmentation combined with a trivial detector already establishes a high baseline for this task, and we propose tricks that further improve detection performance: training against convex hulls of symbol masks, and multichannel output models that enable feature sharing for semantically related symbols. The latter is helpful especially for clefs, which have severe impacts on the overall OMR result. We then integrate the networks into an OMR pipeline by applying a subsequent notation assembly stage, establishing a new baseline result for pitch inference in handwritten music at an f-score of 0.81. Given the automatically inferred pitches we run retrieval experiments on handwritten scores, providing first empirical evidence that utilizing the powerful image processing models brings content-based search in large musical manuscript archives within reach.

1. INTRODUCTION

Optical Music Recognition (OMR), the field of automatically reading music notation from images, has long held the significant promise for music information retrieval of making a great diversity of music available for further processing. More compositions have probably been written than recorded, and more have remained in manuscript form rather than being typeset; this is not restricted to the tens of thousands of manuscripts from before the age of recordings, but holds also for contemporary music, where many manuscripts have been left unperformed for reasons unrelated to their musical quality. Making the content of such manuscript collections accessible digitally and searchable is one of the long-held promises of OMR, and at the same time OMR is reported to be the bottleneck there [17]. On printed music or simpler early music notation, this has been attempted by the PROBADO

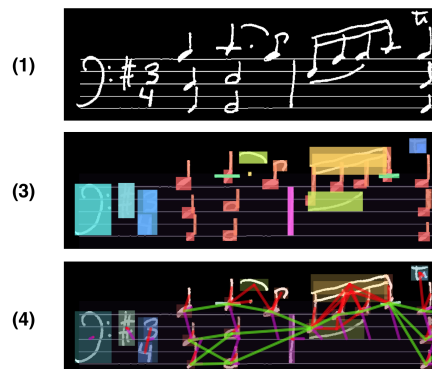


Figure 1. OMR pipeline in this work. Top-down: (1) input score, (3) symbol detection output, (4) notation assembly output. Obtaining MIDI from output of notation assembly stage (for evaluating pitch accuracy and retrieval performance) is then deterministic. Our work focuses on the symbol detection step (1) → (3); notation reconstruction is done only with a simple baseline.

[17, 28] or SIMSSA/Liber Usualis [3] projects. However, for manuscripts, results are not forthcoming.

The usual approach to OMR is to break down the problem into a four-step pipeline: (1) preprocessing and binarization, (2) staffline removal, (3) symbol detection (localization and classification), and (4) notation reconstruction [2]. Once stage (4) is done, the musical content — pitch, duration, and onsets — can be inferred, and the score itself can be encoded in a digital format such as MIDI, MEI¹ or MusicXML. We term OMR systems based on explicitly modeling these stages *Full-Pipeline OMR*.

Binarization and staff removal have been successfully tackled with convolutional neural networks (CNNs) [4, 11], formulated as semantic segmentation. Symbol classification achieves good results as well [12, 13, 33]. However, detecting the symbols on a full page remains the next major bottleneck for handwritten OMR. As CNNs have not been applied to this task yet, they are a natural choice.

Full-Pipeline OMR is not necessarily the only viable approach: recently, *end-to-end OMR* systems have been proposed. [16, 24]. However, they have so far been limited to short excerpts of monophonic music, and it is not clear how to generalize their output design from MIDI equivalents to



© Jan Hajič jr., Matthias Dorfer, Gerhard Widmer, Pavel Pecina. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Jan Hajič jr., Matthias Dorfer, Gerhard Widmer, Pavel Pecina. “Towards Full-Pipeline Handwritten OMR with Musical Symbol Detection by U-Nets”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ <http://music-encoding.org/>

lossless structured encoding such as MEI or MusicXML, so full-pipeline approaches remain justified.

Our work mainly addresses step (3) of the pipeline, applied in the context of a baseline full-pipeline system, as depicted in Fig. 1. We skip stage (2): we treat stafflines as any other object, since we jointly segment and classify and do not therefore have to remove them in order to obtain a more reasonable pre-segmentation. We claim the following contributions:

(1) U-Nets used for musical symbol detection. Applying fully convolutional networks, specifically the U-Net architecture [38], for musical symbol segmentation and classification, without the need for staffline removal. We apply improvements in the training setup that help overcome OMR-specific issues. The results in Sec. 5 show that the improvements one expects from deep learning in computer vision are indeed present.

(2) Full-Pipeline Handwritten OMR Baseline for Pitch Accuracy and Retrieval. We combine our stage (3) symbol detection results with a baseline stage (4) system for notation assembly and pitch inference. This OMR system already achieves promising pitch-based retrieval results on handwritten music notation; to the best of our knowledge, its pitch inference f-score of 0.81 is the first reported result of its kind, and it is the first published full-pipeline OMR system to demonstrably perform a useful task well on handwritten music.

2. RELATED WORK

U-Nets. U-Nets [38] are fully convolutional networks shaped like an autoencoder that introduce skip-connections between corresponding layers of the downsampling and upsampling halves of the model (see Fig. 2). For each pixel, they output a probability of belonging to a specific class. U-Nets are meant for semantic segmentation, not instance segmentation/object detection, which means that they require an ad-hoc detector on top of the pixel-wise output. On the other hand, this formulation avoids domain-specific hyperparameters such as choosing R-CNN anchor box sizes, is agnostic towards the shapes of the objects we are looking for, and does not assume any implicit priors on their sizes. This promises that the same hyperparameter settings can be used for all the visually disparate classes (the one neuralgic point being the choice of receptive field size). Furthermore, U-Nets process the entire image in a single shot — which is a considerable advantage, as music notation often contains upwards of 500 symbols on a single page. A disadvantage of U-Nets (as well as most CNNs) is their sensitivity to the training data distribution, including the digital imaging process. Because of the variability of musical manuscripts, it is likely real-world applications will require case-specific training data, and data augmentation would therefore be used to mitigate this sensitivity; fortunately, fully convolutional networks are known to respond well to data augmentation over sheet music [30] as well as over other application scenarios [9, 23]. Therefore, we consider this choice reasonable, at the very least to establish a strong baseline for handwritten musical symbol

detection with deep learning.

Object Detection CNNs. A standard architecture for object detection is the Regional CNN (R-CNN) family, most notably Faster R-CNN [40] and Mask R-CNN [26]). These networks output probabilities of an object’s presence in each one of a pre-defined sets of anchor boxes, and make the bounding box predictions more accurate with regression. In comparison, the U-Net architecture may have an advantage in dealing with musical symbols that have significantly varying extents, such as beams or stems, as it does not require specifying the appropriate anchor box sizes, and it is significantly faster, requiring only one pass of the network (the detector then requires one connected component search). Furthermore, Faster R-CNN does not output pixel masks, which are useful for archival- and musicology-oriented applications downstream of OMR, such as handwriting-based authorship attribution. Mask R-CNN, admittedly, does not have this limitation, but still requires the same bounding box setup.

Another option is the YOLO architecture [25], specifically the most recent version YOLOv3 [36], which predicts bounding boxes and confidence degrees without the need to specify anchor boxes. A similar approach was proposed in [22], achieving a notehead detection f-score of 0.97, but only with a post-filtering step.

Convolutional Networks in OMR. Convolutional networks have been applied in OMR to symbol classification [33], indicating that they can in principle handle the variability of music notation symbols, but not yet in also finding the symbols on the page. Fully convolutional networks have been successfully applied to staff removal [4], and to resolving the document to a background, staff, text, and symbol layers [11]. However, these are semantic segmentation tasks; whereas we need to make decisions about individual symbols. The potential of U-Nets for symbol detection was preliminarily demonstrated on noteheads [22, 31], but compared to other symbol classes, noteheads are “easy targets”, as they look different from other elements, have constant size, and appear only in one pose (as opposed to, e.g., beams).

OMR Symbol Detection. Localizing symbols on the page has been previously addressed with heuristics rather than machine learning, e.g. with projections [8, 18], Kalman Filters [14], Line Adjacency Graphs [37], or other combinations of low-level image features [39]. On handwritten music, due to its variability, more complex heuristics such as the algorithm of [1] that consists of 14 interdependent steps have been applied.

OMR for Content-Based Retrieval. The idea of using imperfect OMR for retrieval is not new, although originally OMR was attempted in the context of transcribing individual scores. In the PROBADO project [17, 28], an off-the-shelf OMR system was applied to printed Common Western Music Notation (CWMN) scores, allowing retrieval and measure-level score following in a database of 1200 printed scores. The Liber Usualis project at SIMSSA is another such project, on square plainchant notation; it operates at a more fine-grained level that allows for ex-

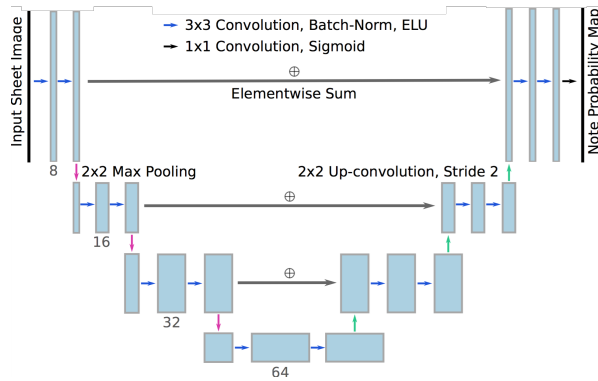


Figure 2. Baseline U-Net model architecture.

ample accurate motif retrieval [3]. However, for CWMN *manuscripts*, we are not aware of similar experiments.

3. MODEL

For all experiments, we use as a basis the same fully convolutional network architecture [38] as shown in Figure 2. There are three down-sampling blocks and three corresponding up-sampling blocks. Each down-sampling block consists of two convolutional layers with batch normalization using the same number of filters; down-sampling is done through 2x2 Max Pooling. After each downsizing step, we use twice the number of filters. The output layer uses sigmoid activation; otherwise, ELU nonlinearity is used. Additionally, we add element-wise-sum residual connections between symmetric layers of the encoder and decoder part of the network.

In the rest of this section, we propose modifications for both architecture and training strategy for symbol detection in handwritten sheet music.

3.1 Convex Hull Segmentation Targets

Our first proposal is to use the convex hull region of individual symbols as a target for training instead of the original segmentation masks. Figure 3 shows an example of the modified training targets. This simple adaptation is an elegant way of dealing with symbols such as f-clefs or c-clefs, which by definition consist of multiple components. As we employ a connected components detector for recognizing the symbols in our experiments in Section 4 we circumvent the need for treating these symbol classes in any special way. This advantage also holds “pre-emptively” for complex symbols which for example contain “holes” and might break up into multiple components after imperfect automatic segmentation, or may be disconnected due to handwriting style (e.g., flats).

3.2 Multichannel Training

Our second proposal is to train multichannel U-Nets predicting the segmentation simultaneously for multiple symbol classes. This design choice has two advantages over

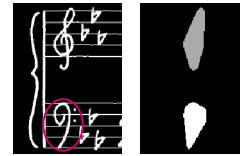


Figure 3. Training on convex hulls circumvents detection problems for symbols consisting of multiple connected components (see f-clef).

training separate detectors for each class. Firstly, at run-time we can predict the segmentation for multiple symbols with a single forward pass of the network. Furthermore, by simultaneously training on multiple symbols at the same time, we allow the model to share low-level feature maps for a certain symbol group (i.e., noteheads, beams and stems), and on the other hand force the model to learn upper-layer features that discriminate well between the various symbols, which – because the capacity of the model stays fixed, and the output layer only uses 1x1 convolutions – could lead to more descriptive representations of the image. In other words, due to the strong correlations across classes induced by music notation syntax, whatever features are learned for one output channel will at the same time be relevant for a different channel; the 1x1 convolution will simply weigh them differently.

However, this setup presents an optimization problem due to imbalanced classes: both in terms of how many foreground pixels there are (i.e. beams vs. duration dots), and with respect to how often they occur on an “average” page of sheet music (noteheads vs. clefs). We address the first issue by splitting the multichannel model into groups of symbols with roughly similar amount of foreground pixels across the dataset. To overcome the second issue, as the training setup operates on randomly chosen windows of the input image (see Sec. 4), we use oversampling: when drawing the random window when a training batch is being built, we check whether the window contains at least one pixel of the target class, and we retry up to five times if there is none. If no target class pixel is found in five tries, we concede and use the last sampled window, even though no pixel of target class is in it. (As opposed to this oversampling, adjusting the weights of the output channels did not lead to improvements.)

Furthermore, if model capacity becomes a limiting factor, we can opt out of sharing the up-sampling part of the model and keep a separate “decoder” for each output channel. This is a compromise that retains some of the speed, space and feature-sharing advantages, but at the same time does not so severely restrict the capacity of the model.

4. EXPERIMENTAL SETUP

We restrict ourselves to the subset of symbol classes that are necessary for pitch inference and basic duration inference (we currently do not detect tuplets – detecting handwritten digits is straightforward enough, the difficulty with tuplets lies in the notation assembly stage). Already this

selection contains symbols with heterogeneous appearance: constant-size, trivial shape (specifically, noteheads, ledger lines, whole and half rests, duration dots), constant-size, non-trivial shape (clefs, flags, accidentals, quarter-, 8th- and 16th rests), and symbols that have simple shapes, but varying extent (stems, beams, barlines).² We assume binary inputs, not least because large-scale OMR symbol detection ground truth is only available for binary images; however, binarization can be done with the same model.

Dataset. We use the MUSCIMA++ dataset, version 1.0 [20]. This is the only publicly available dataset of handwritten music notation with ground truth for symbol detection at a scale that is feasible for machine learning. The dataset contains over 90 000 manually annotated symbols with pixel masks. We use the designated writer-independent test set from MUSCIMA++.

Training Details. We set the network input size to a 256×512 window and randomly sample crops of this size as training samples. We train all our models using the Adam update rule with an initial learning rate of 0.001 [27] and a batch size of 2 (with the 256×512 input window, this is equivalent to batches of a single 512×512 image of [38]). After there is no update on the validation loss for 25 epochs, we divide the learning rate by 5 and continue training from the previously best model. This procedure is repeated two times.

5. RESULTS

As there is no work to which we can compare directly, we first gather at least related OMR solutions, in order to provide whatever context we can for the reader. Then, we report results for symbol detection, and evaluate it in context of downstream tasks: pitch inference in a baseline full-pipeline OMR scenario, as well as first experiments applying our models in retrieval settings.

5.1 Comparison to Existing Systems

Comparison to existing systems is hard, because there are few symbol detection results reported, and even fewer full-pipeline OMR results. Direct comparison is not possible, as the MUSCIMA++ dataset we use has been released only very recently, and previous OMR pipelines (see Sec. 2) generally do not have publicly available code. Furthermore, earlier literature on OMR rarely provides evaluation scores, most of previous work on OMR has (sensibly) focused on printed music rather than manuscripts, and there are few established evaluation practices in OMR anyway [15, 21]. We do our best to at least gather literature where some results on related tasks are given, in order to provide context for our work.

Pitch accuracy, printed music. In printed music, results for pitch accuracy have been consistently very good, when reported. Already in [32], the GAMUT system is said to correctly recover 96 % of pitches in printed music.

² There are also notation symbols that can have non-trivial shape and varying extent, such as slurs or hairpins; however, these are not required for neither pitch, nor duration inference, and we therefore leave them out.

The complex fuzzy system of [39] achieves near-perfect pitch accuracy (98.7 %). Similarly, the CANTOR system evaluated in [5] achieves 98 % semantic accuracy — this time, including polyphonic music. On printed square notation, [19] achieves 95 % pitch accuracy. A combination of systems in [42] achieves over 85 % joint pitch and duration accuracy.

Symbol detection, handwritten music. The most extensive evaluation of symbol detection in handwritten music has been carried out in [1]. Using a complex combination of robust heuristics for segmentation and machine learning for classification, they achieve an average symbol detection f-score of 0.75. These results seem ripe to be surpassed with CNNs: in [31], 98 % handwritten note-head detection accuracy has been reported. For staff detection, a similar architecture has been used in [4] with over 97 % pixel-wise f-score, and similar results are available with a ConvNet pixel classification approach for semantic segmentation into background, text, staves, and notation symbols [11]. At the same time, [33] reports symbol *classification* (without localization) accuracy over 98 %, indicating that CNNs are well capable of generalizing over the variety of handwritten musical symbols. However, we are *not* aware of pitch accuracy results reported on handwritten CWMN scores.

OMR for Retrieval. For retrieval, it is even harder to find comparable results, since evaluation metrics for retrieval depend on the test collection, and there is no such established collection for OMR. Using the open-source Audiveris³ OMR software, [7] matches 9803 printed monophonic fragments from *A Dictionary of Musical Themes* to their electronic counterparts, using a comparable DTW alignment that also (mostly) ignores note duration, reporting a top-1 accuracy of 0.44; however, the collection of themes is a difficult one, since it often contains very similar melodies.

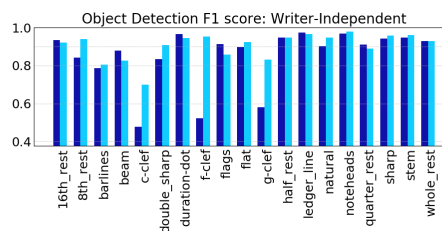


Figure 4. Results for binary segmentation models for individual symbols. Blue: baseline training with mask output; green: training with convex hulls.

5.2 Symbol Detection

We report detection f-scores for the chosen subset of symbols. Aggregating the results is not too meaningful: some rare symbols have an outsized impact on downstream processing (clefs). In Fig. 4, we show the baseline results and compare them to the convex hull setup. Training against

³ <https://github.com/audiveris/audiveris>

Method	c-clef	g-clef	f-clef
single channel – no convex hull	0.48	0.58	0.52
single channel	0.70	0.83	0.95
multi-channel – all	0.16	0.37	0.49
multi-decoder – clefs, oversampling	0.77	0.96	0.93

Table 1. Comparison of detection performance (F-score) of clefs using different segmentation strategies.

convex hulls of objects does address the issue of detecting otherwise disjoint symbols using connected components; otherwise it achieves mixed results.

Compressing the detector with multichannel training without a loss of performance was possible on correlated sub-groups of symbols that bypass the class imbalance problem, such as training together noteheads, stems, beams, and flags; the results worsened when all classes were trained at once. The clefs were most affected by all the changes to the model described in Section 3: improved by convex hull training, neglected when the multichannel model was trained to predict all symbol classes at once, and then drastically improved again when trained as a group with separate decoders and the oversampling strategy. Table 1 summarizes the results for clef detection. Clefs are critical for useful OMR, since they affect the pitch inferred from all subsequent noteheads.

6. APPLICATION SCENARIO: FULL-PIPELINE HANDWRITTEN OMR IN RETRIEVAL

We now explore the utility of the symbol detectors within an OMR pipeline. It is known in OMR that low-level errors can lead to effects on recognition of wildly different magnitudes [15, 35]; in the presence of detection errors, one should therefore see how severely they impact downstream applications. We choose a *retrieval* scenario as the application context for evaluating symbol detection. As opposed to applications where we produce the transcribed score [15, 21, 41], this is straightforward to evaluate.

To verify that our symbol detection approach can yield useful results in an application context, we add a simple notation assembly and pitch inference system on top of the symbol detection results. We choose *retrieval* as the most feasible application of handwritten OMR: there are music manuscript archives with thousands of scores that contain manual copies, and matching them cannot be done without their musical content.

For inferring pitch, we must re-introduce stafflines. However, we can safely assume they have been detected correctly: both [4] and our replication of their experiments with stafflines on this dataset exhibit extremely few errors, and these can be filtered away with a trivial projection heuristic such as that of [18].

6.1 Notation Assembly and Music Inference

Symbol detection alone is not sufficient for decoding musical information: meaningful units are *configurations* of

symbols rather than the symbols themselves [6, 20]. The notation assembly stage is the step where these configurations are recovered (step (4) in the OMR pipeline: see 1). In the MUSCIMA++ dataset, they are represented as an oriented graph; once this graph is recovered, one can perform deterministic pitch inference.⁴

Symbol detection outputs vertices of the notation graph; we therefore need to recover graph edges. Replicating the baseline established in [20], we train a binary classifier over ordered symbol pairs. While this classifier achieves an f-score of 0.92, it makes embarrassing errors: noteheads connected to irrelevant ledger lines in chords, to beams that belong to an entirely different staff, and sometimes to multiple adjacent stems. We discard these obviously wrong edges using straightforward heuristics. We also discard detected objects that are entirely contained within another detected object. The last step is recovering *precedence* edges: we just order rest and noteheads on each staff left-to-right; noteheads connected to the same stem are considered simultaneous, but actual polyphony is ignored.

Once the pitches, durations, and onsets are inferred for the detected noteheads, we then export them as a MIDI file. MIDI is appropriate for retrieval, since it presents straightforward ways of computing similarity. This file then can serve as both the query and the database key for the given score. To compute the similarity of two MIDI files, we align them using Dynamic Time Warping [29] (DTW) over sequences of time frames that contain onsets. The DTW score function for a pair of frames is 1 minus the Dice coefficient of the onset pitch sets in the frames. Then, we match individual pitches within the frame sets that are aligned by DTW and measure the f-score of predicted pitches. DTW is used as the similarity function in [7]; however, we do not reduce polyphonic music to its upper pitch envelope.

6.2 Results

We now report how the full-pipeline baseline on top of the object detection U-Nets predicts pitches, and how it can be used to retrieve related scores.

Pitch accuracy. We use the DTW alignment to directly evaluate pitch classification.⁵ Performing DTW on the inference outputs for page images, we achieve a (micro-)average F-score of only 0.59. Rather than due to errors in symbol detection, this is mostly due to the polyphony de-synchronization effects of bad duration inference; indeed, on (mostly) monophonic music, pitch F-score jumps to 0.78. In order to bypass de-synchronization problems that in fact obscure correct pitch recognition, we split the scores into individual staves (118 in total) and evaluate pitch accuracy on these. The results for the test set staves are reported in Fig. 5. On average, we obtain pitch F-score 0.81, with 0.83 for monophonic staves (and ignoring clef errors, 0.88).

Finally, we evaluate our detector in the context of a retrieval application. We run experiments both on gold-

⁴ A proof-of-concept implementation: <https://github.com/hajicj/muscima>.

⁵ We could evaluate duration classification as well, but due to errors by the notation assembly baseline, this is too low to be worth reporting.

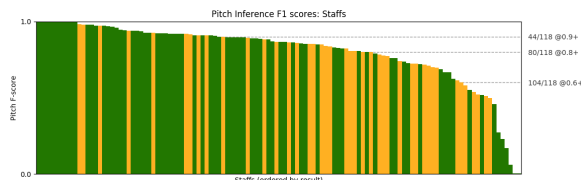


Figure 5. Pitch F-score after DTW alignments on the 118 individual staves in the writer-independent test set, ordered by result. Monophonic staves (darker green) predictably score better than staves with multiple voices or chords (yellow). We found no clear relationship between pitch accuracy and handwriting style.

standard MIDI retrieval and duplicate score retrieval, using the predicted scores; since the similarity metric is pitch f-score, all retrieval experiments work in both directions. Experiments with ground truth MIDI correspond to cross-modal retrieval, where the modalities are a symbolic representation, and the score projected into the MIDI modality using the OMR system; queries with predictions correspond to a simpler scenario where we are querying scores with scores, using the OMR system as a hash function.

Retrieving gold MIDI with scores. Given how small the test set is, retrieving the correct ground truth page — and even staff — should be near-perfect. For staff-to-staff retrieval, $\text{Prec}@1$ is 0.93; for page-to-page and staff-to-page retrieval, this is 1.0, indicating that with our U-Net object detection stage, retrieving gold-standard MIDI using handwritten scores (and vice versa, as the similarity metric is symmetrical) is feasible.

Retrieving scores with scores. The next scenario is to run retrieval not against the ground truth, but against MIDI predicted from different versions of the test set scores. While errors related to differences in handwriting get compounded, the rest of the pipeline imposes consistent limitations on both the database and query recognition outputs and may make the *same* errors on both query and database scores, making the task actually easier. Therefore, we select a *confuse-retrieval* subset of 7 scores from MUSCIMA++ that are as similar to each other as possible: mostly monophonic, and with 0 – 2 sharps. Some of these pieces are musically closely related. For these experiments, our database consists of recognition outputs computed from all *confuse-retrieval* pages in the *training* subset of MUSCIMA++. Queries are taken from predictions on the writer-independent test set: we use both the 7 entire pages and individual staves (34 of those).

The system achieves perfect $\text{Prec}@1$ when pages are used as queries, and 0.94 when using staff queries (2 staff queries did not return the right piece as the top result). The retrieval scores are plotted in Fig. 6. We checked this score also with ground truth queries; this system made only 2 errors as well, but in different queries, which we take as circumstantial evidence that the ground truth MIDI has different issues when matching against a predicted MIDI than a different prediction. When measuring MAP with the cutoff $k=6$ (as there are 7 versions of each page in MUSCIMA++

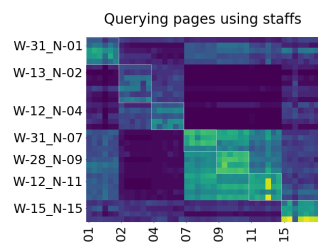


Figure 6. Pitch f-score between predictions on test set staves and (predictions on) training set pages. Notice the pages 07, 09 and 11: these are three movements from J. S. Bach’s Cello suite no. 1, which contain musically highly related material.

and one of them is used for querying), it drops to 0.86.

7. DISCUSSION & CONCLUSIONS

We consider our work a successful step towards enabling applications of hitherto problematic handwritten OMR. The retrieval scenario results are an indication that U-Nets are a workable solution to the handwritten symbol detection bottleneck in the context of full-pipeline OMR. (Here, we must re-state that these results should *not* be interpreted as more than supporting evidence that our object detection method is viable for such scenarios!)

However, U-Nets are still in principle limited by the size of the receptive field: for instance the middle of a long stem looks exactly the same as a barline. We could further leverage syntactic properties of music notation: e.g., the self-attention layer of [34] allows building up the final output from partial recognition results. Fragmenting of long symbols could be overcome with instance segmentation embeddings [10].

To the best of our knowledge, this is also the first time OMR was done with a machine-learning method for notation assembly. We in fact consider this the most interesting line of follow-up work. Recovering the notation graph itself seems like the next bottleneck, especially for duration inference. The non-independent nature of the edges poses an interesting structured prediction challenge, and one could also work towards models that jointly detect symbols and recover their relationships.

Despite their limitations, U-Nets can be used to detect handwritten music notation symbols. They establish a new CNN-based baseline for the object detection task, and we believe the results in pitch inference and a proof-of-concept retrieval scenario indicate that a significant step has been taken towards full-pipeline OMR systems, so that the content of musical manuscripts can become accessible digitally.

8. ACKNOWLEDGMENTS

Jan Hajič jr. and Pavel Pecina acknowledge support by the Czech Science Foundation grant no. P103/12/G084, Charles University Grant Agency grants 1444217 and 170217, and by SVV project 260 453.

9. REFERENCES

- [1] Ana Rebelo. *Robust Optical Recognition of Handwritten Musical Scores based on Domain Knowledge*. PhD thesis, 2012.
- [2] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R. S. Marcal, Carlos Guedes, and Jaime S. Cardoso. Optical Music Recognition: State-of-the-Art and Open Issues. *Int J Multimed Info Retr*, 1(3):173–190, Mar 2012.
- [3] Andrew Hankinson, John Ashley Burgoyne, Gabriel Vigliensoni, Alastair Porter, Jessica Thompson, Wendy Liu, Remi Chiu, and Ichiro Fujinaga. Digital Document Image Retrieval Using Optical Music Recognition. In *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, pages 577–582. FEUP Edições, 2012.
- [4] Antonio-Javier Gallego and Jorge Calvo Zaragoza. Staff-line removal with selectional auto-encoders. *Expert Systems with Applications*, 89:138–148, 2017.
- [5] David Bainbridge. Extensible optical music recognition. page 112, 1997.
- [6] David Bainbridge and Tim Bell. A music notation construction engine for optical music recognition. *Software - Practice and Experience*, 33(2):173–200, 2003.
- [7] Stefan Balke, Sanu Pulimootil Achankunju, and Meinard Müller. Matching Musical Themes based on noisy OCR and OMR input. pages 703–707, 2015.
- [8] P. Bellini, I. Bruno, and P. Nesi. Optical music sheet segmentation. In *Proceedings First International Conference on WEB Delivering of Music. WEDELMUSIC 2001*, pages 183–190. Institute of Electrical & Electronics Engineers (IEEE), 2001.
- [9] Avi Ben Cohen, Idit Diamant, Eyal Klang, Michal Amitai, and Hayit Greenspan. Fully Convolutional Network for Liver Segmentation and Lesions Detection. In Gustavo Carneiro, Diana Mateus, Loïc Peter, Andrew Bradley, João Manuel R. S. Tavares, Vasileios Belagiannis, João Paulo Papa, Jacinto C. Nascimento, Marco Loog, Zhi Lu, Jaime S. Cardoso, and Julien Cornebise, editors, *Deep Learning and Data Labeling for Medical Applications*, pages 77–85, Cham, 2016. Springer International Publishing.
- [10] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic Instance Segmentation with a Discriminative Loss Function. *CoRR*, abs/1708.02551, 2017.
- [11] Jorge Calvo Zaragoza, Gabriel Vigliensoni, and Ichiro Fujinaga. A machine learning framework for the categorization of elements in images of musical documents. In *Third International Conference on Technologies for Music Notation and Representation*, A Coruña, 2017. University of A Coruña.
- [12] Sukalpa Chanda, Debleena Das, Umapada Pal, and Fumitaka Kimura. Offline Hand-Written Musical Symbol Recognition. *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 405–410, sep 2014.
- [13] Cuihong Wen, Jing Zhang, Ana Rebelo, and Fanyong Cheng. A Directed Acyclic Graph-Large Margin Distribution Machine Model for Music Symbol Classification. *PLOS ONE*, 11(3):e0149688, mar 2016.
- [14] V.P. d’Andecy, J. Camillerapp, and I. Leplumey. Kalman filtering for segment detection: application to music scores analysis. In *Proceedings of 12th International Conference on Pattern Recognition*. IEEE Comput. Soc. Press, 1994.
- [15] Donald Byrd and Jakob Grue Simonsen. Towards a Standard Testbed for Optical Music Recognition: Definitions, Metrics, and Page Images. *Journal of New Music Research*, 44(3):169–195, 2015.
- [16] Eelco van der Wel and Karen Ullrich. Optical Music Recognition with Convolutional Sequence-to-Sequence Models. *CoRR*, abs/1707.04877, 2017.
- [17] Christian Fremerey, Meinard Müller, Frank Kurth, and Michael Clausen. Automatic mapping of scanned sheet music to audio recordings. *Proceedings of the International Conference on Music Information Retrieval*, pages 413–418, 2008.
- [18] Ichiro Fujinaga. Optical Music Recognition using Projections. Master’s thesis, 1988.
- [19] Gabriel Vigliensoni, John Ashley Burgoyne, Andrew Hankinson, and Ichiro Fujinaga. Automatic Pitch Detection in Printed Square Notation. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, pages 423–428. University of Miami, 2011.
- [20] Jan Hajič jr. and Pavel Pecina. The MUSCIMA++ Dataset for Handwritten Optical Music Recognition. In *14th International Conference on Document Analysis and Recognition*, pages 39–46, New York, USA, November 2017. Dept. of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University, IEEE Computer Society.
- [21] Jan Hajič jr., Jiří Novotný, Pavel Pecina, and Jaroslav Pokorný. Further Steps towards a Standard Testbed for Optical Music Recognition. In *Proceedings of the 17th International Society for Music Information Retrieval Conference*, pages 157–163, New York, USA, 2016. New York University, New York University.
- [22] Jan Hajič Jr. and Pavel Pecina. Detecting Noteheads in Handwritten Scores with ConvNets and Bounding Box Regression. *CoRR*, abs/1708.01806, 2017.

- [23] Jesus Munoz Bulnes, Carlos Fernandez, Ignacio Parra, David Fernandez Llorca, and Miguel A. Sotelo. Deep fully convolutional networks with random data augmentation for enhanced generalization in road detection. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, oct 2017.
- [24] Jorge Calvo-Zaragoza, Jose J. Valero-Mas, and Antonio Pertusa. End-to-End Optical Music Recognition Using Neural Networks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 472–477, 2017.
- [25] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *CoRR*, abs/1506.02640, 2015.
- [26] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.
- [27] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR) (arXiv:1412.6980)*, 2015.
- [28] F. Kurth, M. Müller, C. Fremerey, Y. Chang, and M. Clausen. Automated synchronization of scanned sheet music with audio recordings. *Proc. ISMIR, Vienna, AT*, pages 261–266, 2007.
- [29] Lawrence R. Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice Hall signal processing series. Prentice Hall, 1993.
- [30] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. Learning Audio-Sheet Music Correspondences for Score Identification and Offline Alignment. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 115–122, 2017.
- [31] Matthias Dorfer, jr. Jan Hajič, and Gerhard Widmer. On the Potential of Fully Convolutional Neural Networks for Musical Symbol Detection. In *Proceedings of the 12th IAPR International Workshop on Graphics Recognition*, pages 53–54, New York, USA, 2017. IAPR TC10 (Technical Committee on Graphics Recognition), IEEE Computer Society.
- [32] Michael Droettboom and Ichiro Fujinaga. Symbol-level groundtruthing environment for OMR. *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, pages 497–500, 2004.
- [33] Alexander Pacha and Horst Eidenberger. Towards a Universal Music Symbol Classifier. In *Proceedings of the 12th International Workshop on Graphics Recognition*, 2017.
- [34] N. Parmar, A. Vaswani, J. Uszkoreit, Ł. Kaiser, N. Shazeer, and A. Ku. Image Transformer. *ArXiv e-prints*, February 2018.
- [35] Pierfrancesco Bellini, Ivan Bruno, and Paolo Nesi. Assessing Optical Music Recognition Tools. *Computer Music Journal*, 31(1):68–93, Mar 2007.
- [36] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. Technical report, 2018.
- [37] K. T. Reed and J. R. Parker. Automatic computer recognition of printed music. *Proceedings - International Conference on Pattern Recognition*, 3:803–807, 1996.
- [38] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III*, pages 234–241, Cham, 2015. Springer International Publishing.
- [39] Florence Rossant and Isabelle Bloch. Robust and Adaptive OMR System Including Fuzzy Modeling, Fusion of Musical Rules, and Possible Error Detection. *EURASIP Journal on Advances in Signal Processing*, 2007(1):081541, 2007.
- [40] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 91–99, 2015.
- [41] Mariusz Szwoch. Using MusicXML to Evaluate Accuracy of OMR Systems. *Proceedings of the 5th International Conference on Diagrammatic Representation and Inference*, pages 419–422, 2008.
- [42] Victor Padilla, Alan Marsden, Alex McLean, and Kia Ng. Improving OMR for Digital Music Libraries with Multiple Recognisers and Multiple Sources. *Proceedings of the 1st International Workshop on Digital Libraries for Musicology - DLfM '14*, pages 1–8, 2014.

SEARCHING PAGE-IMAGES OF EARLY MUSIC SCANNED WITH OMR: A SCALABLE SOLUTION USING MINIMAL ABSENT WORDS

Tim Crawford

Goldsmiths, University of
London
t.crawford@gold.ac.uk

Golnaz Badkobeh

Goldsmiths, University of
London
G.Badkobeh@gold.ac.uk

David Lewis

Oxford eResearch Centre
david.lewis@oerc.ox.ac.uk

ABSTRACT

We define three retrieval tasks requiring efficient search of the musical content of a collection of ~32k page-images of 16th-century music to find: duplicates; pages with the same musical content; pages of related music.

The images are subjected to Optical Music Recognition (OMR), introducing inevitable errors. We encode pages as strings of diatonic pitch intervals, ignoring rests, to reduce the effect of such errors. We extract indices comprising lists of two kinds of ‘word’. Approximate matching is done by counting the number of common words between a query page and those in the collection.

The two word-types are (a) normal ngrams and (b) minimal absent words (MAWs). The latter have three important properties for our purpose: they can be built and searched in linear time, the number of MAWs generated tends to be smaller, and they preserve the structure and order of the text, obviating the need for expensive sorting operations.

We show that retrieval performance of MAWs is comparable with ngrams, but with a marked speed improvement. We also show the effect of word length on retrieval. Our results suggest that an index of MAWs of mixed length provides a good method for these tasks which is scalable to larger collections.

1. INTRODUCTION

The historical repertory of Western classical music is increasingly being made publicly available in the form of downloadable (or merely viewable) digital images; these represent pages of the manuscripts or printed books in which they are preserved, and are no different in this respect from other typical online library materials such as texts or maps.

Search facilities within the individual library systems are entirely text-based, usually making use of existing or specially commissioned catalogue data. In a few cases, special viewing interfaces are provided to enhance the user-experience, such as the parallel presentation of multiple part-books on the web-site of the Bayerische Staats-

bibliothek in Munich.¹ However, the data markup necessary to achieve this has to be done by human experts, which is impractical in general for large collections.

Musicologists need to be able to browse such collections and to search for specific musical parallels within them; librarians need similar facilities for cataloguing purposes (e.g. to identify unknown or unattributed items). This in turn demands fast search methods of adequate accuracy as a first step in the research process to reduce the number of items needing to be examined more exhaustively.

With very few exceptions, music libraries offer online images rather than encoded scores. Providing the latter involves transcription, which can either be done manually by experts, a time-consuming and expensive process, or automatically by OMR, which inevitably introduces errors of various kinds. As OMR techniques improve in future, these errors are likely to diminish, but highly unlikely to disappear altogether.

For fast searching, we need to extract indexes from the OMR output which enable fast searching at high recall. This depends on the musical data extracted and encoded in the indexes being carefully selected to suit a given use-case. For efficient search of the indexes we can benefit from recent advances in string- and pattern-matching algorithms developed for use in bioinformatics for DNA and protein analysis.

In this paper we focus on three musicologically-motivated user tasks given a corpus of digital images of 16th-century printed music: finding duplicate images within the collection (called *dupl* below); finding pages containing substantially the same music as in a query page (*same*); and identifying pages which have non-identical but related or closely relevant music content, such as in different sections or voice parts than the query (*relv*).

We briefly review earlier work on musical corpus-building, content-based music searching and indexing in section 2. We describe our test collection, relevant aspects of the OMR process and our music indexing strategy in section 3. In section 4, we describe the retrieval tasks and our search method in more detail and our experiments and their evaluation in sections 5 & 6. In section 7 we discuss some of the main findings leading to the proposals for further work in section 8.



© Tim Crawford, Golnaz Badkobeh, David Lewis.
Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Tim Crawford, Golnaz Badkobeh, David Lewis. “Searching Page-images of Early Music Scanned with OMR: A Scalable Solution Using Minimal Absent Words”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

1. E.g.: <https://stimmbuecher.digitale-sammlungen.de/view?id=bsb00086863>

2. PREVIOUS WORK

Corpora of historical music

For musicologists, the amount of historical material available online has exploded in recent years, in line with the general availability of data of all kinds, including audio and video files of performances. This does not mean that their requirements for study and analysis are yet adequately met. The sub-discipline of computational, or digital, musicology tends to devote a great deal of effort to data-preparation before the powerful tools of MIR, pattern-matching and statistical analysis can be brought to bear. This is because the majority of the data-resources consist of collections of digital images of the source material, rather than files of its musical content. Traditionally, scores, which attempt to represent the overall musical content of the original documents (which is often, as in the case of the music studied in this paper, distributed between multiple part-books), have been made by human experts; this is inevitably a time-consuming and thus expensive process. The translation (automatic or manual) of musical content from documents or their digital-image surrogates into machine-readable ‘texts’ is generally referred to as music encoding. While digital tools such as score-editing programs have made this easier, by enabling export to standard formats such as MusicXML² and MEL³, the process is in general impractically slow for building large collections.

However, there exist some significant and freely available collections of encoded music, such as those maintained by the Center for Computer Assisted Research in the Humanities at Stanford University,⁴ which present a wide range of classical music encoded in a number of formats. These encodings permit a variety of ways of searching the data for musical features which are offered by software packages such as Humdrum⁵ or Music21.⁶

The online offerings of many digital music libraries in classical music are aggregated in the International Music Score Library Project (IMSLP),⁷ adding curated metadata. The resulting meta-collection (almost nine million pages of music) has rapidly become more-or-less indispensable for performers, teachers and students. Searching within IMSLP for most users is done via metadata rather than musical content. An experimental interface for content-based searching, the Peachnote Ngram Viewer,⁸ works on the output of commercial OMR software run over a large part of the collection; while this is subject to the significant amount of errors introduced in the OMR process, it powerfully demonstrates the potential of efficient search over a large collection.

In the current work, just as in Peachnote, we are not immediately concerned with an abstract or generalized notion of musical similarity. Rather, we select an encoding that represents the musical feature we wish to match.

Our aim is to reduce the search space to a manageable number of musical documents which can be compared or analyzed in more detail manually or by a specialized algorithm. For large collections this task can best be achieved by searching indexes rather than full encodings of each document.

Where the musical features extracted from a document can be represented as some kind of ‘text’, there are many ways of generating useful indexes which can be searched far more quickly than full texts. These have been the subject of information retrieval research for almost half a century, and provide the mechanisms enabling the almost instantaneous search familiar to all who use today’s internet. Indexing methods for music – either symbolic or audio – have received less attention, but a number of viable methods have been proposed and/or have found use [1].

For most of the sixteenth and seventeenth centuries, almost all original material comes in the form of separate voice-parts rather than scores. For the purposes of retrieval these can be treated as linear strings of characters depending on the encoding method. There is a vast literature on string-matching, largely motivated by problems from bioinformatics. Some of the resulting, highly-efficient methods have been proposed for music retrieval.

Music retrieval algorithms inspired by bioinformatics

A very recent survey of MIR applications for algorithms developed in bioinformatics research is contained in [2], although this does not include the method adopted in this paper.

The need for pairwise comparison of potentially extremely long, strings representing the structure of molecules such as DNA or proteins, has been addressed by the development of algorithms such as FASTA [3] and its descendants, such as BLAST,⁹ which are in common use for DNA analysis. The latter algorithm has found musical uses in the audio [4] and symbolic [5] domains. BLAST has also found use in recent work on audio cover-song recognition in [6], where the major speedup in retrieval it brought was found to compensate for a slight degradation in retrieval accuracy. Most recently, [7] reports on the application to music of methods originally designed for bioinformatics. These include multiple sequence-alignment methods such as MAFFT [8].

The present work uses a method which is finding increasing acceptance within bioinformatics, but has not, as far as the authors are aware, previously been applied to music: minimal absent words (MAWs). Here we briefly introduce the concept.

A word is an absent word of a sequence if it does not occur in the sequence. An absent word is minimal if all its proper factors occur in the sequence. Absent words are negative information about the sequence. These objects have been extensively studied in combinatorics on words and it is known that although the number of absent words of a sequence is exponential with respect to the size of the sequence, the number of minimal absent words is only linear with respect to the length of the sequence [9].

2. <http://www.musicxml.com>

3. <http://music-encoding.org>

4. <http://www.musedata.org> and <http://kern.ccarh.org>

5. <http://humdrum.ccarh.org>

6. <http://web.mit.edu/music21/>

7. <http://imslp.org>

8. <http://www.peachnote.com>

9. Basic Local Alignment Search Tool; <http://blast.ncbi.nlm.nih.gov/>

Crochemore et al. in [10] presented a linear-time algorithm to compare two documents by considering all their minimal absent words, using a length-weighted index measure.

In recent years, the significance of minimal absent words has been studied in several biological studies. In [11] absent words for four human genomes were computed, and it was shown that intra-species variations in minimal absent words were lower than inter-species variations.

Furthermore, minimal absent words have been exploited for building phylogenies [12], for measuring dissimilarities/similarities between bio-sequences [13] and for many other applications [14].

3. TEST COLLECTION & OMR

The collection consists of 31,721 page-images of 16th-century printed music, which have all been subjected to OMR. The music was scanned from archival microfilms, in which the several individual part-books for a given item (usually four but up to as many as 12), one for each voice, follow in sequence as preserved under their single shelfmark. In almost every case they show two facing pages in a single image; these were each separated by us into two single page-images.

The collection has associated metadata which gives bibliographical information for each book, but not to the level of musical items; so, for example, while the general sequence of musical items in each book is listed, with titles and original composer ascriptions, the locations of items in the part-books is not recorded. Thus, it is in general impossible to associate automatically a page image with the music on it. This provides the motivation for the present work, aimed at designing a finding aid for researchers or librarians wishing to identify similar or related music within the collection.

The OMR tool we use is Aruspix, a program specifically designed for early printed music.¹⁰ While this represents the current state of the art for this repertory [15], recently reported work suggests that significant progress is possible in the near future [16]. However, it is unlikely that 100% accuracy in OMR will ever be consistently achieved for any repertory; for this reason we maintain that fast, error-robust search methods will always be in demand. Aruspix saves its recognized output as MEI (mensural)¹¹ from which we can extract various kinds of musical sequence. (See Fig 1.)

Typical errors made by OMR systems can be of duration (wrong/missing time-signatures; wrong/missing note-values) and of pitch (wrong/missing clefs; wrong/missing key-signatures; wrong/missing accidentals). The vertical location of symbols such as note-heads on the staff is usually recognized securely; this corresponds to diatonic pitch. Changes of clef tend to compound this effect as pitches are affected over a span of notes (usually until the next line of music), so it is helpful to use relative pitches, i.e. intervals. We have found se-

quences of diatonic intervals to be the most useful for our purposes.

We generate a single diatonic-interval string for each page using a simple alphabetic code devised by RISM¹² for rapid searching of musical incipits (See Figure 1). Letters in upper case represent ascending intervals, lower case descending; same note is indicated by a hyphen.[17]

A typical item, opening only:



MEI output from Aruspix (opening only, simplified):

```
<clef line="3" shape="C" />
<mensur sign="C" slash="1" />
<note pname="e" oct="4" dur="brevis" />
<note pname="d" oct="4" dur="semibrevis" lig="recta" />
<note pname="f" oct="4" dur="semibrevis" />
<note pname="e" oct="4" dur="semibrevis" />
<dot ploc="f" oloc="4" />
<note pname="d" oct="4" dur="semiminima" />
<note pname="c" oct="4" dur="semiminima" />
<note pname="d" oct="4" dur="minima" />
<custos pname="c" oct="4" /> [Spurious: Note missing!]
<note pname="e" oct="3" dur="minima" />
<note pname="e" oct="3" dur="minima" />
<note pname="e" oct="4" dur="minima" />
```

(NB Because the clef has been mis-recognized, all pitches are a third too low; also, in line 11, a note has been mis-read as a *custos*.)

Diatonic pitch sequence:

MEI:	e4	d4	f4	e4	d4	c4	d4	e3	e3	e4
Correct:	g4	g4	a4	g4	f4	e4	f4	g4	g3	g4

Diatonic interval sequence:

MEI:	-1	+2	-1	-1	-1	+2	-7	0	+7
Correct:	0	+1	-1	-1	-1	+1	+1	-8	+8

Encoded diatonic interval sequence:

MEI:	a	B	a	a	a	B	f	-	G	
Correct:	-	A	a	a	a	A	A	g	-	G

Figure 1. The (erroneous) MEI output from Aruspix, and the correct encoding, for a typical item (opening only),¹³ and the sequences we derive from it.

4. TASKS AND METHOD

The three tasks we approach are to recognise page-images which: (a) are duplicates (i.e. different shots/scans of the same page); (b) contain substantially the same music (which may be distributed differently across adjacent pages); (c) contain related but not identical music (this may be from a different voice-part, from a different section of the same piece, or from a derivative work).

Task (a) involves finding near-identical matches; however, the OMR output, and hence the indexes we extract, are not necessarily exactly the same, owing to recognition errors or small differences in photographic conditions, etc. For task (b), although in principle the encodings on which we base our searches should be largely identical, we cannot be sure that each page of different editions of a pieces of music has exactly the same content; often, the page layout is different, or the music is distributed over multiple pages in one or other copy. Furthermore, there

10. <http://www.aruspix.net>

11. <http://music-encoding.org/schema/2.1.1/mei-Mensural.rng>

12. *Répertoire Internationale des Sources Musicales*; see <http://www.rism.info/home.html>

13 D. Phinot (c.1510-c.1555), Altus part of 'Virga Jesse floruit', from *Primus liber cum quatuor vocibus : Mottetti del frutto a quarto* (Venice: Gardane, 1539)

may be extraneous material ‘foreign’ to the query page printed on the same page at the beginning or the end of the piece in question.

The ‘related music’ category is best illustrated by example; all of the following were found as high-ranking matches to the query page (2a) using our methods despite the fact that they tend to diverge after a statement of the opening motif:



Figure 2. Examples of music ‘related’ to a query (a). (N.B. These matches were based on full pages of music, not just on the incipits displayed here.)

Our query page (2a) was the Superius part of ‘D’amours me plains’, a *chanson* by Maistre Rogier.¹⁴ The following item in the same book is a *replicque*, or response, to the *chanson*, with a different text, by Tylman Susato, based on the same musical motifs; this was ranked second. Another piece based on Rogier’s *chanson*, this time with the same text, by Larcier was in fact ranked first. The third-ranked item was the ‘Agnus Dei’ from Thomas Cricquillon’s parody mass on the song, *Missa Damours me plains*; the ‘Sanctus’ from the same mass was ranked in fourth place.

Further examples of ‘related’ music might include separate sections of a work, or arrangements with completely different texts which were catalogued as separate items. In fact, in early testing of our method, we discovered that the *Recercar Undecimo* by an unidentified composer in a 1593 miscellany,¹⁵ is in fact a previously unrecognized instrumental arrangement of a motet, ‘In die tribula-

tionis,’ by ‘Damianus’, probably Damien Havericq (active 1538-56), published half a century earlier in 1549.¹⁶

At first we extracted ngrams from the page-encodings, i.e. fixed-length substrings of length k extracted sequentially starting at each character in the string in turn. These were built into a trie (suffix-tree) structure for efficient searching. We then counted the number of ngrams in common between the query and each page of the collection in turn. Although this worked well enough for task (a), we encountered difficulties with tasks (b) and (c) for two reasons: firstly, this naïve ranking did not take account of the fact that longer pages are more likely to contain ngrams which appear in the query by chance, and secondly, we were ignoring the order of locations of the ngrams, which should be the same in query and target documents, for obvious musical reasons.

The first difficulty was overcome by using Jaccard distance¹⁷ rather than a raw count of coincident ngrams; all results reported here use this measure as a basis for search-result ranking. The second problem can be tackled by including ngram-location in the index and sorting the array of results. However, the process of ensuring an ordered match from the ngram set adds undesirable computational complexity.

Turning to a method that has found wide acceptance in recent bioinformatics, we used minimal absent words (MAWs)¹⁸ instead of ngrams. We have found this to be highly successful, both in terms of the reduction of the amount of data that has to be searched and because of the fact that MAWs retain the order and structure of the original document, avoiding the necessity for the secondary expensive sorting routine.

5. EXPERIMENTS

For the purposes of the comparison between retrieval using ngrams and MAWs, we ran experiments based on the three user tasks outlined above using a version of the software implemented in Javascript on a MacBook Pro (2.5 GHz Intel Core i7 with 8GB RAM), running OS X 10.13.3. The software was run in a standard web browser (Safari) via localhost. While we would not consider this to be a sensible setup for production work, it had the advantage of not requiring network access with consequent latency issues.¹⁹

For each task we ran the searches using indexes of different word-lengths (3-10 characters) and the two word-types (*ngrams* and *MAWs*). In addition (as explained below) we used an index of MAWs of mixed length (4-8 characters).

For ngrams we did not include the result-sorting routine. We expect that sorted ngram results will give the overall best retrieval performance, but this will come at a significant cost in terms of speed, not evaluated here. In

14 Premier livre des cha[n]so[n]s a quatre parties (Antwerp: Susato, 1543, f. xi

15 *Fantasia recercari et contrapunti a tre voci* (Venice: Gardane, 1593)

16 *Libro secondo de li motetti a tre voce da diversi* (Venice: Scotto, 1549), item XVIII

17. https://en.wikipedia.org/wiki/Jaccard_index

18. <http://www.lix.polytechnique.fr/SeminaireDoctorants/AliceHelieuMotsAbsents.pdf>

19. The code and encoded data are accessible at: <http://doc.gold.ac.uk/~mas01tc/ISMIR2018/>

fact, we believe that these will find their best use on reduced result lists after the initial indexed search.

Before each experiment, the appropriate full index needs to be loaded into a trie (suffix-tree) structure. This process can take up to a minute or so for the larger indexes which also use a lot of memory. Index loading is not considered as part of our experiments, since the indexes would need to be retained as a persistent service (probably distributed between machines) in a production system.

Each task has its associated query-list derived from the predetermined ground truth (see below). These contain different numbers of queries (48, 107 and 334 for the *dupl*, *relv* and *same* tasks, respectively). Each query, consisting of a set of index words, was run by searching in turn for each word on the complete index, counting the words in common between query and target pages, with results sorted by Jaccard distance. Where the number of common words was less than 6 the search was regarded as unsuccessful and no results were returned.²⁰ For certain word lengths, no MAWs were generated for some pages (see Discussion, below); for these cases, too, no results were returned.

6. EVALUATION

We had previously gathered ground truth using a web-interface allowing a user to annotate documents in ranked results as (a) a duplicate image of the same page (*dupl*); (b) a page containing substantially the same music (*same*); or (c) related or relevant music, such as that belonging to a different voice-part or section of a work (*relv*).

In the three graphs that follow we present the average rank at which known matches from the ground truth lists for a given word length were retrieved in the three experiments. Since we were mainly interested in high-ranking matches, we gave all items falling beneath the rank of 20 a uniform rank of 25.

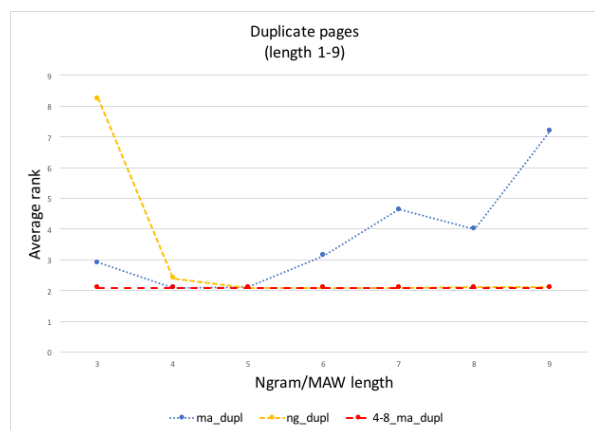


Figure 3. Average ranks for matches of ‘duplicate’ pages.

In our experiments with our test dataset, retrieval performance for the *dupl* task was found to be similar for

ngrams and MAWs of length 5 characters. We do not expect, however, that this will remain true for all other collections, and it is not the case for the other tasks. For this reason, we also performed all the tasks with a mixed-length index of MAWs (4-8 characters) which gave results almost identical to ngrams in the *dupl* task, consistently high in the ranked results in the case of the *same* task, and the overall best for the *relv* task.

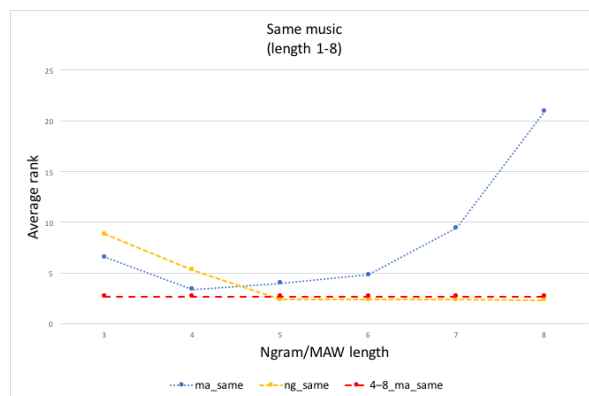


Figure 4. Average ranks for matches of ‘same music’ pages

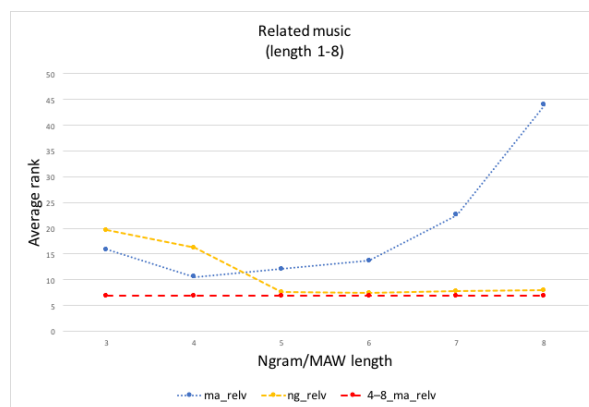


Figure 5. Average ranks for matches of ‘related music’

The experiments are named using ‘ng’ and ‘ma’ to indicate the use of ngrams or MAWs. The dashed lines on the graphs represent the average rank for the searches using mixed-length MAWs (4-8 chars); these are not quite as good as the best results for ngrams, but very close, and the speed is much faster.

7. DISCUSSION

The usefulness of MAWs is highly data-dependent. Over a length-range of 3 to 10 characters, the number of MAWs generated for each page, while lower than the number of ngrams of those lengths, falls off in a way that means that there is simply not enough data for consistent recognition beyond a certain length.

20. This arbitrary number was arrived at in early testing as lower numbers gave essentially useless results.

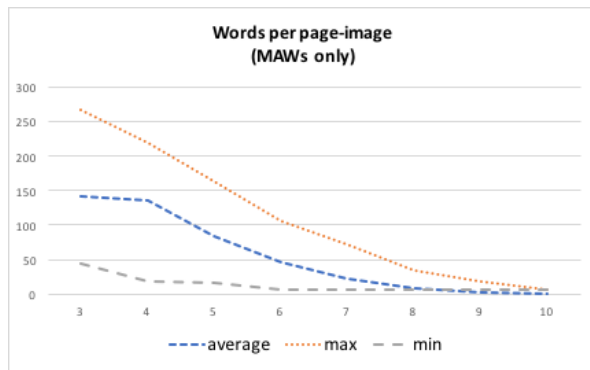


Figure 6. Number of minimal absent words per page-image (average, maximum and minimum)

However, a database of MAWs with mixed lengths (4-8) always produces enough data for matching and performs almost as well as the best ngram length, but is much faster in operation.

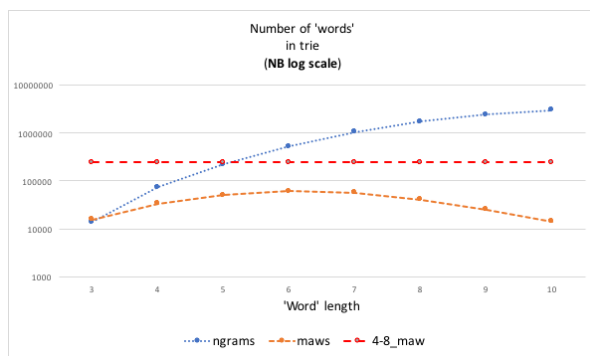


Figure 7. Number of words in the trie structure for the entire collection. NB Log scale!

Finally, we show the average search time per word in the collection for each word-length:

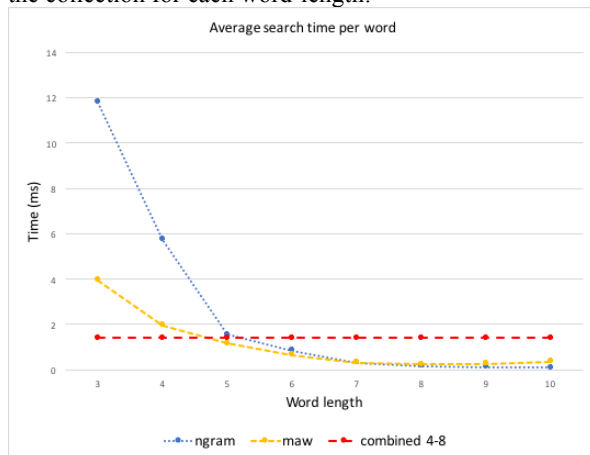


Figure 8. Average search time (ms) per word for each word length

Bearing in mind that we need to search for each word in the query page in turn, it can be seen that the lower numbers per page of MAWs compared to ngrams, and the consequently smaller index size brings a significant speed

advantage. This is particularly important in a search tool which is to be operated by human researchers, who increasingly expect retrieval response comparable to that encountered in everyday web searching.

A possibly interesting finding, whose significance needs further investigation, is that there is a fairly consistent range of ngram and MAW lengths (viz. roughly between 4 and 8 characters) that produces useful results - this may relate to the nature of the musical data, i.e. to the 'language' or style of the music, but this needs to be tested formally with a range of different repertoires.

8. FURTHER WORK

In future work, we intend to compare the efficacy and performance of MAWs with standard algorithms such as BLAST.

Since our use of ngrams in this research was to provide a benchmark for the efficacy of MAWs, limited attempts have been made to optimise them for retrieval speed. We are confident that with the data that we now have on the most effective ngram lengths, effort can be put into algorithmic efficiency for a comparison of the two technologies based on their real-world speed.

In many retrieval tasks, it is sufficient simply to return a ranked list of the k best matches for a query, but in the tasks we investigate here, there is an approximately binary relevance judgement to be made. The number of relevant documents can vary from 0 to over 100, so finding an appropriate thresholding value is important. Statistical approaches to thresholding have proved useful in the high-dimensional spaces associated with audio searching [18], and this is a *sine qua non* in text retrieval.

We intend to increase the size of our test collection to investigate how well it scales. In order to achieve this, we hope to establish a consortium of international music libraries to contribute images and metadata, with the ultimate goal of providing a comprehensive search tool for musicologists. This requires further work on system architecture and management of distributed data and processing.

In principle, there is no reason why similar techniques could not be used on other monophonic repertoires, and we hope to widen the scope of our work through our continuing association with projects such as SIMSSA21 and TROMPA.²²

MAWs present a valuable new method for music research which is scalable to collections a good deal bigger than our test set of 32k pages. The technique is generally applicable to any repertoire which is reducible to monophonic parts or streams, allowing fast approximate retrieval of large queries over web-scale collections of noisy data.

21. *Single Interface for Music Score Searching and Analysis* (project funded by Social Sciences and Humanities Research Council, Canada)

22. *Towards Richer Online Music Public-domain Archives* (Horizon 2020 project funded by the EU, 2018-21)

9. ACKNOWLEDGMENTS

We gratefully acknowledge the help and advice of Prof. Maxime Crochemore and Dr Jeremy Pickens in the writing of this paper. Our thanks are due to Solon Pissis for help with adapting his MAW-extraction code for our musical purposes. The work was partially funded by the UK AHRC project, Transforming Musicology, AH/L006820/1.

10. REFERENCES

- [1] M. Schedl, E. Gómez and J. Urbano: “Music Information Retrieval: Recent Developments and Applications,” *Foundations and Trends in Information Retrieval*, Vol. 8, No. 2-3 127–261, 2014.
- [2] D. Bountouridis: “Music Information Retrieval Using Biologically-Inspired Techniques,” PhD dissertation, Utrecht University, 2018.
- [3] W. R. Pearson and D. J. Lipman: “Improved tools for biological sequence comparison,” *Proc Natl Acad Sci USA*. 85(8), 2444-8, April 1988.
- [4] R. B. Dannenberg and N. Hu: “Pattern Discovery Techniques for Music Audio”, *Proceedings of the International Symposium on Music Information Retrieval*, 2002.
- [5] J. Kilian and H. Hoos: “MusicBLAST — Gapped Sequence Alignment for MIR”, *Proceedings of the International Symposium on Music Information Retrieval*, 2004.
- [6] B. Martin, D.G. Brown, P. Hanna and P. Ferraro: “BLAST for Audio Sequences Alignment: A Fast Scalable Cover Identification Tool,” *Proceedings of the International Symposium on Music Information Retrieval*, 529-534, 2012.
- [7] D. Bountouridis, D.G Brown, F. Wiering and R.C. Veltkamp: “Melodic similarity and applications using biologically-inspired techniques”, *Applied Sciences*, Special Issue on Sound and Music Computing, 7. 12, 2017.
- [8] K. Katoh, K. Misawa, K. Kuma, and T. Miyataa: “MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform,” *Nucleic Acids Res*. 15; 30 (14), 3059-66, July 2002.
- [9] M. Beal, F. Mignosi, A. Restivo and M. Sciortino: “Forbidden words in symbolic dynamics,” *Advances in Applied Mathematics*, Vol 25(2), 163–193, 2000.
- [10] M. Crochemore, G. Fici, R. Mercas and S. Pissis: “Linear-Time Sequence Comparison Using Minimal Absent Words & Applications,” *Proceedings of the Latin American Theoretical Informatics Symposium (LATIN)*, 334-346, 2016.
- [11] S.P. Garcia and A.J. Pinho: “Minimal absent words in four human genome assemblies,” *PLOS ONE*, Vol. 6 (12), 2011.
- [12] S. Chairungsee and M. Crochemore: “Using minimal absent words to build phylogeny,” *Theoretical Computer Science*, Vol. 450, 109–116, 2012.
- [13] C. Barton, A. Heliou, L. Mouchard and S.P. Pissis: “Linear-time computation of minimal absent words using suffix array,” *BMC Bioinformatics* Vol. 15 (1), 2014.
- [14] W.K. Sung, *Algorithms in Bioinformatics: A Practical Introduction*, CRC Press, London, UK, 2009.
- [15] L. Pugin and T. Crawford, “Evaluating OMR on the Early Music Online Collection,” *Proceedings of the 14th International Society for Music Information Retrieval Conference*, pp. 439–44, 2013.
- [16] J. Calvo-Zaragoza, J.J. Valero-Mas and A. Pertusa: “End-to-end Optical Music Recognition Using Neural Networks,” *Proceedings of the International Symposium on Music Information Retrieval*, pp. 472–477, 2017.
- [17] J. Diet and M. Gerritsen: “Encoding, Searching, and Displaying Music Incipits in the RISM-OPAC,” *Music Encoding Conference 2013*, Mainz, Germany (unpublished).
- [18] M. Casey, M. Slaney and C. Rhodes: “Analysis of minimum distances in high-dimensional musical spaces,” *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 16 (5), 1015-1028, 2008.

OPTICAL MUSIC RECOGNITION IN MENSURAL NOTATION WITH REGION-BASED CONVOLUTIONAL NEURAL NETWORKS

Alexander Pacha

Institute of Visual Computing and Human-Centered Technology, TU Wien, Austria
alexander.pacha@tuwien.ac.at

Jorge Calvo-Zaragoza

PRHLT Research Center
Universitat Politècnica de València, Spain
jcalvo@upv.es

ABSTRACT

In this work, we present an approach for the task of optical music recognition (OMR) using deep neural networks. Our intention is to simultaneously detect and categorize musical symbols in handwritten scores, written in mensural notation. We propose the use of region-based convolutional neural networks, which are trained in an end-to-end fashion for that purpose. Additionally, we make use of a convolutional neural network that predicts the relative position of a detected symbol within the staff, so that we cover the entire image-processing part of the OMR pipeline. This strategy is evaluated over a set of 60 ancient scores in mensural notation, with more than 15000 annotated symbols belonging to 32 different classes. The results reflect the feasibility and capability of this approach, with a weighted mean average precision of around 76% for symbol detection, and over 98% accuracy for predicting the position.

1. INTRODUCTION

The preservation of the musical heritage over the centuries makes it possible to study a certain artistic or cultural paradigm. Most of this heritage exists in written form and is stored in cathedrals or music libraries [10]. In addition to the possible issues related to the ownership of the sources, this storage protects the physical preservation of the sources over time, but also limits their accessibility. That is why efforts are being made to improve this situation through initiatives to digitize musical archives [17,21]. These digital copies can easily be distributed and studied without compromising their integrity.

Nevertheless, this digitalization, which indeed represents a progress with respect to the aforementioned situation, is not enough to exploit the actual potential of this heritage. To make the most out of it, the musical content itself must be transcribed into a structured format that can be processed by a computer [6]. In addition to indexing

the content and thereby enabling tasks such as content-based search, this could also facilitate large-scale data-driven musicological analysis in general [39].

Given that the transcription of sources is extremely time-consuming, it is desirable to resort to automatic systems. Optical music recognition (OMR) is a field of research that investigates how to build systems that decode music notation from images. Regardless of the approach used to achieve such objective, OMR systems vary significantly due to the differences amongst musical notations, document layouts, or printing mechanisms.

The work presented here deals with manuscripts written in mensural notation, specifically with sources from the 17th century, attributed to the Pan-Hispanic framework. Although this type of mensural notation is generally considered as an extension of the European mensural notation, the Pan-Hispanic situation of that time underwent a particular development that fostered the massive use of handwritten copies. Due to this circumstance, the need for developing successful OMR systems for handwritten notation becomes evident.



Figure 1. A sample page of ancient music, written in mensural notation.

We address the optical music recognition of scores written in mensural notation (see Figure 1) as an object detection and classification task. In this notation, the symbols are atomic units,¹ which can be detected and categorized independently. Although there are polyphonic composi-

¹ Except for beamed notes, in which the beam can be considered an atomic symbol itself.



tions from that era, each voice was placed on its own page, so we can consider the notation as monophonic on the graphical level. Assuming the aforementioned simplifications allows us to formulate OMR as an object detection task in music score images, followed by a classification stage that determines the vertical position of each detected object within a staff. If the clef and other alterations are known, the vertical position of a note encodes its pitch.

We propose using region-based convolutional neural networks, which represent the state of the art in computer vision for object detection, and demonstrate their capabilities of detecting and categorizing the musical symbols that appear in the image of a music score with a high precision. We believe that this work provides a solid foundation for the automatic encoding of scores into a machine-readable music format like Music Encoding Initiative (MEI) [38] or MusicXML [15]. At present, there are thousands of manuscripts of this type that remain to be digitized and transcribed. Although each manuscript may have its own particularities (such as the handwriting style or the layout organization), the approach developed in this work presents a common and extensible formulation to all of them.

2. RELATED WORK

Most of the proposed solutions to OMR have focused on a multi-stage approach [34]. This traditional workflow involves steps that have been addressed isolatedly, such as image binarization [4,47], staff and text segmentation [44], staff-line detection and removal [5, 11, 46], and symbol classification [3, 30, 33]. In other works, a full pipeline is proposed for a particular type of music score [31, 32, 43].

Recent works have shown that the image-processing pipeline can largely be replaced with machine-learning approaches, making use of deep learning techniques such as convolutional neural networks (CNNs) [1, 16, 29, 45]. CNNs denote a breakthrough in machine learning, especially when dealing with images. They have been applied with great success to many computer vision tasks, often reaching or even surpassing human performance [18, 22]. These neural networks are composed of a series of filters that operate locally (i.e. convolutions, pooling) and compute various representations of the input image. These filters form a hierarchy of layers, each of which represents a different level of abstraction [20]. The key is that these filters are not fixed but learnt from the raw data through a gradient descent optimization process [23], meaning that the network can learn to extract data-specific, high-level features.

Here, we formulate OMR for mensural notation as an object detection task in music score images. Object detection in images is one of the fundamental problems in computer vision, for which deep learning can provide excellent solutions. Traditionally, the task has been addressed by means of heuristic strategies based on the extraction of low-level, general-purpose features such as SIFT [28] or HOG [7]. Szegedy and colleagues [8, 42] redefined the use of CNNs for object detection for the first time. Instead

of classifying the image, the neural network predicted the bounding box of the object within the image. Around the same time, the ground-breaking work of Girshick et al. [14] definitely changed the traditional paradigm. In their work, a CNN was in charge of predicting whether each object of the vocabulary appeared in selected bottom-up regions of the image. This scheme has been referred to as region-based convolutional neural network (R-CNN). Afterwards, several extensions and variations have been proposed with the aim of improving both the quality of the detection and the efficiency of the process. Well-known examples include Fast R-CNN [13], Faster R-CNN [37], R-FCN [24], SSD [27] or YOLO [35, 36].

In this work, we use these region-based convolutional neural networks for OMR, which are trained for the direct detection and categorization of music symbols in a given music document. Thereby allowing for an elegant formulation of the task, since the training process only needs score images along with their corresponding set of symbols and the regions (bounding boxes) in which they appear.

3. AN OMR-PIPELINE FOR MENSURAL SCORES

Music scores written in mensural notation share many properties with scores written in modern notation: the sequence of tones and pauses is captured as notes and rests within a reference frame of five parallel lines, temporally ordered along the x-axis with the y-axis representing the pitch of notes. But unlike modern notation, mensural scores are notated monophonically with a smaller vocabulary of only around 30 different glyphs, reducing the overall complexity significantly and thus allowing for a simplified pipeline that consists of only three stages. A representative subset of the symbols that appear in the considered notation is depicted in Table 1.





















Group	Symbol			
Note	Semibrevis	Minima	Col. Minima	Semiminima
				
Rest	Longa	Brevis	Semibrevis	Semiminima
				
Clef	C Clef	G Clef	F Clef (I)	F Clef (II)
				
Time	Major	Minor	Common	Cut
				
Others	Flat	Sharp	Dot	Custos
				

Table 1. Subset of classes from mensural notation. The symbols are depicted without considering their pitch or vertical position on the staff.

3.1 Music Object Detection

The first stage takes as input an entire high-quality image that contains music symbols. The entire image is fed into

a deep convolutional neural network for object detection and yields the bounding boxes of all detected objects along with their most likely class (e.g., *g-clef*, *minima*, *flat*).

3.2 Position classification

After detecting the symbols and classifying them, the second stage performs position classification of each detected object to obtain the relative position with respect to the reference frame (staff) which is required to recover a notes pitch. For this process, we extract a local patch from the full image with the object of interest in the center and feed the image into another CNN, which outputs the vertical position, encoded as shown in Figure 2.

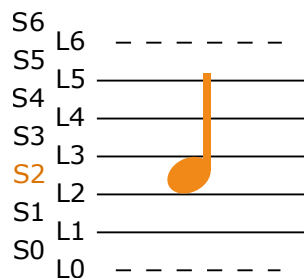


Figure 2. Encoding of the vertical staff line position into discrete categories. The five continuous lines in the middle form the regular staff and the dashed lines represent ledger lines, that are inserted locally as needed. A note between the second and third line from the bottom would be classified as S2 (orange).

3.3 Semantics Reconstruction and Encoding

Given the detected objects and their relative position to the staff line, the final step is to reconstruct the musical semantics and encode the output into the desired format (e.g., into modern notation [48]). This step has to translate the detected objects into an ordered sequence for further processing. Depending on the application and desired output, semantic rules need to be taken care of, such as grouping beams with their associated notes to infer the right duration or altering the pitch of notes when accidentals are encountered.

4. EXPERIMENTS

To evaluate the proposed approach, we conducted experiments² for the first two steps of the pipeline. While a full system would also require the third step, we refrain from implementing it, to not restrict this approach to a particular applications. It is also noteworthy, that translating mensural notation into modern notation can be seen as its own field of research that requires a deep understanding of

both notational languages, which exceeds the scope of this work.

4.1 Dataset

Our corpus consists of 60 fully-annotated pages in mensural notation from the 16th-18th century. The manuscript represents sacred music, composed for vocal interpretation.³ The compositions were written in music books by copyists of that time. To ensure the integrity of the physical sources, the images were taken with a camera instead of scanning the books in a flatbed scanner, leading to sub-optimal conditions in some cases. An overview of the considered corpus is given in Table 2.

Pages	60
Total number of symbols	15258
Different classes	32
Different positions within a staff	14
Average size of a symbol ($w \times h$)	44×84 pixels
Number of symbols per image	42–447 (\varnothing 250)
Image resolution ($w \times h$)	$\sim 3000 \times 2000$ pixels
Dots per inch (DPI)	300

Table 2. Statistics of the considered corpus.

The ground-truth data is collected using a framework, in which an electronic pen is used to trace the music symbols, similar to that of [2]. The bounding boxes of the symbols are then obtained by computing the rectangular extent of the users' strokes.

4.2 Setup

Our experiments are based on previous research by [29], where a sliding-window-approach is used to detect handwritten music symbols in sub-regions of a music score. In contrast to their work, we are able to detect hundreds of tiny objects in the full page within a single pass. To train a network in a reasonable amount of time within the constraints of modern hardware, it is currently necessary to shrink the input image to be no longer than 1000px on the longest edge, which corresponds to a downscaling operation by a factor of three on our dataset.

For detecting music objects, the Faster R-CNN approach [37] with the Inception-ResNet-v2 [41] feature extractor has been shown to yield very good results for detecting handwritten symbols [29]. It works by having a region-proposal stage for generating suggestions, where an

² Source code is available at <https://github.com/apacha/Mensural-Detector>

³ The dataset is subject to ongoing musicological research and can not be made public at this point in time, so it is only available upon request.

object might be, followed by a classification stage, which confirms or discards these proposals. Both stages are implemented as CNNs and trained jointly on the provided dataset. The first stage scans the image linearly along a regular grid with user-defined box proposals in each cell of that grid.

To be able to generate meaningful proposals, the shape of these boxes has to be similar to the actual shape of the objects that should be found. Since the image contains a large number of very tiny objects (sometimes only a few pixels), a very fine grid is required. After a statistical analysis of the objects appearing in the given dataset, including dimension clustering [35], several experiments were conducted to study the effects of size, scale, and aspect ratios of the above-mentioned boxes, concluding that sensibly chosen priors for these boxes work similarly good as the boxes obtained from the statistical analysis. For the down-scaled image, boxes of 16x16 pixels, iterating with a stride of 8 pixels and using the scales 0.25, 0.5, 1.0, and 2.0, with aspect ratios of 0.5, 1.0, and 2.0 represent a meaningful default configuration. Accounting for the high density of objects, the maximum number of box proposals is set to 1200 with a maximum of 600 final detections per image.

For the second step of our proposed pipeline, another CNN is trained to infer the relative position of an object to its staff line upon which it is notated (see Figure 2). Different off-the-shelf network architectures are evaluated (VGG [40], ResNet [19], Inception-ResNet-v2 [41]) with the more complex models slightly outperforming the simpler ones. Using pre-trained weights instead of random initialization accelerates the training, improves the overall result, and is therefore used throughout all experiments. The input to the classification network is a 224×448 pixels patch of the original image with the target object in the center (see Figure 3). The exact dimensions of the patch are not important, as long as the image contains enough vertical and horizontal context to classify even symbols notated above or below the staff. When objects appear too close to the border, the image is padded with the reflection along the extended edge to simulate the continuation of the page as shown in Figures 3(d) and 3(e).

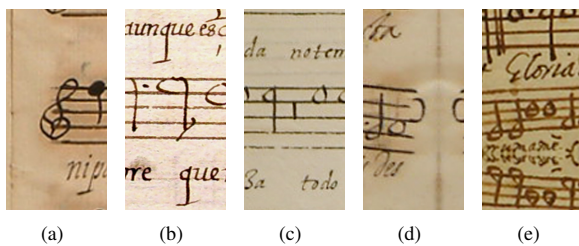


Figure 3. Sample inputs for the position classification network depicting a *g-clef* (a), *semiminima* (b), *brevis rest* (c), *custos* (d) and *semibrevis* (e), with vertical (d) and horizontal (e) reflections of the image to enforce the target object to be in the center, while preserving meaningful context.

It is important to notice that the vertical position defines the semantical meaning only for some symbols (e.g.,

the pitch of a *note* or the upcoming pitch with a *custos*). Classes for which the position is either undefined or not of importance include *barlines*, *fermatas*, different *time-signatures*, *beams* and in particular for mensural notation: the *augmentation dot*. Symbols from these classes can be excluded from the second step.

4.3 Evaluation metrics

Concerning the music object detection stage, the model provides a set of bounding box proposals, as well as the recognized class of the objects therein. The model also yields a *score* of its confidence for each proposal. A bounding box proposal B_p is considered positive if it overlaps with the ground-truth bounding box B_g exceeding 60%, according to the Intersection over Union (IoU) criterion: ⁴

$$\frac{\text{area}(B_p \cap B_g)}{\text{area}(B_p \cup B_g)}$$

If the recognized class matches the actual category of the object, it is considered a true positive, being otherwise a false positive. Additional detections of the same object are computed as false positives as well. Those objects for which the model makes no proposal are considered false negatives. Given that the prediction is associated with a score, different values of *precision* and *recall* can be obtained for each possible threshold. To obtain a single metric, Average Precision (AP) can be computed, which is defined as the area under this precision-recall curve. An AP value can be computed independently for each class, and then we provide the mean AP (mAP) as the mean across all classes. Since our problem is highly unbalanced with respect to the number of objects of each class, we also compute the weighted mAP (w-mAP), in which the mean value is weighted according to the frequency of each class. For the second part of the pipeline (position classification), we evaluate the performance with the accuracy rate (ratio of correctly classified samples).

5. RESULTS

Both experiments yielded very promising results while leaving some room for improvement. The detection of objects in the full image (see Figure 4) was evaluated by training on 48 randomly selected images and testing on the remaining 12 images with a 5-fold cross-validation. This task can be performed very well and yielded 66% mAP and 76% w-mAP. When considering practical applications, the weighted mean average precision indicates the effort needed to correct the detection results, because it reflects the fact that symbols from classes that appear frequently are generally detected better than rare symbols.

When reviewing the error cases, a few things can be observed: Very tiny objects such as the *dot*, *semibrevis rest* and *minima rest* pose a significant challenge to the network, due to their small size and extremely similar appearance (see Figure 5). This problem might be mitigated,

⁴ as defined for the PASCAL VOC challenge [9]



Figure 4. Detected objects in the full image with the detected class being encoded as the color of the box. This example achieves a mAP of approximately 68% and a w-mAP of 85%.

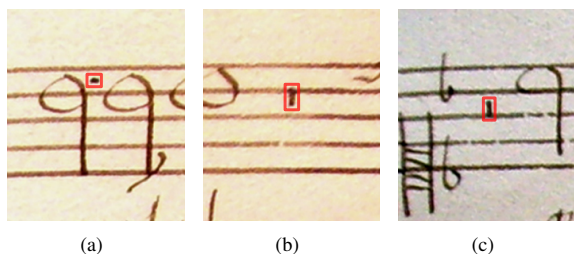


Figure 5. The smallest objects from the dataset that are hard to detect and often confused (from left to right): *dot*, *semibrevis rest*, and *minima rest*.

by allowing the network to access the full resolution image, which potentially has more discriminative information than the downsized image. Unsurprisingly, classes that are underrepresented such as *dots*, *barlines*, or all types of *rests* are also frequently missed or incorrectly classified, leading to average precision rates of only 10–40% for these classes.

Another interesting observation can be made, that in many cases, objects were detected but the IoU with the underlying ground-truth was too low for considering them a true positive detection (see Figure 6 with a red box being very close to a white box).

For the second experiment, a total of 13246 sym-

bols were split randomly into a training (80%), validation (10%) and test set (10%). The pre-trained Inception-ResNet-v2 model is then fine-tuned on this dataset and achieves over 98% accuracy on the test set of 1318 samples. Analyzing the few remaining errors reveals that the model makes virtually no errors and that the misclassified samples are mostly human annotation errors or data inconsistencies.

For inference, both networks can be connected in series. Running both detection and classification takes about 30 seconds per image when running on a GPU (GeForce 1080 Ti) and 210 seconds on a CPU.

6. CONCLUSION

In this work, we have shown that the optical music recognition of handwritten music scores in mensural notation, can be performed accurately and extendible by formulating it as an object detection problem, followed by a classification stage to recover the position of the notes within the staff. By using a machine learning approach with region-based convolutional neural networks, this problem can be solved by simply providing annotated data and training a suitable model on that dataset. However, we are aware that our proposal still has room for improvement. In future work we would like to:

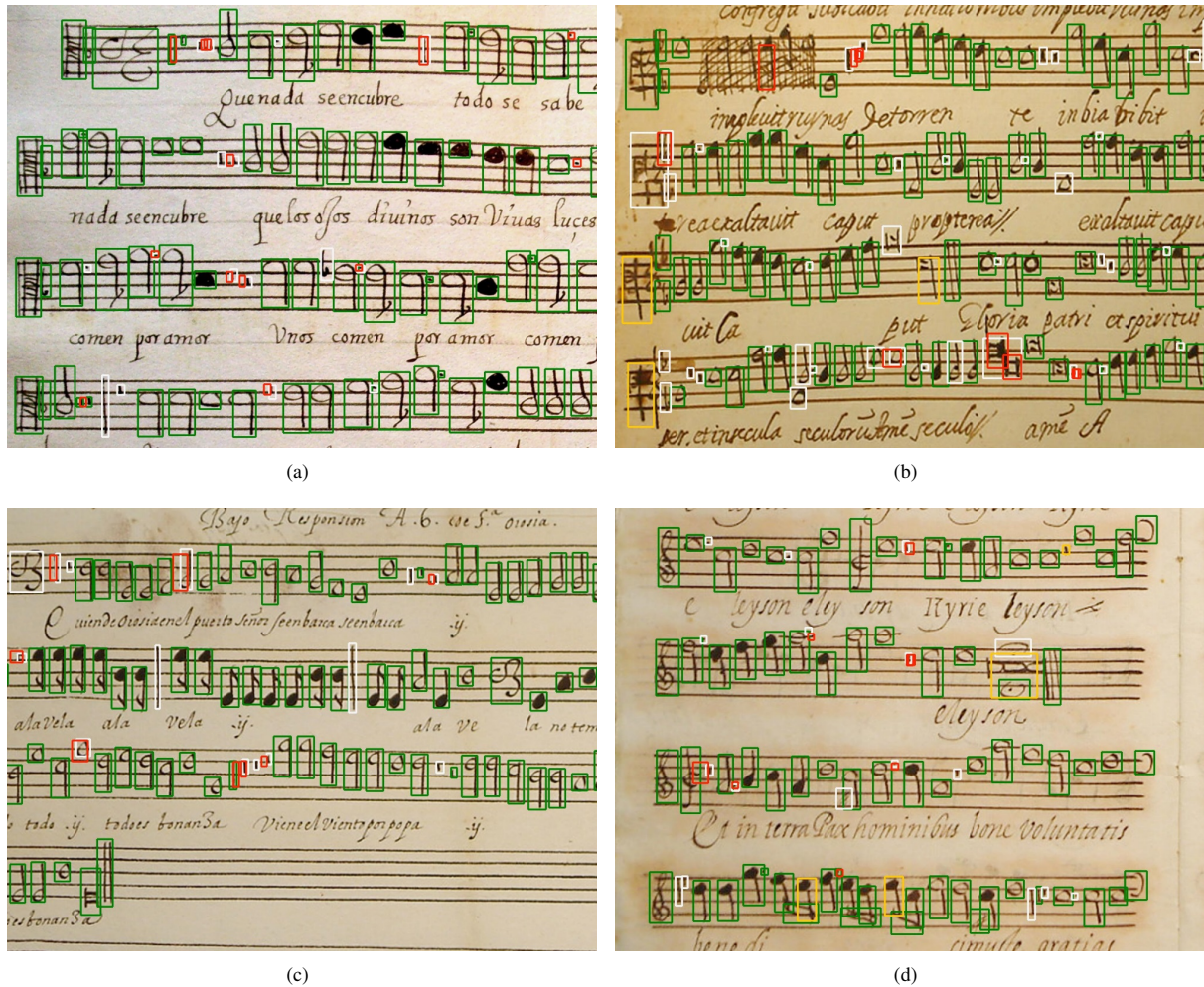


Figure 6. Visualization of the performance of the object detection stage with selected patches of the music documents: green boxes indicate true positive detections; white boxes are false negatives, that the network missed during detection; red boxes are false positive detections, where the model reported an object, although there is no ground-truth; yellow boxes are also false positives, where the bounding-box is valid, but the assigned class was incorrect.

- evaluate the use of different network architectures, such as feature pyramid networks [25,26], that might improve the detection of small objects, which we have identified as the biggest source of error at the moment. These networks allow the use of high-resolution images directly, without the inherent information loss, that is caused by the downscaling operation.
- merge the staff position classification with the object detection network, by adding another output to the neural network, so the model simultaneously predicts the staff position, the bounding box and the class label.
- apply and evaluate the same techniques for other notations, including modern notation
- study models or strategies that reduce (or remove) the need for specific ground-truth data of each type of manuscript. For example, unsupervised training

schemes such as the one proposed in [12], which allows the network to adapt to a new domain by simply providing new, unannotated images.

We believe that this research avenue represents a ground-breaking work in the field of OMR, as the presented approach would potentially deal with any type of music scores by just providing undemanding ground-truth data to train the neural models.

7. ACKNOWLEDGEMENT

Jorge Calvo-Zaragoza thanks the support from the European Union's H2020 grant READ (Ref. 674943), the Spanish Ministerio de Economía, Industria y Competitividad through Juan de la Cierva - Formación grant (Ref. FJCI-2016-27873), and the Social Sciences and Humanities Research Council of Canada.

8. REFERENCES

- [1] J. Calvo-Zaragoza and D. Rizo. End-to-End Neural Optical Music Recognition of Monophonic Scores. *Applied Sciences*, 8(4):606–629, 2018.
- [2] J. Calvo-Zaragoza, D. Rizo, and J. M. Iñesta. Two (note) heads are better than one: pen-based multimodal interaction with music scores. In *17th International Society for Music Information Retrieval Conference*, pages 509–514, 2016.
- [3] J. Calvo-Zaragoza, A. J. G. Sánchez, and A. Pertusa. Recognition of Handwritten Music Symbols with Convolutional Neural Codes. In *14th IAPR International Conference on Document Analysis and Recognition*, pages 691–696, 2017.
- [4] J. Calvo-Zaragoza, G. Vigliensoni, and I. Fujinaga. Pixel-wise binarization of musical documents with convolutional neural networks. In *15th IAPR International Conference on Machine Vision Applications*, pages 362–365, 2017.
- [5] J. S. Cardoso, A. Capela, A. Rebelo, C. Guedes, and J. P. da Costa. Staff detection with stable paths. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1134–1139, 2009.
- [6] G. S. Choudhury, M. Droetboom, T. DiLauro, I. Fujinaga, and B. Harrington. Optical music recognition system within a large-scale digitization project. In *1st International Symposium on Music Information Retrieval*, pages 1–6, 2000.
- [7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.
- [8] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2154, 2014.
- [9] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.
- [10] I. Fujinaga, A. Hankinson, and J. E. Cumming. Introduction to SIMSSA (Single Interface for Music Score Searching and Analysis). In *1st International Workshop on Digital Libraries for Musicology*, pages 1–3, 2014.
- [11] A.-J. Gallego and J. Calvo-Zaragoza. Staff-line removal with selectional auto-encoders. *Expert Systems with Applications*, 89:138–148, 2017.
- [12] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.
- [13] R. Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.
- [14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.
- [15] M. Good and G. Actor. Using MusicXML for file interchange. In *Third International Conference on WEB Delivering of Music*, page 153, 2003.
- [16] J. Hajič Jr. and P. Pecina. Detecting Noteheads in Handwritten Scores with ConvNets and Bounding Box Regression. *Computing Research Repository*, abs/1708.01806, 2017.
- [17] A. Hankinson, J. A. Burgoyne, G. Vigliensoni, A. Porter, J. Thompson, W. Liu, R. Chiu, and I. Fujinaga. Digital Document Image Retrieval Using Optical Music Recognition. In *Proceedings of the 13th International Society for Music Information Retrieval Conference*, pages 577–582, 2012.
- [18] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.
- [19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [20] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012.
- [21] A. Laplante and I. Fujinaga. Digitizing musical scores: Challenges and opportunities for libraries. In *3rd International workshop on Digital Libraries for Musicology*, pages 45–48. ACM, 2016.
- [22] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [24] Y. Li, K. He, J. Sun, et al. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, pages 379–387, 2016.
- [25] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature Pyramid Networks for Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

- [26] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection. *Computing Research Repository*, abs/1708.02002, 2017.
- [27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37, 2016.
- [28] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [29] A. Pacha, K.-Y. Choi, B. Coüasnon, Y. Ricquebourg, R. Zanibbi, and H. Eidenberger. Handwritten music object detection: Open issues and baseline results. In *13th IAPR Workshop on Document Analysis Systems*, pages 163–168, 2018.
- [30] A. Pacha and H. Eidenberger. Towards a Universal Music Symbol Classifier. In *12th IAPR International Workshop on Graphics Recognition*, pages 35–36, 2017.
- [31] L. Pugin. Optical music recognition of early typographic prints using hidden markov models. In *7th International Conference on Music Information Retrieval*, pages 53–56, 2006.
- [32] C. Ramirez and J. Ohya. Automatic recognition of square notation symbols in western plainchant manuscripts. *Journal of New Music Research*, 43(4):390–399, 2014.
- [33] A. Rebelo, A. Capela, and J. S. Cardoso. Optical recognition of music symbols. *International Journal on Document Analysis and Recognition*, 13(1):19–31, 2010.
- [34] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. Marcal, C. Guedes, and J. S. Cardoso. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190, 2012.
- [35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.
- [36] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6517–6525, 2017.
- [37] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [38] P. Roland. The music encoding initiative (MEI). In *Proceedings of the First International Conference on Musical Applications Using XML*, pages 55–59, 2002.
- [39] X. Serra. The computational study of a musical culture through its digital traces. *Acta Musicologica*. 2017; 89 (1): 24-44., 2017.
- [40] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *Computing Research Repository*, abs/1409.1556, 2014.
- [41] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *31st AAAI Conference on Artificial Intelligence*, pages 4278–4284, 2017.
- [42] C. Szegedy, A. Toshev, and D. Erhan. Deep Neural Networks for Object Detection. In *Advances in Neural Information Processing Systems 26*, pages 2553–2561, 2013.
- [43] L. J. Tardón, S. Sammartino, I. Barbancho, V. Gómez, and A. Oliver. Optical Music Recognition for Scores Written in White Mensural Notation. *EURASIP Journal on Image and Video Processing*, 2009(1):1–23, 2009.
- [44] R. Timofte and L. Van Gool. Automatic stave discovery for musical facsimiles. In *Asian Conference on Computer Vision*, pages 510–523, 2012.
- [45] E. van der Wel and K. Ullrich. Optical music recognition with convolutional sequence-to-sequence models. In *18th International Society for Music Information Retrieval Conference*, pages 731–737, 2017.
- [46] M. Visaniy, V. C. Kieu, A. Fornés, and N. Journet. The ICDAR 2013 music scores competition: Staff removal. In *International Conference on Document Analysis and Recognition*, pages 1407–1411, 2013.
- [47] Q. N. Vo, S. H. Kim, H. J. Yang, and G. Lee. An MRF model for binarization of music scores with complex background. *Pattern Recognition Letters*, 69:88–95, 2016.
- [48] Yu-Hui Huang, Xuanli Chen, Serafina Beck, David Burn, and Luc J. Van Gool. Automatic Handwritten Mensural Notation Interpreter: From Manuscript to MIDI Performance. In *16th International Society for Music Information Retrieval Conference*, pages 79–85, 2015.

CAMERA-PRIMUS: NEURAL END-TO-END OPTICAL MUSIC RECOGNITION ON REALISTIC MONOPHONIC SCORES

Jorge Calvo-Zaragoza

PRHLT Research Center

Universitat Politècnica

de València, Spain

jcalvo@prhlt.upv.es

David Rizo

Instituto Superior de Enseñanzas Artísticas

de la Comunidad Valenciana (ISEA.CV)

Universidad de Alicante, Spain

drizo@dlsi.ua.es

ABSTRACT

The optical music recognition (OMR) field studies how to automate the process of reading the musical notation present in a given image. Among its many uses, an interesting scenario is that in which a score captured with a camera is to be automatically reproduced. Recent approaches to OMR have shown that the use of deep neural networks allows important advances in the field. However, these approaches have been evaluated on images with ideal conditions, which do not correspond to the previous scenario. In this work, we evaluate the performance of an end-to-end approach that uses a deep convolutional recurrent neural network (CRNN) over non-ideal image conditions of music scores. Consequently, our contribution also consists of Camera-PrIMuS, a corpus of printed monophonic scores of real music synthetically modified to resemble camera-based realistic scenarios, involving distortions such as irregular lighting, rotations, or blurring. Our results confirm that the CRNN is able to successfully solve the task under these conditions, obtaining an error around 2% at music-symbol level, thereby representing a groundbreaking piece of research towards useful OMR systems.

1. INTRODUCTION

The optical music recognition (OMR) discipline was born several decades ago [28], and nowadays there are still too many open problems to consider it a solved task. This applies not only for handwritten notation but also for the case of printed scores [4]. Unfortunately, unlike other automatic content transcription domains, such as speech recognition [23] or optical character recognition [24], the latest advances in pattern recognition and machine learning—namely deep learning—have not definitively broken the long-term glass ceiling.

Actually, other computer music domains are taking advantage of these advances, but quite often, especially in symbolic music research, the lack of big enough datasets

block their improvement. If OMR technologies were able to convert the massive printed scores libraries¹ into structured, symbolic scores, all those fields would obtain interesting corpora to work on.

Furthermore, out of the scientific community, the availability of tools that transcribe sheet music without errors into symbolically-encoded music would help professional and amateur musicians to take advantage of the plenty of computer music tools at hand that cannot work directly with digital images.

Following the steps of other aforementioned disciplines, we claim that the problem can be appropriately addressed with holistic approaches, i.e., end-to-end, where systems learn with just pairs of inputs and their corresponding transcripts. Here, these pairs consist of sheet music and their symbolic encoding.

In this work, we extend previous proposals that applied neural network models over monodic digitally-rendered music scores [8]. However, we evaluate here their performance with a set of scores that are rendered simulating camera-based conditions. Our objective is to study whether the approach is feasible for non-ideal image conditions. Although we do not experiment with fully-fledged scores yet, we believe that this avenue is promising for reaching the final objective of dealing with any kind of input score. Thus, in this work we introduce the so-called *Camera-Printed Images of Music Staves* (Camera-PrIMuS) dataset of monodic single-staff printed scores, that have been distorted to resemble photographed scores and encoded in such a way a neural network recognizer can manage.

Our experiments demonstrate that the considered neural models are able to learn even in difficult situations where none of the current commercial OMR systems might be successful. The results reflect that an error rate below 2%, at symbol level, can be attained.

The paper is organized as follows: first, a brief background about OMR is given in Sect. 2; then, the construction of Camera-PrIMuS dataset is detailed in Sect. 3; the neural end-to-end framework is described and formalized in Sect. 4; the experimental results that demonstrate the suitability of the approach are reported in Sect. 5; and finally, the conclusions are discussed in Sect. 6.



© Jorge Calvo-Zaragoza, David Rizo. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: Jorge Calvo-Zaragoza, David Rizo. "Camera-PrIMuS: Neural End-to-End Optical Music Recognition on Realistic Monophonic Scores", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ Libraries such as <http://imslp.org>

2. BACKGROUND

Most of the existing OMR approaches work in a multi-stage fashion [38]. These systems typically perform an initial processing of the image that consists of several steps of document analysis, not always strictly related to the musical domain. Examples of this stage comprise the binarization of the image [10], the detection of the staves [11], the delimitation of the staves in terms of bars [45], or the separation among the different sources of information [5].

The staff-line removal stage requires a special mention. Although staff lines represent a very important element in music notation, their presence hinders the automatic segmentation of musical symbols. Therefore, much effort has been devoted to successfully solving this stage [14, 15, 18]. Recently, results have reached values closer to the optimum over standard benchmarks [7, 17].

In the next step, remaining symbols are classified into music-notation categories. A number of works can be found in the literature that deal with this task [30, 37], including deep learning classification as well [6, 32].

Recently, it has been demonstrated that the traditional pipeline up to symbol classification can be replaced by deep region-based neural networks [31], which both localize and classify music-notation primitives from the input image. Either way, once graphical symbol are identified, they must be assembled to eventually obtain actual music notation. Previous attempts to this stage proposed the use of heuristic strategies based on graphical and syntactical rules [13, 36, 40, 43].

Full approaches are more common when recognizing mensural notation, where the OMR challenge is more restricted than that of modern Western notation because of the absence of simultaneous written voices in the same staff and a lower number of symbols to be recognized [9, 33, 44].

3. THE CAMERA-PRIMUS DATASET

The training of a machine learning based system requires a good quality training dataset with enough size to statistically include a representative sample of the problem to be solved. The *Camera-based Printed Images of Music Staves* (Camera-PrIMuS) dataset has been devised to fulfil both requirements². Thus, the objective pursued when creating this ground-truth data is not to represent the most complex musical notation corpus, but to collect the highest possible number of scores readily available to be represented in formats suitable for heterogeneous OMR experimentation and evaluation.

Camera-PrIMuS is an extension of a previously published PrIMuS dataset [8]. It contains 87 678 real-music incipits,³ each one represented by six files: the Plaine and Easie Code (PAEC) source [3], an image with the rendered score, the same image distorted resembling a camera-based scenario, the music symbolic representation of the incipit

² The dataset is freely available at <https://grfia.dlsi.ua.es/primus/>.

³ An incipit is a short sequence of notes from the beginning of a melody or musical work usually used for identifying it

Order	Filter	Ranges of used parameters
1	-implode	[0, 0.07]
2	-chop	[1, 5], [1, 6], [1, 300], [1, 50]
3	-swirl	[-3, 3]
4	-spread	-2
5	-shear	[-5, 5], [-1.5, 1.5]
6	-shade	[0, 120], [80, 110]
7	-wave	[0, 0.5], [0, 0.4]
8	-rotate	[0, 0.3]
9	-noise	[0, 1.2]
10	-wave	[0, 0.5], [0, 0.4]
11	-motion-blur	[-7, 5], [-7, 7], [-7, 6]
12	-median	[0, 1.1]

Table 1. GraphicsMagick filter sequence

both in Music Encoding Initiative format (MEI) [39] and in an on-purpose simplified encoding (semantic encoding), and a sequence containing the graphical symbols shown in the score with their position in the staff, without any musical meaning (agnostic encoding). These two agnostic and semantic representations, that will be described below, are especially designed to be considered in our framework.

Pursuing the objective of considering real music, and being restricted to use short single-staff scores, an export in PAEC format of the RISM dataset [29] has been used as source. The PAEC is then formatted to be fed into the musical engraver Verovio [34], that outputs both the musical score in SVG format—that is posteriorly converted into PNG format (Fig. 1(a))—and the MEI encoding containing the symbolic semantic representation of the score in XML format. Verovio is able to render scores using three different fonts, namely: Leipzig, Bravura, and Gootville. This capability has been used by randomly choosing one of the those fonts in the rendering of the different incipits, leading to a higher variability in the dataset. The on-purpose semantic and agnostic representations (Figs. 1(c) and 1(d)) have been obtained as a conversion from the MEI files. Finally, the PNG image file is distorted, as described below, in order to simulate imperfections introduced by taking a picture of the sheet music from a (bad) camera (Fig. 1(b)).

To simulate distortions, the GraphicsMagick image processing tool⁴ has been used. Among the huge amount of filters this tool contains, a number of them have been used and tweaked empirically. Table 1 contains the filters used and the ranges considered for each parameter, from which random values are selected at each instance. Filters have been applied using the order shown in the table.

3.1 Semantic and agnostic representations

The suitable encoding of input data for the neural network determines the scope of its performance. Most of the available symbolic representations [41], being devised for other purposes such as music analysis (e.g. ***kern*), or music

⁴ <http://www.graphicsmagick.org>

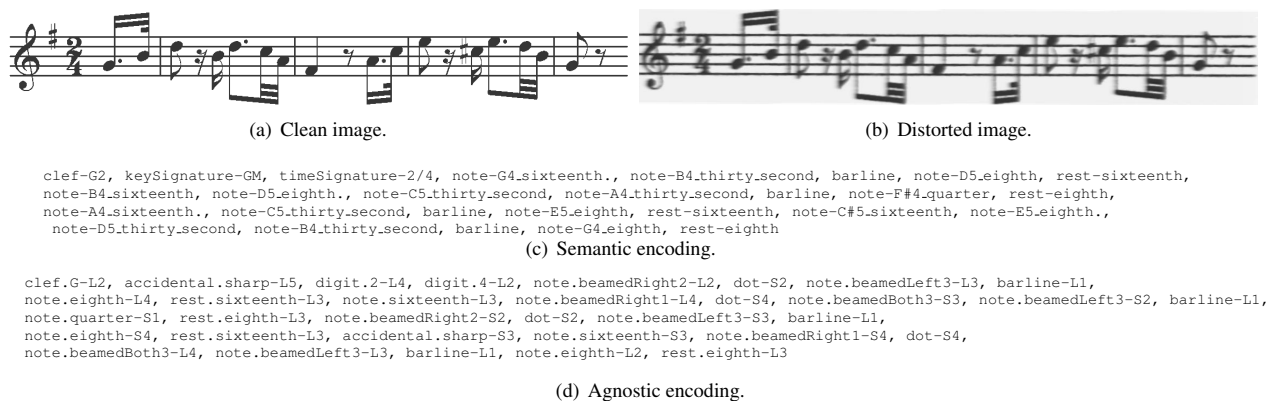


Figure 1. Example of a short item in the corpus: Incipit RISM ID no. 000100367, Incipit 28.1.1 30 *Canons*, Luigi Cherubini. MEI and Plaine and Easie Code files are also included in the corpus but omitted here.

notation (such as MEI [39] or MusicXML [20])—for naming just a few—do not encode a self-contained chunk of information for each musical element. This is why two representations devised on-purpose compliant with this requirement were introduced in [8], namely the semantic and the agnostic ones. For practical issues, none of the representations is musically exhaustive, but representative enough to serve as a starting point from which to build more complex systems.

The semantic representation contains symbols with musical meaning, *e.g.*, a G Major key signature (see Fig. 1(c)); the agnostic encoding (see Fig. 1(d)) consists of musical symbols without musical meaning that should be eventually interpreted in a final parsing stage [16], *e.g.* a D Major key signature is represented as a sequence of two *sharp* symbols. This way, the alphabet used for the agnostic representation is much smaller, which allows to study the impact of the alphabet size and the number of examples shown to the network for its training. Note that in the agnostic representation, a *sharp* symbol in the key signature is the same pictogram as a *sharp* accidental altering the pitch of a note. A complete description of the grammars describing these encodings can be found in [8].

More specifically, the agnostic representation contains a list of graphical symbols in the score, each of them tagged given a catalogue of pictograms without a predefined musical meaning, and located in a position in the staff (*e.g.*, third line, first space). The Cartesian plane position of symbols has been encoded relatively, following a left-to-right, top-down ordering (see encoding of fractional meter in Fig. 1(d)). In order to represent beaming of notes, they have been vertically sliced generating non-musical pictograms (see elements with prefix *note.beamed* in Fig. 1(d)).

As mentioned above, this new way of encoding complex information in a simple sequence allows us to feed the network in a relatively easy way. Note that the agnostic representation is different from a primitive-based segmentation of the image, which is the usual internal representation of traditional OMR systems [12, 25].

The agnostic representation has an additional advantage: in other less known music notations, such as the early neumatic and mensural notations, or in the case of non-Western notations, it might be easier to transcribe the manuscript through two stages: one stage performed by any non-musical expert that only needs to identify pictograms (agnostic representation), and a second stage where a musicologist, maybe aided by a computer, interprets them to yield a semantic encoding.

4. NEURAL END-TO-END APPROACH FOR OPTICAL MUSIC RECOGNITION

As introduced above, some previous work have proved that it is possible to successfully accomplish the recognition of monodic staves in an end-to-end approach by using neural networks [8]. This section contains a brief description of such framework.

A single-voice monophonic staff is assumed to be the basic unit; that is, a single monodic staff will be processed at each instance. Formally, let $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \dots\}$ be our end-to-end application domain, where x_i represents a single staff image and y_i is its corresponding sequence of music symbols, each of which belongs to a fixed alphabet set Σ .

Given an input staff image, the OMR problem can be solved by retrieving its most likely sequence of music symbols \hat{y} :

$$\hat{y} = \arg \max_{y \in \Sigma^*} P(y|x) \quad (1)$$

A graphical scheme of the considered framework is given in Figure 2. The input image depicting a monodic staff is fed into a Convolutional Recurrent Neural Network (CRNN), which consists of two sequential parts: a convolutional block and a recurrent block. The convolutional block is in charge of learning how to deal with the input image [47]. In this way, the user is prevented from performing a pre-processing of the image because this block is able to learn adequate features from the training set. These

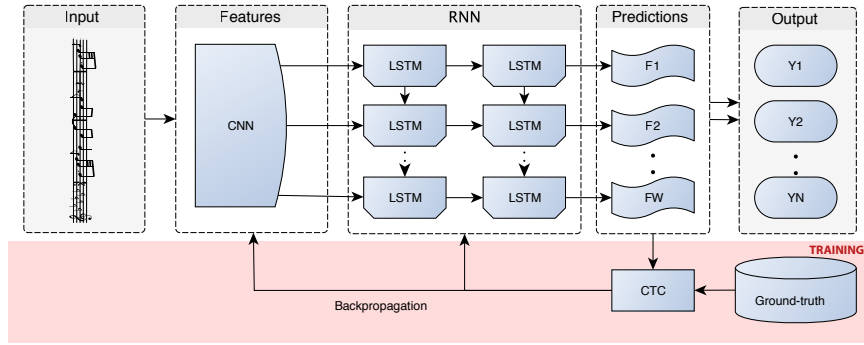


Figure 2. Graphical scheme of the end-to-end neural approach considered.

extracted features are provided to the recurrent block [21], producing the sequence of musical symbols that approximates Eq. 1.

Since both convolutional and recurrent blocks are configured as feed-forward models, the training stage can be carried out jointly. This scheme can be easily implemented by connecting the output of the last layer of the convolutional block with the input of the first layer of the recurrent block, concatenating all the output channels of the convolutional part into a single image. Then, columns of the resulting image are treated as individual frames for the recurrent block.

The traditional training mechanisms for a CRNN need a framewise expected output, where a frame is a fixed-width vertical slice of the image. However, as the goal is to not recognize frames but complete symbols, either semantic or agnostic, and Camera-PriMuS does not contain sequences of labelled frames, a Connectionist Temporal Classification (CTC) loss function [22] has been used to solve this mismatch.

Basically, CTC drives the CRNN to optimize its parameters so that it is likely to give the correct sequence y given an input x . As optimizing this likelihood exhaustively is computationally expensive, CTC performs a local optimization using an Expectation-Maximization algorithm similar to that used for training Hidden Markov Models [35]. Note that CTC is only used for training, while at the decoding stage the framewise CRNN output can be straightforwardly decoded into a sequence of music symbols (details are given below).

4.1 Implementation details

The specific organization of the neural model is given in Table 2. As observed, variable-width single-channel (grayscale) input image are rescaled at a fixed height of 128 pixels, without modifying their aspect ratio. This input is processed through a convolutional block inspired by a VGG network, a typical model in computer vision tasks [42]: four convolutional layers with an incremental number of filters and kernel sizes of 3×3 , followed by a 2×2 max-pool operator. In all cases, Batch Normalization [27] and Rectified Linear Unit activations [19] are considered.

Input($128 \times W \times 1$)
Convolutional block
Conv($32, 3 \times 3$), MaxPooling(2×2)
Conv($64, 3 \times 3$), MaxPooling(2×2)
Conv($128, 3 \times 3$), MaxPooling(2×1)
Conv($256, 3 \times 3$), MaxPooling(2×1)
Recurrent block
BLSTM(256)
BLSTM(256)
Dense($ \Sigma + 1$)
Softmax()

Table 2. Instantiation of the CRNN used in this work, consisting of 4 convolutional layers and 2 recurrent layers. Notation: Input($h \times w \times c$) means an input image of height h , width w and c channels; Conv($n, h \times w$) denotes a convolution operator of n filters and kernel size of $h \times w$; MaxPooling($h \times w$) represents a down-sampling operation of the dominating value within a window of size ($h \times w$); BLSTM(n) means a bi-directional Long Short-Term Memory unit of n neurons; Dense(n) denotes a dense layer of n neurons; and Softmax() represents the *softmax* activation function. Σ denotes the alphabet of musical symbols considered.

At the output of this block, two bidirectional recurrent layers of 256 neurons, implemented as Long Short-Term Memory (LSTM) units [26], try to convert the resulting filtered image into a discrete sequence of musical symbols that takes into account both the input sequence and the modelling of the musical representation. Note that each frame performs an independent classification, modelled with a fully-connected layer with as many neurons as the size of the alphabet plus 1 (a *blank* symbol necessary for the CTC function). The activation of these neurons is given by a *softmax* function, which allows interpreting the output as a posterior probability over the alphabet of music symbols [2].

The learning process is carried out by means of stochastic gradient descent (SGD) [1], which modifies the CNN parameters through back-propagation to minimize the

CTC loss function. In this regard, the mini-batch size is established to 16 samples per iteration. The learning rate of the SGD is updated adaptively following the Adadelta algorithm [46].

Once the network is trained, it is able to provide a prediction in each frame of the input image. These predictions must be post-processed to emit the actual sequence of predicted musical symbols. Thanks to training with the CTC loss function, the final decoding can be performed greedily [22]: when the symbol predicted by the network in a frame is the same as the previous one, it is assumed that they represent frames of the same symbol, and only one symbol is concatenated to the final sequence. There are two ways to indicate that a new symbol is predicted: either the predicted symbol in a frame is different from the previous one, or the predicted symbol of a frame is the *blank* symbol, which indicates that no symbol is actually found.

Thus, given an input image, a discrete musical symbol sequence is obtained. Note that the only limitation is that the output cannot contain more musical symbols than the number of frames of the input image, which in our case is highly unlikely to happen.

5. EXPERIMENTS

5.1 Experimental setup

Once introduced the Camera-PrIMuS dataset, and a model able to learn the OMR task from it, some experiments have been performed whose results may serve as a baseline to which other works can be compared.⁵

Currently, there is an open debate on which evaluation metrics should be used in OMR [4]. This is especially arguable because of the different points of view that the use of its output has: it is not the same whether the intention of the OMR is to automatically play the content or to archive it in a digital library. Here we are only interested in the computational aspect itself. Hence, we shall consider metrics focused on the symbol and sequence recognition, avoiding any music-specific consideration, such as:

- Sequence Error Rate (ER) (%): ratio of incorrectly predicted sequences (at least one error).
- Symbol Error Rate (SER) (%): the average number of elementary editing operations (insertions, deletions, or substitutions) needed to produce the reference sequence from the one predicted by the model, normalized by its length.

Note that the length of the agnostic and semantic sequences are usually different because they are encoding different aspects of the same source. Therefore, the comparison in terms of Symbol Error Rate, in spite of being normalized, may not be totally fair. On the other hand, the Sequence Error Rate allows a more reliable comparison because it only takes into account the perfectly pre-

dicted sequences (in which case, the outputs in different representations are equivalent).

5.2 Performance

We show in this section the results obtained in our experiments. We consider three different data partitions: 80% of the data is used as training set, to optimize the network according to the CTC loss function; 10% of the data is used as validation set, which is used to decide when to stop the optimization to prevent over-fitting; the evaluation results are computed with the remaining 10%, which constitutes the test partition.

In order to study the ability of the system to learn in different situations, four scenarios have been evaluated depending upon which set of images are used for training and testing, either the clean original files or the synthetically distorted ones. We report in Table 3 the whole evaluation.

The results show that the system, trained with the appropriate set, is able to correctly recognize in almost all scenarios, with error rates at symbol level below 2%. In an ideal scenario, where only clean images are given, the semantic encoding outperforms the agnostic one. The behaviour is different when distorted images are used, for which the agnostic representations behave much better. What seems most interesting from these results is the ability of the system to learn from distorted images and correctly classify both distorted and clean versions. This leads us to conclude that the networks are being able to abstract the content from the image condition. As a qualitative example of the performance attained, the sample of Figure 1 was correctly classified using both encodings.

In an informal analysis, we observed that the most repeated error, both in agnostic and semantic encodings, is the incorrect classification of the ending bar line. In addition to it, no other repeating mistake has been found. Also, we checked that most of the wrongly recognized samples only failed at 1 symbol. Another interesting feature to emphasize is that we observed an independence of the mistakes with respect to the length of the ground-truth sequence, i.e., errors are not accumulated and, therefore, the number of mistakes do not necessarily increase with longer sequences. Figures 3 and 4 depict two examples of wrongly recognized sequences.

6. CONCLUSIONS

The suitability of a neural network approach to solve the OMR task in an end-to-end fashion has been evaluated on realistic single-staff printed monodic scores from a real world dataset. To this end, the new Camera-PrIMuS dataset has been introduced, containing 87 678 images synthetically distorted to resemble a camera-based scenario.

The neural network model considered consists of a CRNN, in which convolutions process the input image and recurrent blocks deal with the sequential nature of the problem. In order to train this model directly using symbol sequences, instead of fine-grained annotated images, the so-called CTC loss function has been utilized.

⁵For the sake of reproducible research, source code and trained models are available at <https://github.com/calvozaragoza/tf-deep-omr>.

		Evaluation			
		Clean		Distortions	
		Agnostic	Semantic	Agnostic	Semantic
Training	Clean	1.1 / 21.7	0.8 / 12.5	44.3 / 94.1	59.7 / 97.9
	Distortions	1.4 / 24.9	3.3 / 44.6	1.6 / 24.7	3.4 / 38.3

Table 3. Average SER (%) / ER (%) reported in all possible combinations of training and evaluation conditions.



(a) Distorted image file of Incipit RISM ID no. 000104754, Incipit 1.1.1 *Achille in Sciro. Excerpts. Niccolò Jommelli.*

clef-G2, keySignature-DM, timeSignature-C, note-D5.half, tie, note-D5.quarter., note-F#4.eighth, barline, note-G4.half, note-F#4.quarter, rest-quarter, barline, note-B4.eighth, rest-eighth, note-A4.eighth, rest-eighth, note-B4.half, **[rest-eighth-L3]** note-E5.eighth., note-C#5.sixteenth, barline, note-F#5.half, tie, note-F#5.quarter., note-F#4.eighth, barline, note-G4.half, note-F#4.quarter, rest-quarter, barline

(b) Semantic encoding network output. The symbol in *italics* should be classified as note-B4.eighth, and the bold symbol between brackets has been omitted by the network.

clef-G-L2, accidental.sharp-L5, accidental.sharp-S3, metersign.C-L3, note.half-L4, slur.start-L4, slur.end-L4, note.quarter-L4, dot-S4, note.eighth-S1, barline-L1, note.half-L2, note.quarter-S1, rest.quarter-L3, barline-L1, note.eighth-L3, rest.eighth-L3, note.eighth-S2, rest.eighth-L3, *fermata.above-S6*, note.quarter-L3, note.beamedRight1-S4, dot-S4, note.beamedLeft2-S3, barline-L1, note.half-L5, slur.start-L5, slur.end-L5, note.quarter-L5, dot-S5, note.eighth-S1, barline-L1, note.half-L2, note.quarter-S1, rest.quarter-L3, barline-L1

(c) Agnostic encoding network output. Wrong symbols have been highlighted in italic face symbols. They should be note.eighth-L3 and rest.eighth-L3, respectively.

Figure 3. This incipit contains distortions that are very hard to recognize, such as the scratch at the beginning of the staff and some overlapped ink. Despite these difficulties, just two symbols in each encoding have been wrongly recognized.



(a) Distorted image file of Incipit RISM ID no. 000100170, Incipit 1.1.1 *Trios. Joseph Haydn.*

clef-G2, keySignature-FM, timeSignature-C, note-F4.quarter, rest-quarter, rest-eighth, note-A4.sixteenth, note-Bb4.sixteenth, note-C5.eighth, note-C5.eighth, barline, note-C5.eighth, note-F5.eighth, note-A4.eighth, note-A4.eighth, note-A4.eighth, note-C5.eighth, note-F4.eighth, note-F4.eighth, barline, note-E4.eighth, note-D4.eighth, note-D4.quarter, tie, note-D4.eighth, note-C5.sixteenth, note-Bb4.sixteenth, note-A4.sixteenth, note-G4.sixteenth, note-F4.sixteenth, *note-D4.thirty.second*, barline

(b) Semantic encoding network output. The italic font face symbol should be classified as a sixteenth note.

clef-G-L2, accidental.flat-L3, metersign.C-L3, note.quarter-S1, rest.quarter-L3, rest.eighth-L3, note.beamedRight2-S2, note.beamedLeft2-L3, note.beamedRight1-S3, note.beamedLeft1-S3, barline-L1, note.beamedRight1-S3, note.beamedBoth1-L5, note.beamedBoth1-S2, note.beamedLeft1-S2, note.beamedRight1-S2, note.beamedBoth1-S3, note.beamedBoth1-S1, note.beamedLeft1-S1, barline-L1, note.beamedRight1-L1, note.beamedLeft1-S0, note.quarter-S0, slur.start-S0, slur.end-S0, note.beamedRight1-S0, note.beamedBoth2-S3, note.beamedLeft2-L3, note.beamedRight2-S2, note.beamedBoth2-L2, note.beamedBoth2-S1, note.beamedLeft2-S0, barline-L1

(c) Agnostic encoding network output. All symbols are correctly detected.

Figure 4. Incipit correctly recognized using the agnostic representation but with one mistake using the semantic encoding.

Our experiments have reflected the correct construction and the usefulness of the corpus. The end-to-end neural optical recognition model has demonstrated its ability to learn from adverse conditions and to correctly classify both perfectly clean images and imperfect pictures. In regard to the output encoding, the agnostic representation has been shown to be more robust against the image distortions, while semantic encoding maintains a fair performance.

Given these promising results, from the musical point of view, the next steps seem obvious: first, we would like to complete the catalogue of symbols, thus including chords and multiple-voice polyphonic staves. In the long-term, the intention is to consider fully-fledged real piano or orchestral scores. Concerning the most technical aspect, it would be interesting to study a multi-prediction model that uses

all the different representations at the same time. Given the complementarity of the agnostic and semantic representations, it is feasible to think of establishing a synergy that ends up with better results in all senses.

7. ACKNOWLEDGEMENT

This work was partially supported by the Spanish Ministerio de Economía, Industria y Competitividad through HispaMus project (TIN2017-86576-R) and Juan de la Cierva - Formación grant (Ref. FJCI-2016-27873), and the Social Sciences and Humanities Research Council of Canada.

8. REFERENCES

- [1] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.
- [2] H. Bourlard and C. Wellekens. Links between markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1167–1178, 1990.
- [3] B. Brook. The Simplified 'Plaine and Easie Code System' for Notating Music: A Proposal for International Adoption. *Fontes Artis Musicae*, 12(2-3):156–160, 1965.
- [4] D. Byrd and J. G. Simonsen. Towards a Standard Testbed for Optical Music Recognition: Definitions, Metrics, and Page Images. *Journal of New Music Research*, 44(3):169–195, 2015.
- [5] J. Calvo-Zaragoza, F. J. Castellanos, G. Vigliensoni, and I. Fujinaga. Deep neural networks for document processing of music score images. *Applied Sciences*, 8(5):654–674, 2018.
- [6] J. Calvo-Zaragoza, A.-J. Gallego, and A. Pertusa. Recognition of handwritten music symbols with convolutional neural codes. In *14th IAPR International Conference on Document Analysis and Recognition*, pages 691–696, 2017.
- [7] J. Calvo-Zaragoza, A. Pertusa, and J. Oncina. Staff-line detection and removal using a convolutional neural network. *Machine Vision & Applications*, 28(5-6):665–674, 2017.
- [8] J. Calvo-Zaragoza and D. Rizo. End-to-end neural optical music recognition of monophonic scores. *Applied Sciences*, 8(4):606–629, 2018.
- [9] J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal. Early handwritten music recognition with hidden markov models. In *15th International Conference on Frontiers in Handwriting Recognition*, pages 319–324, 2016.
- [10] J. Calvo-Zaragoza, G. Vigliensoni, and I. Fujinaga. Pixel-wise binarization of musical documents with convolutional neural networks. In *Fifteenth IAPR International Conference on Machine Vision Applications*, pages 362–365, 2017.
- [11] V. B. Campos, J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal-Ruiz. Sheet music statistical layout analysis. In *15th International Conference on Frontiers in Handwriting Recognition*, pages 313–318, 2016.
- [12] L. Chen, E. Stolterman, and C. Raphael. Human-Interactive Optical Music Recognition. In *17th International Society for Music Information Retrieval Conference*, pages 647–653, 2016.
- [13] B. Couasnon. Dmos: A generic document recognition method, application to an automatic generator of musical scores, mathematical formulae and table structures recognition systems. In *6th International Conference on Document Analysis and Recognition*, pages 215–220, 2001.
- [14] C. Dalitz, M. Droettboom, B. Pranzas, and I. Fujinaga. A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):753–766, 2008.
- [15] J. Dos Santos Cardoso, A. Capela, A. Rebelo, C. Guedes, and J. Pinto da Costa. Staff Detection with Stable Paths. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1134–1139, 2009.
- [16] H. Fahmy and D. Blostein. A graph grammar programming style for recognition of music notation. *Machine Vision and Applications*, 6(2-3):83–99, March 1993.
- [17] A. Gallego and J. Calvo-Zaragoza. Staff-line removal with selectional auto-encoders. *Expert Systems with Applications*, 89:138–48, 2017.
- [18] T. Géraud. A morphological method for music score staff removal. In *21st International Conference on Image Processing*, pages 2599–2603, Paris, France, 2014.
- [19] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.
- [20] M. Good and G. Actor. Using MusicXML for File Interchange. *International Conference on Web Delivering of Music*, page 153, 2003.
- [21] A. Graves. *Supervised sequence labelling with recurrent neural networks*. PhD thesis, Technical University Munich, 2008.
- [22] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *23rd International Conference on Machine Learning*, pages 369–376, 2006.
- [23] A. Graves, A.-R. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.
- [24] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.
- [25] J. Hajic and P. Pecina. The MUSCIMA++ dataset for handwritten optical music recognition. In *14th IAPR International Conference on Document Analysis and Recognition*, pages 39–46, 2017.

- [26] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [27] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *32nd International Conference on Machine Learning*, pages 448–456, 2015.
- [28] M. Kasser. Optical character-recognition of printed music: A review of two dissertations. *Perspectives of New Music*, 11(1):250–254, 1972.
- [29] K. Keil and J. A. Ward. Applications of RISM data in digital libraries and digital musicology. *International Journal on Digital Libraries*, 50(2):199, January 2017.
- [30] S. Lee, S. J. Son, J. Oh, and N. Kwak. Handwritten music symbol classification using deep convolutional neural networks. In *3rd International Conference on Information Science and Security*, 2016.
- [31] A. Pacha, K.-Y. Choi, B. Couïasnon, Y. Ricquebourg, R. Zanibbi, and H. Eidenberger. Handwritten music object detection: Open issues and baseline results. In *13th IAPR Workshop on Document Analysis Systems*, 2018.
- [32] A. Pacha and H. Eidenberger. Towards a universal music symbol classifier. In *12th International Workshop on Graphics Recognition*, pages 35–36, 2017.
- [33] L. Pugin. Optical music recognition of early typographic prints using hidden markov models. In *7th International Conference on Music Information Retrieval*, pages 53–56, 2006.
- [34] L. Pugin, R. Zitellini, and P. Roland. Verovio - A library for Engraving MEI Music Notation into SVG. In *International Society for Music Information Retrieval*, 2014.
- [35] L. Rabiner and B.-H. Juang. *Fundamentals of speech recognition*. Prentice hall, 1993.
- [36] C. Raphael and J. Wang. New Approaches to Optical Music Recognition. In *12th International Society for Music Information Retrieval Conference*, pages 305–310, 2011.
- [37] A. Rebelo, A. Capela, and J. S. Cardoso. Optical recognition of music symbols: A comparative study. *International Journal on Document Analysis and Recognition*, 13(1):19–31, March 2010.
- [38] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. S. Marçal, C. Guedes, and J. S. Cardoso. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190, 2012.
- [39] P. Roland. The music encoding initiative (MEI). In *Proceedings of the First International Conference on Musical Applications Using XML*, pages 55–59, 2002.
- [40] F. Rossant and I. Bloch. Robust and adaptive omr system including fuzzy modeling, fusion of musical rules, and possible error detection. *EURASIP Journal on Advances in Signal Processing*, 081541, 2007.
- [41] E. Selfridge-Field. *Beyond MIDI: The handbook of musical codes*. MIT Press, 1997.
- [42] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [43] M. Szwoch. Guido: A musical score recognition system. In *9th International Conference on Document Analysis and Recognition*, pages 809–813, 2007.
- [44] L. J. Tardón, S. Sammartino, I. Barbancho, V. Gómez, and A. Oliver. Optical music recognition for scores written in white mensural notation. *EURASIP Journal on Image and Video Processing*, 2009.
- [45] G. Vigliensoni, G. Burlet, and I. Fujinaga. Optical measure recognition in common music notation. In *14th International Society for Music Information Retrieval Conference*, pages 125–30, 2013.
- [46] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.
- [47] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *13th European Conference on Computer Vision — Part I*, pages 818–833, 2014.

DOCUMENT ANALYSIS OF MUSIC SCORE IMAGES WITH SELECTIONAL AUTO-ENCODERS

Francisco J. Castellanos¹
Gabriel Vigliensoni³

Jorge Calvo-Zaragoza²
Ichiro Fujinaga³

¹ Software and Computing Systems, University of Alicante, Spain

² PRHLT Research Center, Universitat Politècnica de València, Spain

³ Schulich School of Music, McGill University, Canada

fcastellanos@dlsi.ua.es

ABSTRACT

The document analysis of music score images is a key step in the development of successful Optical Music Recognition systems. The current state of the art considers the use of deep neural networks trained to classify every pixel of the image according to the image layer it belongs to. This process, however, involves a high computational cost that prevents its use in interactive machine learning scenarios. In this paper, we propose the use of a set of deep selectional auto-encoders, implemented as fully-convolutional networks, to perform image-to-image categorizations. This strategy retains the advantages of using deep neural networks, which have demonstrated their ability to perform this task, while dramatically increasing the efficiency by processing a large number of pixels in a single step. The results of an experiment performed with a set of high-resolution images taken from Medieval manuscripts successfully validate this approach, with a similar accuracy to that of the state of the art but with a computational time orders of magnitude smaller, making this approach appropriate for being used in interactive applications.

1. INTRODUCTION

The Optical Music Recognition (OMR) is a computational process that reads musical notation from images, with the aim of automatically exporting the content to a structured format [1]. Given the complexity of the task, the process is usually divided into different stages, the first of which is the document analysis. This stage consists of detecting and categorizing the different sources of information that appear in images of musical scores—e.g., classifying each pixel into one of four possible categories: background, staff line, musical note, or lyrics—and it is important for creating robust OMR systems [29]. That is, if subsequent

stages receive the image in a reliable state, systems tend to generalize more easily.

Many researchers have proposed different algorithms to deal with specific steps within the document processing stage of the Optical Music Recognition (OMR) workflow. Traditionally, these strategies consist of heuristic workflows specifically designed for the scores at hand, exploiting specific details of the images to improve the performance of the detection. Music documents, however, especially from the Medieval and Renaissance era, come in a wide variety of notational styles and formats, resulting in a heterogeneous collection. Therefore, the previous approaches may be beneficial in the short term but they do not scale well [4, 6]. In many cases, a workflow must be developed anew for dealing with manuscripts with different notation, from a disparate time period, or with a differing level of image degradation.

Recent work has demonstrated the feasibility of using machine learning for document analysis [21, 25, 36]. In comparison to systems with hand-crafted heuristic rules, the advantage of using machine learning-based techniques lies in their generalizability, only needing labeled examples to build a new classification model [12]. In addition to this important advantage, the use of these techniques, in particular Convolutional Neural Networks (CNN), has proven to outperform the traditional strategies considered for document analysis in the OMR domain [6]. The main idea behind this approach is training a CNN to distinguish the category to which each pixel of the image belongs. That is, given a pixel of the image, and taking into account the pixels of its neighborhood, a model is trained to predict the category (e.g., note, staff line, and lyrics). In this way, the document analysis process consists of classifying every single pixel of the image into its actual category, thus separating the different layers of the document accordingly. Given that the classification is performed at pixel level, thin elements such as staff lines, note stems, as well as small artifacts, can be properly detected.

The problem with the aforementioned process is that it entails a high computational cost because it needs to classify every single pixel of an image. Since OMR is a process that lends itself to be used interactively [8, 9], there is a need of accelerating the processing of documents without sacrificing the classification quality, in order to present



© Francisco J. Castellanos, Jorge Calvo-Zaragoza, Gabriel Vigliensoni, Ichiro Fujinaga. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Francisco J. Castellanos, Jorge Calvo-Zaragoza, Gabriel Vigliensoni, Ichiro Fujinaga. “Document Analysis of Music Score Images with Selectional Auto-Encoders”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

a user-friendly environment.

We present in this paper a new framework based on machine learning that replaces the current pixel-wise model by a patch-wise model. In this approach, we process a complete sub-image (patch) in a single step, making predictions of many pixels simultaneously. This can be carried out by means of neural networks that learn how to compute an image-to-image prediction.

We evaluate the new approach over a set of high-resolution images taken from Medieval music manuscripts. The patch-wise model attains a similar accuracy to that of the state of the art but reducing the computational cost by several orders of magnitude.

The rest of the paper is organized as follows. We give a brief review of related work in Section 2. A formalization of the task, as well as the proposed solution, is detailed in Section 3. We empirically demonstrate in Section 4 that our model drastically reduces the computational time by orders of magnitude without the classification quality. Finally, we summarize the main conclusions of the present work in Section 5, pointing out some potential future work.

2. BACKGROUND

The classical workflow for OMR considers an initial document analysis stage [29], to process the input image before proceeding to the automatic recognition of the content. This first stage is crucial to increase the robustness of the system and to reduce the complexity of subsequent stages by providing correctly segmented images.

A common first step within the document analysis stage is binarization, in which background and foreground layers are separated. In addition to typical document image binarization techniques [15, 19, 30], some music-specific document binarization techniques have been proposed [28, 35]. Next, if the lyrics are part of the musical content, they need to be recognized as well. This is why there have been some proposals to separate the staves and the text [3, 7]. Once staff sections have been isolated, staff-line removal may take place. Although staff lines are necessary for music interpretation, most OMR workflows are based on detecting and removing the staff lines to perform connected component analysis on the remaining musical symbols. A comprehensive review and comparison of the first attempts for staff-line removal can be consulted in Dalitz et al. [10], and new techniques are being continuously developed [11, 13, 16]. In addition to these stages, we also find very specific processes that depend on the specific characteristics of the manuscript of interest, such as measure isolation [33], page-border removal [26], or frontispiece detection in Medieval manuscripts [31].

Recently, the full document processing of music score images has been implemented using CNNs, which learn to classify each pixel of the image according to its category [6]. This approach allows the analysis of entire documents with a generic method to any type of manuscript as long as there is appropriate training data. In addition to these advantages, this approach has proven to outperform the traditional strategies, and so it can be considered the

state of the art in document analysis of music score images.

However, this process takes a long time because it has to perform an independent classification for each pixel of the image. Since images used are usually at high resolution involving millions of pixels, the resulting long computational time prevents its use in an interactive machine learning environment, where the user expects quick responses from the machine learning process while training it. Hence, in this work, we propose an image-to-image approach using neural networks, with the aim of maintaining the advantages of the state of the art but dramatically reducing the temporal cost.

3. FRAMEWORK

Formally, we define the task of document analysis of music score images as the process of assigning a category to each pixel of the image based on the layer of information to which it belongs. Specifically, we instantiate the task to the set of categories $\{\text{background, note, staff line, text}\}$. The reasoning behind this set is that it consists of the layers that lead to a general analysis of the image for the purpose of OMR, given that: musical notes are essential to recover the musical information; staff lines are necessary to divide the score into staves, as well as to estimate the pitch of the notes; text is also key for music interpretation but its information must be recognized with different algorithms (i.e., Optical Character Recognition); the rest of pixels can be considered as background. However, we show below that the chosen formulation can be extended to any other type of category set provided that sufficient labeled data is available.

As mentioned above, the aim of this work is to alleviate the computational cost involved in a pixel-wise classification approach. We address the issue here by using a set of auto-encoders, which learn an image-to-image mapping. Within our context, this means that the image can be processed in one step at a higher order of efficiency.

Conventional auto-encoders consist of feed-forward neural networks for which the input and output must be exactly the same. The network typically consists of two stages that learn the functions f and g , which are called encoder and decoder functions, respectively. Formally speaking, given an input x , the network must minimize a divergence $L(x, g(f(x)))$. An auto-encoder might initially appear to be pointless because it is trained to learn the identity function. Nevertheless, the encoder function f is typically forced to produce a representation with a lower dimensionality than the input. The encoder function therefore provides a meaningful compact representation of the input, which might be of great interest for feature learning or dimensionality reduction [37].

In our case, we modify this traditional behavior so that the model specializes in selecting the pixels that belong to each of the elements from the category set. This type of model is referred to as Selectional Auto-Encoder (SAE) [13]. An SAE is trained to perform a function such that $s : \mathbb{R}^{(w \times h)} \rightarrow [0, 1]^{(w \times h)}$. In other words, it learns a

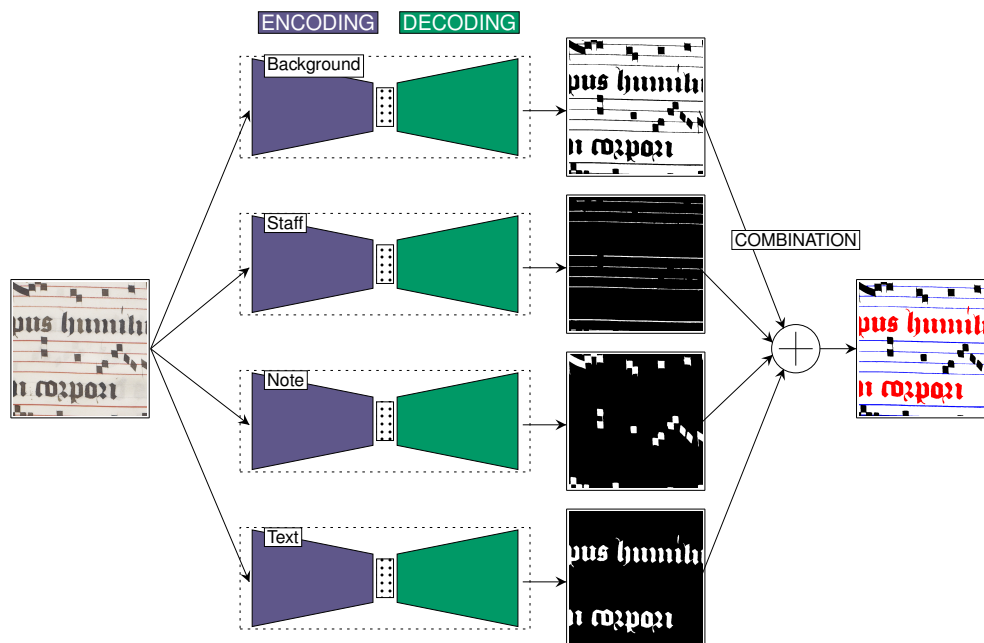


Figure 1. Graphical scheme of the SAE-based *1-vs-all* approach for document analysis of music scores images. The outputs of the individual SAE are represented as grayscale masks in which the white color represents the maximum selectional value. Coloring for the final combination: background in white, music symbols in black, staff lines in blue, and text in red.

binary map over a $w \times h$ image that preserves the input shape. The predicted value for each pixel indicates its selection level, representing 1 as the maximum. Then, the network is trained to minimize the divergence between a binary image in which only the pixels that belong to the category of interest are activated.

Actually, an SAE represents a two-class categorizer with one class represented by the value 0 and another represented by the value 1. To perform a multi-class document analysis like the one formalized above, we follow a *1-vs-all* strategy, much in the same way as other binary classifiers such as the Support Vector Machine [20]. That is, we train a different SAE focused on each category, assuming the category of interest as 1 and the remaining ones as 0. At the time of inference, the outputs of all the trained SAEs are combined to obtain a global analysis of the document.

We find two important advantages of predicting each layer separately. On the one hand, the extraction of a specific layer only requires the ground-truth data of the targeted category, thus reducing the effort involved in preparing the training set if only a subset of the categories is pursued. On the other hand, the predictions provided by each SAE could be processed separately—e.g., to apply different thresholds to each result or to resolve inconsistencies when many predictions disagree about a specific region—which might be interesting depending on the way the subsequent stages of the OMR workflow operate.

Below we discuss more details about the actual implementation of the described framework for the present work.

3.1 Implementation details

An SAE can be configured in many ways. We specifically consider a Fully-Convolutional Network (FCN) topology, given the good results obtained by this type of neural networks in this task [32], and in general for any image-related task [23].

An FCN is a type of neural network that is entirely based on filters (i.e., convolutions). These filters are configured in a hierarchy of layers that provide multiple representations from the input image with different levels of abstraction: while the first layers emphasize details of the image, the last layers focus on high-level entities [22]. The parameters of the convolutions are typically optimized by backpropagation [24] through a training set, with the objective of generalizing to unseen data.

Consequently, the hierarchy of layers of our SAE consists of a series of convolutional plus pooling layers, until an intermediate layer is attained. As these layers are applied, filters are able to relate parts of the image that were initially far apart. Then, it follows a series of convolutional plus up-sampling layers that reconstruct the image up to the same input size copying neighboring pixels. The last layer consists of a set of neurons with *sigmoid* activation that predict a value in the range of $[0, 1]$, depending on the *selectional* level predicted for the corresponding input pixel. This selectional level is expected to approach 1 as the model is more confident that the pixel belongs to the category of interest. This specific configuration needs to be tweaked for the problem at issue, and so we will perform some preliminary experiments to evaluate different options.

The training stage consists of providing the SAE with

Corpus	Salzinnes	Einsiedeln
Pages	10	10
Avg. height and width per page (in pixels)	5 100 × 3 200	5 550 × 3 650
Background	80.6	79.1
% Note	11.2	10.0
Staff line	4.5	6.9
Text	3.7	4.0

Table 1. Overview of the corpus used in our experiments: number of pages, average size per page, and class distribution (in %).

examples of images and their corresponding ground truth, that is, binary maps over the pixels that belong to the category of interest. The *cross-entropy* loss function between each output activation and its expected activation is computed. Then, filters are tuned using stochastic gradient descent optimization [2] with a mini-batch size of 16 and the adaptive learning rate strategy proposed by Zeiler [38].

Once all the corresponding SAEs for the categories considered in this work ($SAE_{background}$, SAE_{note} , SAE_{staff} , SAE_{text}) are trained, they can be used to perform the document analysis process. In order to compute a single category for each pixel, we select the category whose SAE retrieves the highest selection value. A graphical scheme of this operation is depicted in Figure 1.

Given that our SAE is configured as a fully-convolutional model (i.e., without any dense layer), the input and the output layers can be of an arbitrary size. In practice, however, processing a high-resolution musical score has a high memory consumption. This is why in our case we need to divide the input music score into equal patches of 256×256 pixels, which was the largest size feasible with our computational resources. Theoretically, this limitation should not affect the performance of the models except for the case of the edges of the input patches. This can be palliated by considering overlap at the time of splitting the input image, and ignoring the edges of the predictions made.

4. EXPERIMENTS

4.1 Experimental setup

For the evaluation of our approach, we consider high-resolution image scans of two ancient music manuscripts. The first corpus is a subset of 10 pages of the Salzinnes Antiphonal manuscript (CDM-Hsmu M2149.14),¹ music score dated 1554–5. The second corpus is 10 pages of the Einsiedeln, Stiftsbibliothek, Codex 611(89), from 1314.² Table 1 gives an overview of this corpora with some of their specific features. For our experiments, the images have been considered in their grayscale format.

¹ <https://cantus.simssa.ca/manuscript/133/>

² <http://www.e-codices.unifr.ch/en/sbe/0611/>

The ground-truth data was created manually by labeling pixels into the four categories mentioned above. Although in this work we circumscribe the experiments to corpora from Medieval music manuscripts, we believe that their difficulty and wealth of information (at the image level) allows us to generalize the conclusions to any type of music score image.

In order to provide a more reliable assessment, we follow a corpus-wise 5-fold cross validation scheme. In each iteration of each corpus, 2 complete pages—not necessarily consecutive ones—are used for test evaluation, 2 pages are used as validation, and 6 pages for training the SAE models. The reported results will represent averages over these 5 independent evaluation processes. It should be noted that the experiments in both corpora have been performed individually, since in the context of machine learning, it could be assumed that the samples belong to the same domain. Despite this assumption, future research aims to expand the experimental setup to include more realistic scenario with cross-manuscript experiments.

As can be observed, the distribution of each class is highly biased, background being the most represented class. Given this distribution, we consider appropriate metrics for such imbalanced datasets. For instance, the F_1 typically represents a fair metric in these scenarios. In a two-class classification problem, this measure can be computed as

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}, \quad (1)$$

where True Positive (TP) stands for the correctly classified elements of the relevant class, False Positive (FP) represents the misclassified elements from the relevant class, and False Negative (FN) stands for the misclassified elements of non-relevant class.

To compute single values encompassing all possible categories, this metric can be reformulated into macro F_1 [27], which is computed as the average of all class-wise metrics.

4.2 Network selection

In this section we carry out a preliminary study to evaluate how some of the parameters of the SAE configuration affect the accuracy of the classification. It is worth mentioning that the different configurations may behave differently according to the category of interest (background, text, note, or staff). In this regard, however, we assume for this study a general assessment taking into account all classes simultaneously.

There exist a huge number of possibilities for establishing the organization of the neural model [18]. In order to reduce the search, we restrict ourselves to evaluate only the most interesting hyper-parameterization, namely the depth of the encoding/decoding blocks and whether encoding and decoding layer actually perform down-sampling and up-sampling operators. The latter points to an interesting issue: performing down- and up-sampling operations allows intermediate filters to focus on different levels

Depth	Down/Up-Sampling	
	No	Yes
1	90.0	90.7
2	91.5	94.9
3	93.3	96.0
4	94.2	95.4

Table 2. Macro average F_1 (%) of the 5-fold cross-validation over the validation partitions, with respect to the depth of the encoding/decoding layers and whether or not considering sampling operators.

of abstraction within the image, also reducing the intrinsic complexity—since the image in the intermediate layers would be smaller. However, keeping the original size throughout the process avoids having to learn to reconstruct the image, at the cost of losing the benefits discussed above for the opposite case.

The rest of the parameters are fixed manually, based on informal testing, as follows: the number of filters per convolution are set to 128 and the size of the convolutional kernels to 5×5 . Also, all intermediate convolutional filters use Rectified Linear Unit (ReLU) activations [17].

Table 2 shows the macro average F_1 attained by each different SAE configuration on the validation sets.

Concerning the depth of the encoding/decoding blocks, a progress towards an upward trend is observed. In the case of using sampling operations, this trend finds a peak at 3 layers. In the opposite case (i.e., with no sampling), the improvements are more subtle and the peak is not reached within the number of layers considered. Due to computational resources, we were not able to carry out experiments with more layers, so it is not possible to know when the peak would be reached.

On the other hand, regardless of the number of layers chosen, we can observe that there is a clear tendency in the advantage of doing down- and up-sampling operations, since the latter case is always better than its analog for the same depth in the experiments carried out.

According to these results, the final SAE configuration for all the categories is shown in Table 3.

4.3 Results

In this section we analyze in detail the performance that was attained using the best SAE configuration of the previous section in comparison to the pixelwise CNN-based approach, that currently represents the state of the art in this task [5]. All experiments have been performed in similar conditions on a general-purpose computer with the following technical specifications: Intel(R) Core(TM) i7-7700HQ CPU @2.8GHz \times 4, 32GB RAM, GTX1070 GPU and Linux Mint 18.2 (64 bits) operating system. The code has been written using Python language (v2.7) and Keras framework.

Given that the objective of this paper is not only to measure the accuracy of the new model but also its efficiency,

Table 5 shows a comparison of both aspects in terms of macro F_1 and the approximated time needed to process a document. Traditionally, the training cost is not taken into account when evaluating these systems because the process is usually performed offline. Note, however, that both approaches involved a similar training cost in the order of several hours on Graphical Processing Units.

Accuracy results show a visible difference between the corpora considered. While results are closer to the optimum in Salzinnes, both approaches seem to find more difficulties in Einsiedeln. However, this difference is not obvious in a qualitative evaluation, as depicted in Table 4.

It can be observed that the SAE-based strategy generally obtains a higher F_1 than that based on CNNs. Note, however, that the objective of this experiment is not to demonstrate that the SAE-based approach outperforms significantly the state of the art, but to obtain results that can be considered similar, which is clearly reported according to these figures. On the other hand, the computation time needed to process a complete manuscript page is drastically lower with the SAE, going from several hours to a few minutes. This happens because the CNN approach has to classify each pixel of the image, whereas the SAE approach can make predictions of many pixels simultaneously (in our experiments, 256×256). Obviously, the network of the latter approach is more complex, but it clearly compensates with respect to the temporal cost.

Thus, this comparison with the state of the art demonstrates that the proposed approach allows obtaining a similar performance when performing the document analysis, with a radically lower computational cost, thus making an important contribution to the field of OMR.

5. CONCLUSIONS

In this paper we have presented a machine-learning strategy for the document analysis of music score images. The strategy consists in training SAE, configured as convolutional neural networks, that allow to extract the different layers of information found in documents through an image-to-image formulation.

In a preliminary study, we have determined some of the parameters that lead to a better configuration of the SAE. In particular, we have evaluated the depth of the encoder/decoder layers, as well as the relevance of whether performing or not down- and up-sampling operations. Generally, increasing the number of layers is beneficial, to a certain extent, while sampling operators lead to a much more effective network.

Although we did not exhaustively test the various possible network configurations for this first study, we have shown that the proposed approach can achieve the accuracy similar to the state-of-the-art algorithms, and more importantly, with an efficiency improvement of orders of magnitude.

Our results represent the first step towards an interactive scenario in which the user and the system can interact to solve the OMR task. This scenario has already been devised before [34]; however, our approach allows us to be

Input	Encoding	Decoding	Output
$[0, 255]^{256 \times 256}$	Conv(128,5,5,ReLU)	Conv(128,5,5,ReLU)	$[0, 1]^{256 \times 256}$
	MaxPool(2,2)	UpSamp(2,2)	
	Conv(128,5,5,ReLU)	Conv(128,5,5,ReLU)	
	MaxPool(2,2)	UpSamp(2,2)	
	Conv(128,5,5,ReLU)	Conv(128,5,5,ReLU)	
	MaxPool(2,2)	UpSamp(2,2)	
		Conv(1,5,5,Sigmoid)	

Table 3. Detailed description of the selected SAE architecture, implemented as a FCN. Conv(f, h, w, a) stands for a convolution operator of f filters, with $h \times w$ pixel kernels with an a activation function; MaxPool(h, w) stands for the max-pooling operator with a $w \times h$ kernel and stride; UpSamp(h, w) denotes an up-sampling operator of h rows and w columns; *ReLU* and *Sigmoid* denote Rectifier Linear Unit and Sigmoid activations, respectively.



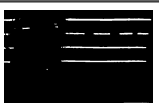



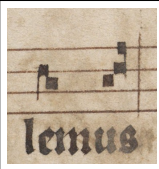


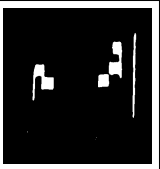

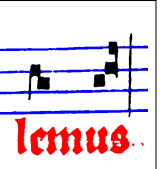
Original	Prediction				Result
	Background	Staff	Note	Text	
					
					

Table 4. Qualitative examples of document analysis over selected patches of the corpora (Salzinnes, first row; Einsiedeln, second row), depicting the original piece of the document along with the individual SAE predictions, and the resulting analysis. The predictions of the individual SAE are represented as grayscale masks in which the white color represents the maximum selectional value. Coloring for the final result: background in white, music symbols in black, staff lines in blue, and text in red.

Strategy	Macro F_1		Time per page
	Salzinnes	Einsiedeln	
SAE	95.5	90.3	~ 1 minute
CNN	91.3	88.4	~ 6 hours

Table 5. Comparison of our SAE-based approach with the state-of-the-art (CNN) performance taking into account both accuracy and efficiency of the document analysis process.

closer to real practice since the document analysis processing stage no longer implies a bottleneck.

Nevertheless, the costly training process is still an obstacle for this scenario in which models must re-trained according to user's corrections. Therefore, addressing this matter is essential in future work. Among the possible options, we want to consider the use of pre-trained models that can be adapted with few new samples and less demanding training procedures.

Also, we are especially interested in the aspect of cross-manuscript adaptation. That is, how to exploit models

specifically trained for a manuscript in other manuscripts with a different layout organization. In this way, the initial effort to obtain ground-truth data from the manuscript at issue can be reduced. We believe that semi-supervised learning algorithms could be of interest in this case, for which the models learn to adapt to a new manuscript by just providing them with new (unlabeled) images. This can be performed by promoting convolutional filters that are both useful for the classification task and invariant with respect to the differences among manuscript types [14].

6. ACKNOWLEDGEMENT

This work was supported by the Spanish Ministerio de Economía, Industria y Competitividad through HispaMus project (TIN2017-86576-R) and Juan de la Cierva - Formación grant (Ref. FJCI-2016-27873), and the Social Sciences and Humanities Research Council of Canada.

7. REFERENCES

- [1] D. Bainbridge and T. Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35(2):95–121, 2001.
- [2] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.
- [3] J. A. Burgoyne and I. Fujinaga. Lyric extraction and recognition on digital images of early music sources. In *Proceedings of the 10th International Society for Music Information Retrieval Conference*, pages 723–728, 2009.
- [4] J. A. Burgoyne, L. Pugin, G. Eustace, and I. Fujinaga. A comparative survey of image binarisation algorithms for optical recognition on degraded musical sources. In *Proceedings of the 8th International Conference on Music Information Retrieval*, pages 509–512, 2007.
- [5] J. Calvo-Zaragoza, F. J. Castellanos, G. Vigliensoni, and I. Fujinaga. Deep neural networks for document processing of music score images. *Applied Sciences*, 8(5):654–674, 2018.
- [6] J. Calvo-Zaragoza, G. Vigliensoni, and I. Fujinaga. One-step detection of background, staff lines, and symbols in medieval music manuscripts with convolutional neural networks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, Suzhou, China*, pages 724–730, 2017.
- [7] V. B. Campos, J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal. Sheet music statistical layout analysis. In *15th International Conference on Frontiers in Handwriting Recognition, Shenzhen, China*, pages 313–318, 2016.
- [8] L. Chen and C. Raphael. Human-directed optical music recognition. *Electronic Imaging*, 2016(17):1–9, 2016.
- [9] L. Chen, E. Stolterman, and C. Raphael. Human-interactive optical music recognition. In *ISMIR*, pages 647–653, 2016.
- [10] C. Dalitz, M. Droettboom, B. Pranzas, and I. Fujinaga. A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):753–766, 2008.
- [11] J. Dos Santos Cardoso, A. Capela, A. Rebelo, C. Guedes, and J. Pinto da Costa. Staff detection with stable paths. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1134–1139, 2009.
- [12] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, NY, 2nd edition, 2001.
- [13] A. Gallego and J. Calvo-Zaragoza. Staff-line removal with selectional auto-encoders. *Expert Systems with Applications*, 89:138–48, 2017.
- [14] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.
- [15] B. Gatos, I. Pratikakis, and S. J. Perantonis. Adaptive degraded document image binarization. *Pattern Recognition*, 39(3):317–327, 2006.
- [16] T. Géraud. A morphological method for music score staff removal. In *International Conference on Image Processing*, pages 2599–2603, 2014.
- [17] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL*, pages 315–323, 2011.
- [18] D. Graupe. *Principles of Artificial Neural Networks*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2nd edition, 2007.
- [19] N. R. Howe. Document binarization with automatic parameter tuning. *International Journal on Document Analysis and Recognition*, 16(3):247–258, 2013.
- [20] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.
- [21] F. D. Julca-Aguilar and N. S. T. Hirata. Image operator learning coupled with CNN classification and its application to staff line removal. In *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan*, pages 53–58, 2017.
- [22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *26th Annual Conference on Neural Information Processing Systems*, pages 1106–1114, 2012.
- [23] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [25] B. Moysset, C. Kermorvant, C. Wolf, and J. Louradour. Paragraph text segmentation into lines with recurrent neural networks. In *13th International Conference on Document Analysis and Recognition*, pages 456–460. IEEE, 2015.
- [26] Y. Ouyang, J. A. Burgoyne, L. Pugin, and I. Fujinaga. A robust border detection algorithm with application to medieval music manuscripts. In *Proceedings of the 2009 International Computer Music Conference*, pages 101–104, 2009.
- [27] A. Özgür, L. Özgür, and T. Güngör. Text categorization with class-based and corpus-based keyword selection. In *International Symposium on Computer and Information Sciences*, pages 606–615, 2005.

- [28] T. Pinto, A. Rebelo, G. A. Giraldi, and J. S. Cardoso. Music score binarization based on domain knowledge. In *5th Iberian Conference on Pattern Recognition and Image Analysis, Las Palmas de Gran Canaria, Spain*, pages 700–708, 2011.
- [29] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. S. Marçal, C. Guedes, and J. S. Cardoso. Optical music recognition: State-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190, 2012.
- [30] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.
- [31] C. Segura, I. Barbancho, L. J. Tardón, and A. M. Barbancho. Automatic search and delimitation of frontispieces in ancient scores. In *18th European Signal Processing Conference*, pages 254–258, 2010.
- [32] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, 2017.
- [33] G. Vigliensoni, G. Burlet, and I. Fujinaga. Optical measure recognition in common music notation. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, pages 125–130, 2013.
- [34] G. Vigliensoni, J. Calvo-Zaragoza, and I. Fujinaga. An environment for machine pedagogy: Learning how to teach computers to read music. In *Proceedings of IUI Workshop on Music Interfaces for Listening and Creation, Tokyo, Japan*, pages 1–4, 2018.
- [35] Q. N. Vo, S. H. Kim, H. J. Yang, and G. Lee. An MRF model for binarization of music scores with complex background. *Pattern Recognition Letters*, 69(Supplement C):88–95, 2016.
- [36] Q. N. Vo, S. H. Kim, H. J. Yang, and G. Lee. Binarization of degraded document images based on hierarchical deep supervised network. *Pattern Recognition*, 74:568–586, 2018.
- [37] W. Wang, Y. Huang, Y. Wang, and L. Wang. Generalized autoencoder: A neural network framework for dimensionality reduction. In *Workshops of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 490–497, June 2014.
- [38] M. D. Zeiler. ADADELTA: An adaptive learning rate method. *Computer Research Repository*, abs/1212.5701, 2012.

GENRE-AGNOSTIC KEY CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS

Filip Korzeniowski and Gerhard Widmer

Institute of Computational Perception,
Johannes Kepler University, Linz, Austria
filip.korzeniowski@jku.at

ABSTRACT

We propose modifications to the model structure and training procedure to a recently introduced Convolutional Neural Network for musical key classification. These modifications enable the network to learn a genre-independent model that performs better than models trained for specific music styles, which has not been the case in existing work. We analyse this generalisation capability on three datasets comprising distinct genres. We then evaluate the model on a number of unseen data sets, and show its superior performance compared to the state of the art. Finally, we investigate the model’s performance on short excerpts of audio. From these experiments, we conclude that models need to consider the harmonic coherence of the whole piece when classifying the local key of short segments of audio.

1. INTRODUCTION

The musical key is the highest-level harmonic representation in Western tonal music. It thus plays a central role in understanding the semantic content of a piece. Such understanding drives not only theoretical analyses of music, but is also relevant for modern music creators, who mix samples from various different pieces that fit well harmonically into a new composition. However, deriving the key of a musical piece is a demanding task that only experts can perform. It is thus impractical to annotate large music collections by hand. Therefore, we need computational key classification systems.

Most key classification systems (e.g. [9, 16, 17, 21]) conform to the same principle: they extract a time-frequency representation of the audio, filter out nuisances, map this representation to chroma vectors, and accumulate them over time. The resulting feature vector is then compared to template vectors for each key. The drawbacks of such approaches include that key templates differ for different musical genres [9] and favour one key mode over another [1]. This leads to key classification systems that perform well only on the musical styles they were designed for. Although

there are attempts to address these issues [2], ideally, we would want a model that handles different kinds of input autonomously, and does not need human intervention to e.g. balance mode probabilities.

Data-driven methods bear the potential to meet this requirement. Recently, an end-to-end neural-network-based key classification model was introduced [14]. Although it generalised better across musical genres than hand-crafted approaches, it still achieved the best results when tuned specifically for a musical style. In this paper, we present modifications to the model structure and its training procedure that enable the model to learn a key classifier that is agnostic to genre. Not only does it perform better than the model proposed in [14] on all genres the latter is optimised for; it does so not despite, but *because* it is trained on various musical styles, instead of a specific one (see Sec. 3.4).

2. METHOD

We build upon the same audio processing pipeline used in [14], and input to the network a log-magnitude log-frequency spectrogram (5 frames per second, frame size 8 192, sample rate 44 100 Hz, 24 bins per octave). We limit the frequency range to the harmonically most relevant 65 Hz to 2 100 Hz, as found in [12].

The network structure proposed in [14] was modelled after typical processing pipelines used for key classification. It features five convolutional layers of 5×5 kernels for spectrogram processing, followed by a dense projection into a frame-wise embedding space, which is then averaged over time and classified using a softmax layer. All layers except the last use the exponential-linear activation function [4] (ELU). The architecture, which we name *KeyNet*, is summarised in Table 1a.

During training, the model is shown the complete spectrogram of a piece. Its weights are then adapted using stochastic gradient descent to minimise the categorical cross-entropy between the predicted key distribution and the ground truth. We will refer to the KeyNet architecture, when trained using full spectrograms, as *KeyNet/F*.

2.1 Adaptations of the Training Procedure

The outlined training scheme has two drawbacks. First, the computation of a single update is expensive; the network has to process the full spectrogram (e.g. 600×105 values for



© Filip Korzeniowski and Gerhard Widmer. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: Filip Korzeniowski and Gerhard Widmer. “Genre-Agnostic Key Classification With Convolutional Neural Networks”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

a two-minute piece), and keep intermediate results for back-propagating the error. Training is thus slow and requires much memory. Second, it keeps the variety of the data lower than necessary, as the network sees the same spectrograms at every epoch.

To circumvent these drawbacks, we show the network only short snippets instead of the whole piece at training time (similar to random cropping in computer vision). These snippets should be as short as possible to reduce computation time, but have to be long enough to contain the relevant information to determine the key of a piece. From our datasets, we found 20 s to be sufficient (with the exception of classical music, which we need to treat differently, due to the possibility of extended periods of modulation—see Sec. 3.1 below). Each time the network is presented a song, we cut a random 20 s snippet from the spectrogram. The network thus sees a different variation of each song every epoch.

During testing, the network processes the whole piece. This gives better results than when using only a snippet. Since we do not have to store intermediate results and process each piece many times as in training, memory space and run time are not an issue. We will refer to KeyNet models trained using spectrogram snippets as *KeyNet/S*.

We expect this modification to have the following effects. a) Back-propagation will be faster and require less memory, because the network sees shorter snippets; we can thus train faster, and process larger models. b) The network will be less prone to over-fitting, since it almost never sees the same training input; we expect the model to generalise better. c) The network will be forced to find evidence for a key in each excerpt of the training pieces, instead of relying on parts where the key is more obvious; by asking more of the model, we expect it to pick up more subtle relationships between the audio and its key.

2.2 Adaptations of the Model Structure

The KeyNet architecture uses a dense layer to project the processed spectrogram into a key embedding space. In its original formulation, which uses an embedding space with 48 dimensions and 8 feature maps in the convolutional layers, this projection accounts for 65 % of the network’s parameters. Dense layers are also more prone to over-fitting than convolutional layers.

We thus propose to use a network architecture that does away with dense layers, and relies on convolutions and pooling only. At the same time, we move away from modelling the network based on traditional key classification methods—recall that the components of KeyNet were designed to correspond to components in typical key classification pipelines—and instead use a general network architecture for classification, based on the all-convolutional net [19]. The new architecture is summarised in Table 1b, and will be referred to as *AllConv*. As with KeyNet/S, we will train this architecture only with the snippet method.

We expect this change to improve results and generalisation because a) convolutional layers over-fit less than dense layers; b) given the same number of parameters, deeper

(a) KeyNet Architecture			(b) AllConv Architecture		
Layer Type	FMaps	Params	Layer Type	FMaps	Params
Input			Input		
Conv-ELU	N_f	5×5	Conv-ELU	N_f	5×5
Conv-ELU	N_f	5×5	Conv-ELU	N_f	3×3
Conv-ELU	N_f	5×5	Pool-Max		2×2
Conv-ELU	N_f	5×5	Conv-ELU	$2N_f$	3×3
Conv-ELU	N_f	5×5	Conv-ELU	$2N_f$	3×3
Conv-ELU	N_f	5×5	Pool-Max		2×2
Dense-ELU		$2 \cdot N_f$	Conv-ELU	$4N_f$	3×3
Pool-Time Avg.			Conv-ELU	$4N_f$	3×3
Dense-Softmax		24	Pool-Max		2×2
			Conv-ELU	$8N_f$	3×3
			Conv-ELU	$8N_f$	3×3
			Conv-ELU	24	1×1
			Pool-Global Avg.		
			Softmax		

Table 1. Neural Network architectures. N_f is a parameter that controls the model complexity. Horizontal lines denote dropout layers [20]. Here, dropout is applied on complete feature maps, not individual units. Each convolution is followed by batch normalisation [11]. *FMaps* indicates the number of feature maps, while *Params* the parameters of the layer (kernel size, pool size, or number of units).

networks are more expressive than shallower ones [8, 15]; c) comparable architectures have shown to perform well in other audio-related tasks [7, 13].

3. EXPERIMENTS

We first evaluate how the proposed modifications affect the key classification performance in Sec. 3.3. Then, we analyse how the number and genre of training data influence results in Sec. 3.4.

3.1 Data

Since we are interested in how well the models generalise across different genres, we use datasets that encompass three distinct musical styles. As in [14], we apply pitch shifting in the range of -4 to +7 semitones to increase the amount of training data.

Electronic Dance Music: Here, we use songs from the GiantSteps MTG Key dataset¹, collected by Ángel Faraldo. It comprises 1486 distinct two-minute audio previews from www.beatport.com, with key ground truth for each excerpt. We only use excerpts labelled with a single key and a high confidence (1077 pieces), and split them into 80 % training and 20 % validation. For testing, we use the GiantSteps Key Dataset². It comprises 604 two-minute audio previews from the same source (but distinct from the training set).

¹ <https://github.com/GiantSteps/giantsteps-mtg-key-dataset>

² <https://github.com/GiantSteps/giantsteps-key-dataset>

Pop/Rock Music: For this genre, we use the McGill Billboard dataset [3]³. It consists of 742 unique songs sampled from the American Billboard charts between 1958 and 1991. We split these songs into subsets of 62.5% for training, 12.5% for validation, and 25% for testing. We determine the global key for each song using the procedure described in [14], which leaves us with 625 songs with key annotations in total. The exact division and key ground truths are available online⁴.

Classical Music: To cover this genre, we collected 1504 (mostly piano) pieces from our internal database for which we could derive the key from the piece’s title. Classical pieces often modulate their key, but usually start in the key denoted in the title. We thus only use the first 30 s of each recording. Tracking key modulations is left for future work. We then select 81 % for training, 9 % for validation, and 10 % for testing.

3.2 Metrics

We adopt the standard evaluation score for Key Classification as defined in the MIREX evaluation campaign⁵. It goes beyond simple accuracy, as it considers harmonic similarities between key classes. A prediction can fall into one of the following categories:

Correct: if the tonic and the mode (major/minor) of prediction and target correspond.

Fifth: if the tonic of the prediction is the fifth of the target (or vice versa), and modes correspond.

Relative Minor/Major: if modes differ and either a) the predicted mode is minor and the predicted tonic is 3 semitones below the target, or b) the predicted mode is major and the predicted tonic is 3 semitones above the target.

Parallel Minor/Major: if modes differ but the predicted tonic matches the target.

Other: Prediction errors not caught by any category, i.e. the most severe errors.

Then, a weighted score can be computed as $w = r_c + 0.5 \cdot r_f + 0.3 \cdot r_r + 0.2 \cdot r_p$, where r_c , r_f , r_r , and r_p are the ratios of the correct, fifth, relative minor/major, and parallel minor/major, respectively. We will use this weighted score for our comparisons.

3.3 Evaluation of the Adaptations

To evaluate the effect of our proposed adaptations, we train the three setups (KeyNet/F, KeyNet/S, AllConv) with the combined data of all datasets. We will consider *validation* results in the first sets of experiments, and show results on

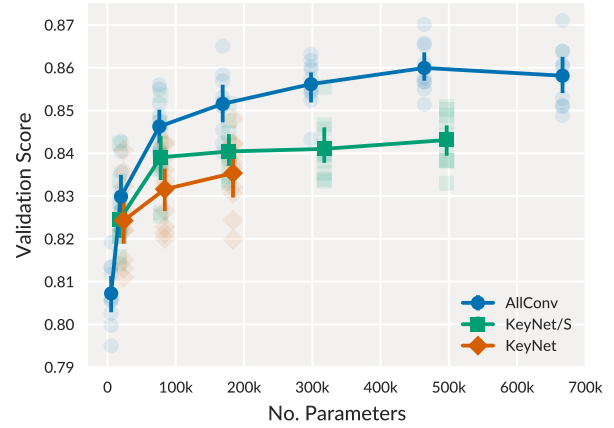


Figure 1. Average validation score over 10 runs for the different model setups. Whiskers represent 95 % confidence intervals computed by bootstrapping. Transparent dots show results of the individual runs. We see that given similar network sizes, the AllConv model performs best. Also, using snippet training (KeyNet/S) improves results compared to full spectrogram training (KeyNet/F), and enables training larger networks.

the testing sets only for our analyses and final evaluations. This way, we ensure that the final results are unbiased.

The capacity of a neural network depends not only on the architecture, but also on its size. For a fair comparison, we evaluate each architecture with varying network sizes. For the AllConv architecture, we select the number of feature maps $N_f \in \{2, 4, 8, 12, 16, 20, 24\}$. For the KeyNet architecture, the network size depends on the number of feature maps in the convolutional layers and the size of the embedding space. For practical reasons, we set the size of the embedding space to be $2N_f$, and select $N_f \in \{8, 16, 24, 32, 40\}$. Note that if we train on full spectrograms (KeyNet/F), we could not train networks with $N_f > 24$ due to memory constraints. For each model, we tried dropout probabilities of $p \in \{0.0, 0.1, 0.2\}$.

Figure 1 presents the results of the three model configurations. For each model and model capacity, we select the best dropout probability based on the validation results. The experiments show that both adaptations are beneficial. Training with snippets instead of full spectrograms gives better results at smaller network capacities and enables training of larger networks. The AllConv architecture achieves even better results, regardless of its size.

We can quantify two reasons for this, which are consequences of the expected benefits of the adaptations: *better generalization* through increased data variety and the absence of dense layers, and *better expressivity* through deeper architectures and by training the network on a more difficult task. For the first, better generalisation, we compare the average ratio of validation accuracy to training accuracy for each of the models (higher indicates less over-fitting): 0.945, 0.969, and 0.982 for KeyNet/F, KeyNet/S, and AllConv, respectively. For the second, *model expressiveness*, we compare the model’s capability to fit the training data in terms of accuracy: 0.837, 0.858, and 0.907 for KeyNet/F,

³ <http://ddmal.music.mcgill.ca/research/billboard>

⁴ <http://www.cp.jku.at/people/korzeniowski/bb.zip>

⁵ <http://www.music-ir.org/mirex>

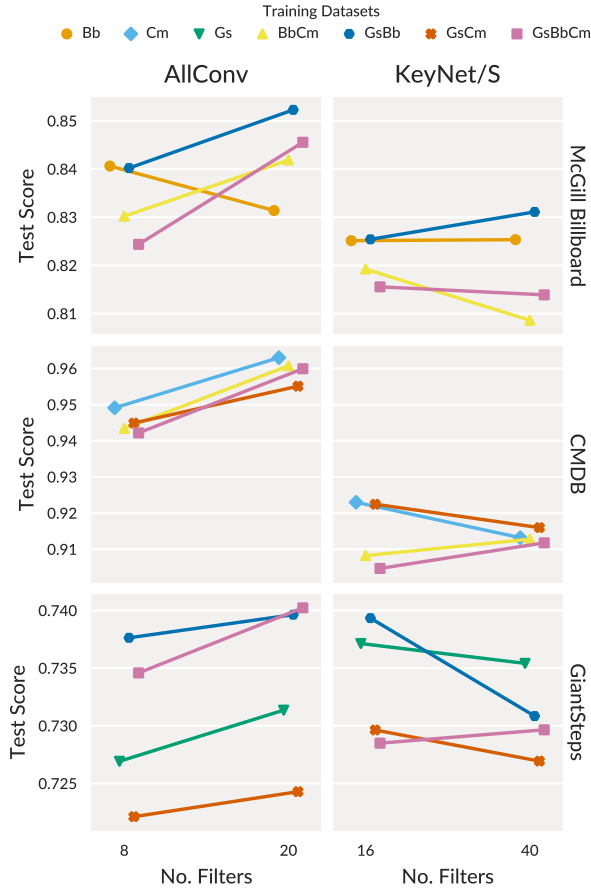


Figure 2. Average test scores over 10 runs for each architecture (columns), split by dataset (rows). The smaller models are on the left of each column. Colors indicate the training data used: *Bb* stands for the Billboard dataset, *Cm* for the classical music dataset, and *Gs* for the GiantSteps dataset. Each row shows the results of runs where the training set also contained the training data of the respective test set genre (e.g. in the first row, we only see runs where McGill Billboard data was included in training).

KeyNet/S, and AllConv, respectively. Stronger models that generalise better achieve better results.

3.4 Influence of Training Data

We then want to see how the number and genre of the datasets used for training affects results. To this end, we select the hyper-parameter settings for AllConv and KeyNet/S that achieved the best average results in the previous experiment: $N_f = 20, p = 0.1$ for AllConv, $N_f = 40, p = 0.1$ for KeyNet/S. Additionally, we consider smaller models of each type, i.e. $N_f = 8$ for AllConv and $N_f = 16$ for KeyNet/S, both without dropout. Under these settings, both architectures have a comparable number of parameters. We train these models using all possible 1, 2, and 3-combinations of the datasets, and evaluate them on all data. The results are shown in Fig. 2.

The main observations are: a) increasing model capacity is more beneficial to the AllConv model than KeyNet/S, re-

gardless of dataset; b) adding capacity to the AllConv model enables it to better deal with diverse data—the biggest gains of additional parameters are achieved if the model is trained on a combined dataset (pink line)—while this is not always the case for KeyNet/S (see the Billboard results, where it seems that adding classical music to the training set impairs the performance of this model); c) given enough capacity in the AllConv model, training using the complete data performs better than (or almost equal to) fitting a specific genre, while the opposite is the case for KeyNet/S, where specialised models outperform the general ones. We thus argue that the AllConv model not only copes better with diverse training data, but that it leverages the diversity in the training data to perform as well as it does.

4. EVALUATION

Motivated by the results above, the remainder of our analysis focuses on the AllConv model. To thoroughly investigate its performance and compare it to the state of the art, we evaluate it on the following unseen datasets:

KeyFinder⁶: 1 000 songs from a variety of popular music genres. Unfortunately, we have only the audio for 998 of the songs available.

Isophonics⁷: 180 songs by The Beatles, 19 songs by Queen, and 18 songs by Zweieck. Since these songs contain key modulations, we split them into single key segments and retain only segments annotated as major or minor keys, as was done for the 2017 MIREX evaluation campaign⁸.

Robbie Williams [6]: 65 songs by Robbie Williams, which we also split into single key segments as outlined above.

Rock⁹ [5]: 200 songs taken from Rolling Stone’s “500 Greatest Songs of All Time” list. As with the McGill Billboard dataset, only the tonics are annotated. We first split the songs according to the annotated tonics, and then follow a similar procedure as described in [14]: if more than 80 % of the tonic chords are in either major or minor, the mode is set accordingly; if there are no tonic chords in a segment, we consider dominant chords in the same way.

We select the best AllConv model based on the validation score over the compound data of Electronic, Pop/Rock and Classical music. On average, models with $N_f = 20$ and dropout probability of 0.1 performed best. However, the best single model used $N_f = 24$ (see Fig. 1), and was consequently chosen as final model.

In Table 2, we compare this model to other models proposed in the academic literature. For each dataset, we show the results of the best competing system, if available.

⁶ <http://www.ibrahimshaath.co.uk/keyfinder/>

⁷ <http://isophonics.net/datasets>

⁸ http://www.music-ir.org/mirex/wiki/2017:Audio_Key_Detection_Results

⁹ <http://rockcorpus.midside.com/>

Dataset	Model	Weighted	Correct	Fifth	Relative	Parallel	Other
GiantSteps	AllConv	74.6	67.9	7.0	8.1	4.1	12.9
	CK1 [14]	74.3	67.9	6.8	7.1	4.3	13.9
Billboard	AllConv	85.1	79.9	5.6	4.2	6.2	4.2
	CK2 [14]	83.9	77.1	9.0	4.9	4.2	4.9
Classical	AllConv	96.6	95.2	1.4	1.4	1.4	0.7
	-	-	-	-	-	-	-
KeyFinder	AllConv	76.1	70.0	5.7	7.4	4.7	12.1
	bgate [10]	72.4	65.0	8.6	6.5	5.4	14.4
Isophonics	AllConv	82.5	76.3	7.6	5.4	3.7	7.1
	BD1 [2]	75.1	66.0	13.6	5.1	3.9	9.2
R. Williams	AllConv	81.2	72.4	10.8	10.3	1.3	5.2
	HS1 [18]	77.1	68.8	10.1	9.0	3.2	9.0
Rock	AllConv	74.3	69.3	6.5	1.7	6.0	16.5
	-	-	-	-	-	-	-

Table 2. Evaluation results. Best results are in boldface.

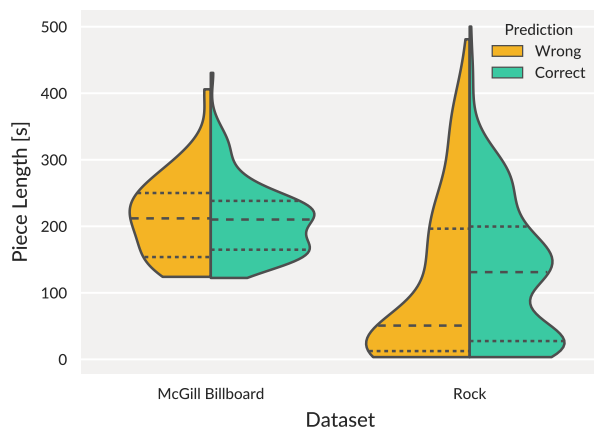


Figure 3. Distributions of the length of correctly and incorrectly classified excerpts depending on the dataset they come from. Densities are estimated using kernel density estimation. Horizontal lines with long dashes indicate the median, those with short dashes the quartiles. The densities are normalised, i.e. they do not indicate how many instances were classified correctly (or incorrectly), but only the distribution of excerpt lengths within each group.

For the GiantSteps and Billboard datasets, the best competing systems were variants of the neural-network-based model from [14]. For the pre-segmented Isophonics and Robbie Williams datasets, we use the results available on the MIREX 2017 website⁸. For the KeyFinder dataset, we report the best results achieved using the open-source reference implementation¹⁰ of the algorithms from [10].

As we can see, the proposed model *performs best for all datasets* for which comparisons were possible. Keep in mind that the systems we compare to are often specifi-

cally tuned for a genre (CK1, CK2, HS1, bgate) or set up to favour certain key modes prevalent in a dataset (BD1), while we use the same, general model for all datasets. For example, CK1 performs badly on the Billboard dataset ($w = 72.8$), BD1 on the GiantSteps ($w = 59.6$), and HS1 on the Isophonics dataset ($w = 64.1$). In this light, it is remarkable that the proposed model consistently out-performs the others.

However, the results also point us to a deficiency of the model. Recall that for some datasets (e.g. Rock), we split the files according to key annotations, and process each excerpt individually. If we compare the results on the Rock dataset with those on the Billboard dataset, we see a large discrepancy, although both sets comprise similar musical styles. As Fig. 3 demonstrates, the duration of a classified excerpt plays a major role here: for the Billboard set, the median length of excerpts classified correctly matches the one of incorrect classifications; for the Rock set, however, the median lengths differ greatly: 131 s vs. 51 s, for correctly and incorrectly classified excerpts, respectively. The distribution of excerpt lengths that are classified correctly is thus very different from the one of incorrectly classified excerpts in the Rock set. The shorter an excerpt, the more likely it is classified incorrectly.

This is not surprising per se. Determining the key of a piece requires a certain amount of musical context. However, it shows that in order to move beyond global key classification, and towards recognising key modulations, it will not suffice to detect key boundaries and apply known methods within these boundaries. To recognise key modulations, classifying short excerpts individually will reach a glass ceiling. Instead, we will need models that consider the hierarchical harmonic coherence of the whole piece.

¹⁰ <https://github.com/angelfaraldo/edmkey>

5. CONCLUSION

We have presented a genre-agnostic key classification model based on the system developed in [14], with improvements of the training procedure and network structure. These improvements enable faster training, better generalisation, and training larger and thus more powerful models, which can leverage diverse training data instead of being impaired by it. The resulting key classifier generalises well over datasets of different musical styles, and out-performs systems that are specialised for specific genres (see Table 2).

6. ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC) under the EU’s Horizon 2020 Framework Programme (ERC Grant Agreement number 670035, project “Con Espressione”).

7. REFERENCES

- [1] Joshua Albrecht and Daniel Shanahan. The Use of Large Corpora to Train a New Type of Key-Finding Algorithm: An Improved Treatment of the Minor Mode. *Music Perception: An Interdisciplinary Journal*, 31(1):59–67, 2013.
- [2] Gilberto Bernardes, Matthew E. P. Davies, and Carlos Guedes. Automatic musical key estimation with adaptive mode bias. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, March 2017.
- [3] John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis. In *12th International Society for Music Information Retrieval Conference (ISMIR)*, Miami, USA, October 2011.
- [4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *International Conference on Learning Representations (ICLR)*, *arXiv:1511.07289*, San Juan, Puerto Rico, February 2016.
- [5] Trevor de Clercq and David Temperley. A corpus analysis of rock harmony. *Popular Music*, 30(01):47–70, January 2011.
- [6] Bruno Di Giorgi, Massimiliano Zanoni, Augusto Sarti, and Stefano Tubaro. Automatic chord recognition based on the probabilistic modeling of diatonic modal harmony. In *Proceedings of the 8th International Workshop on Multidimensional Systems*, Erlangen, Germany, September 2013.
- [7] Hamid Eghbal-Zadeh, Bernhard Lehner, Matthias Dorfer, and Gerhard Widmer. CP-JKU Submissions for DCASE-2016: A Hybrid Approach Using Binaural I-Vectors and Deep Convolutional Neural Networks. In *Detection and Classification of Acoustic Scenes and Events (DCASE)*, September 2016.
- [8] Ronen Eldan and Ohad Shamir. The Power of Depth for Feedforward Neural Networks. In *Proceedings of Machine Learning Research (COLT 2016)*, New York, USA, June 2016.
- [9] Ángel Faraldo, Emilia Gómez, Sergi Jordà, and Perfecto Herrera. Key Estimation in Electronic Dance Music. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello, editors, *Advances in Information Retrieval*, volume 9626, pages 335–347. Springer International Publishing, Cham, 2016.
- [10] Ángel Faraldo, Sergi Jordà, and Perfecto Herrera. A Multi-Profile Method for Key Estimation in EDM. In *Proceedings of the AES International Conference on Semantic Audio*, Erlangen, Germany, June 2017.
- [11] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167*, March 2015.
- [12] Filip Korzeniowski and Gerhard Widmer. Feature Learning for Chord Recognition: The Deep Chroma Extractor. In *17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, USA, August 2016.
- [13] Filip Korzeniowski and Gerhard Widmer. A Fully Convolutional Deep Auditory Model for Musical Chord Recognition. In *26th IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Salerno, Italy, September 2016.
- [14] Filip Korzeniowski and Gerhard Widmer. End-to-End Musical Key Estimation Using a Convolutional Neural Network. In *25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece, August 2017.
- [15] Shiyu Liang and R. Srikant. Why Deep Neural Networks for Function Approximation? In *International Conference on Learning Representations (ICLR)*, *arXiv:1610.04161*, Toulon, France, April 2017.
- [16] Katy Noland and Mark Sandler. Signal Processing Parameters for Tonality Estimation. In *Audio Engineering Society Convention 122*. Audio Engineering Society, May 2007.
- [17] Steffen Pauws. Musical Key Extraction From Audio. In *Proceedings of the 5th International Society for Music Information Retrieval Conference (ISMIR)*, Barcelona, Spain, October 2004.
- [18] Hendrik Schreiber. MIREX 2017: CNN-Based Automatic Musical Key Detection Submissions HS1/HS2/HS3. Technical report, MIREX, 2017.

- [19] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. In *International Conference on Learning Representations (ICLR), Workshop Track*, *arXiv:1412.6806*, May 2014.
- [20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [21] David Temperley. What’s Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered. *Music Perception: An Interdisciplinary Journal*, 17(1):65–100, October 1999.

DEEP WATERSHED DETECTOR FOR MUSIC OBJECT RECOGNITION

Lukas Tuggener^{1,3} Ismail Elezi^{1,2}
Jürgen Schmidhuber³ Thilo Stadelmann¹

¹ ZHAW Datalab, Zurich University of Applied Sciences, Winterthur, Switzerland

² Dept. of Environmental Sciences, Informatics and Statistics, Ca'Foscari University of Venice, Italy

³ Faculty of Informatics, Università della Svizzera Italiana, Lugano, Switzerland

tugg@zhaw.ch, ismail.elezi@unive.it, juergen@idsia.ch, stdm@zhaw.ch

ABSTRACT

Optical Music Recognition (OMR) is an important and challenging area within music information retrieval, the accurate detection of music symbols in digital images is a core functionality of any OMR pipeline. In this paper, we introduce a novel object detection method, based on synthetic energy maps and the watershed transform, called Deep Watershed Detector (DWD). Our method is specifically tailored to deal with high resolution images that contain a large number of very small objects and is therefore able to process full pages of written music. We present state-of-the-art detection results of common music symbols and show DWD's ability to work with synthetic scores equally well as with handwritten music.

1. INTRODUCTION AND PROBLEM STATEMENT

The goal of Optical Music Recognition (OMR) is to transform images of printed or handwritten music scores into machine readable form, thereby understanding the semantic meaning of music notation [2]. It is an important and actively researched area within the music information retrieval community. The two main challenges of OMR are: first the accurate detection and classification of music objects in digital images; and second, the reconstruction of valid music in some digital format. This work is focusing solely on the first task, meaning that we recover position and class (based on the shape only) of every object without inferring any higher level information.

Recent progress in computer vision [9] thanks to the adaptation of convolutional neural networks (CNNs) [8, 15] provide a solid foundation for the assumption that OMR systems can be drastically improved by using CNNs as well. Initial results of applying deep learning [26] to heavily restricted settings such as staffline removal [25], symbol classification [20] or end-to-end OMR for monophonic scores [5], support such expectations.

In this paper, we introduce a novel general object detection method called Deep Watershed Detector (DWD) motivated by the following two hypotheses: a) deep learning can be used to overcome the classical OMR approach of having hand-crafted pipelines of many preprocessing steps [21] by being able to operate in a fully data-driven fashion; b) deep learning can cope with larger, more complex inputs than simple glyphs, thereby learning to recognize musical symbols in their context. This will disambiguate meanings (e.g., between staccato and augmentation dots) and allow the system to directly detect a complex alphabet.

DWD operates on full pages of music scores in one pass without any preprocessing besides interline normalization and detects handwritten and digitally rendered music symbols without any restriction on the alphabet of symbols to be detected. We further show that it learns meaningful representation of music notation and achieves state-of-the-art detection rates on common symbols.

The remaining structure of this paper is as follows: Sec. 2 puts our approach in context with existing methods; in Sec. 3 we derive our original end-to-end model, and give a detailed explanation on how we use the deep watershed transform for the task of object recognition; Sec. 4 reports on experimental results of our system on the *DeepScores* digitally rendered dataset in addition to the *MUSCIMA++* handwritten dataset before we conclude in Sec. 5 with a discussion and give pointers for future research.

2. RELATED WORK

The visual detection and recognition of objects is one of the most central problems in the field of computer vision. With the recent developments of CNNs, many competing CNN-based approaches have been proposed to solve the problem. R-CNNs [10], and in particular their successors [23], are generally considered to be state-of-the-art models in object recognition, and many developed recognition systems are based on R-CNN. On the other hand, researchers have also proposed models which are tailored towards computational efficiency instead of detection accuracy. YOLO systems [22] and Single-Shot Detectors [18] while slightly compromising on accuracy, are significantly faster than R-CNN models, and can even achieve super real-time performance.

A common aspect of the above-mentioned methods is



© Lukas Tuggener, Ismail Elezi, Jürgen Schmidhuber, Thilo Stadelmann. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Lukas Tuggener, Ismail Elezi, Jürgen Schmidhuber, Thilo Stadelmann. "Deep Watershed Detector for Music Object Recognition", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

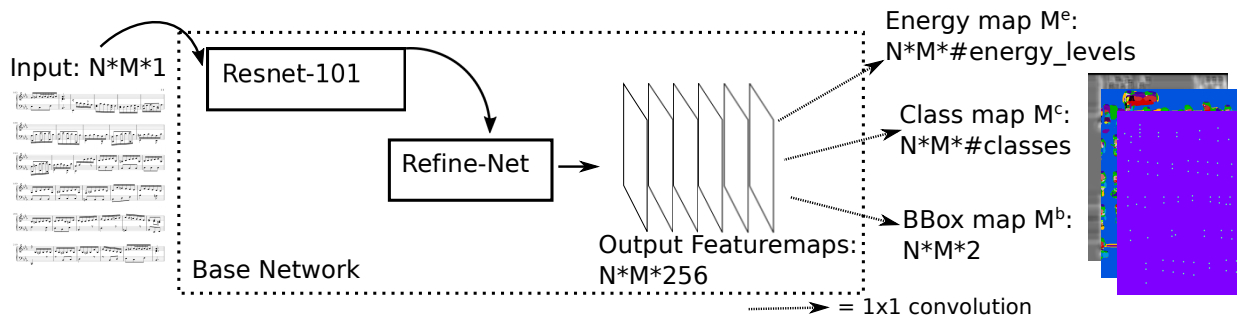


Figure 1. Illustration of the DWD network and its sub-components together with input and outputs. The outputs have been cropped to improve visibility

that they are specifically developed to work on cases where the images are relatively small, and where images contain a small number of relatively large objects [7, 17]. On the contrary, musical sheets usually have high-resolution, and contain a very large number of very small objects, making the mentioned methods not suitable for the task.

The watershed transform is a well understood method that has been applied to segmentation for decades [4]. Bai and Urtasun [1] were first to propose combining the strengths of deep learning with the power of this classical method. They proposed to directly learn the energy (in our application the distance to an object center) for the watershed transform such that all dividing ridges are at the same height. As a consequence, the components can be extracted by a cut at a single energy level without leading to over-segmentation. The model has been shown to achieve state of the art performance on object segmentation.

For the most part, OMR detectors have been rule-based systems working well only within a hard set of constraints [21]. Typically, they require domain knowledge, and work well only on simple typeset music scores with a known music font, and a relatively small number of classes [24]. When faced with low-quality images, complex or even handwritten scores [3], the performance of these models quickly degrades, to some degree because errors propagate from one step to another [20]. Additionally, it is not clear what to do when the classes change, and in many cases, this requires building the new model from scratch.

In response to the above mentioned issues some deep learning based, data driven approaches have been developed. Hajic and Pecina [13] proposed an adaptation of Faster R-CNN with a custom region proposal mechanism based on the morphological skeleton to accurately detect noteheads, while Choi et al. [6] were able to detect accidentals in dense piano scores with high accuracy, given previously detected noteheads, that are being used as input-features to the network. A big limitation of both approaches is that the experiments have been done only on a tiny vocabulary of the musical symbols, and therefore their scalability remains an open question.

To our knowledge, the best results so far has been reported in the work of Pacha and Choi [19] where they explored many models on the *MUSCIMA++* [11] dataset of handwritten music notation. They got the best results with

a Faster R-CNN model, achieving an impressive score on the standard mAP metric. A serious limitation of that work is that the system was not designed in an end-to-end fashion and needs heavy pre- and post-processing. In particular, they cropped the images in a context-sensitive way, by cutting images first vertically and then horizontally, such that each image contains exactly one staff and has a width-to-height-ratio of no more than 2 : 1, with about 15% horizontal overlap to adjacent slices. In practice, this means that all objects significantly exceeding the size of such a cropped region will neither appear in the training nor testing data, as only annotations that have an intersection-over-area of 0.8 or higher between the object and the cropped region are considered part of the ground truth. Furthermore, all the intermediate results must be combined to one concise final prediction, which is a non-trivial task.

3. DEEP WATERSHED DETECTION

In this section we present the Deep Watershed Detector (DWD) as a novel object detection system, built on the idea of the deep watershed transform [1]. The watershed transform [4] is a mathematically well understood method with a simple core idea that can be applied to any topological surface. The algorithm starts filling up the surface from all the local minima, with all the resulting basins corresponding to connected regions. When applied to image gradients, the basins correspond to homogeneous regions of said image (see Fig. 2a). One key drawback of the watershed transform is its tendency to over segment. This issue can be addressed by using the deep watershed transform. It combines the classical method with deep learning by training a deep neural network to create an energy surface based on an input image. This has the advantage that one can design the energy surface to have certain properties. When designed in a way that all segmentation boundaries have energy zero, the watershed transform is reduced to a simple cutoff at a fixed energy level (see Fig. 2b). An objectness energy of this fashion has been used by Bai and Urtasun for instance segmentation [1]. Since we want to do object detection, we further simplify the desired energy surface to having small conical energy peaks of radius n pixels at the center of each object and be zero everywhere else (see Fig. 2c).

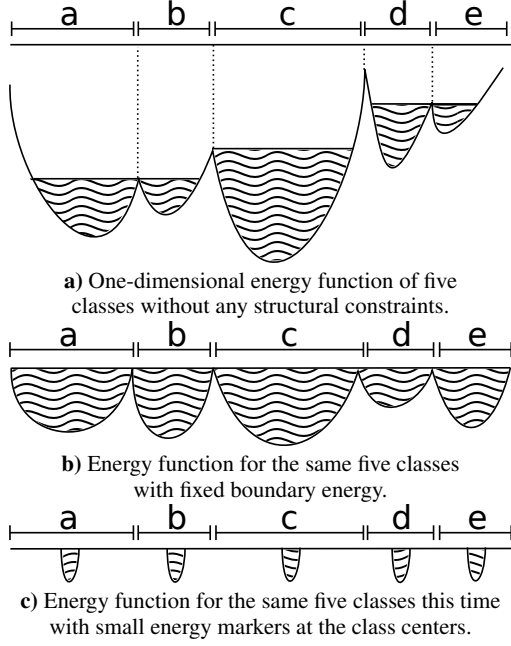


Figure 2. Illustration of the watershed transform applied to different one-dimensional functions.

More formally, we define our energy surface (or: energy map) M^e as follows:

$$M^e_{(i,j)} = \max \begin{cases} \operatorname{argmax}_{c \in C} [E_{max} \cdot (1 - \frac{\sqrt{(i-c_i)^2 + (j-c_j)^2}}{r})] \\ 0 \end{cases} \quad (1)$$

where $M^e_{(i,j)}$ is the value of M^e at position (i, j) , C is the set of all object centers and c_i, c_j are the center coordinates of a given center c . E_{max} corresponds to the maximum energy and r is the radius of the center marking.

At first glance this definition might lead to the misinterpretation that object centers that are closer together than r cannot be disambiguated using the watershed transform on M^e . This is not the case since we can cut the energy map at any given energy level between 1 and E_{max} . However, using this method it is not possible to detect multiple bounding boxes that share the exact same center.

3.1 Retrieving Object Centers

After computing an estimate \hat{M}^e of the energy map, we retrieve the coordinates of detected objects by the following steps:

1. Cut the energy map at a certain fixed energy level and then binarize the result.
2. Label the resulting connected components, using the two-pass algorithm [30]. Every component receives a label l in $1..n$, for every component o^l we define O^l_{ind} as the set of all tuples (i, j) for which the pixel with coordinates j and i is part of o^l .

3. The center \hat{c}^l of any component o^l is given by its center of gravity:

$$\hat{c}^l = o^l_{center} = |O^l_{ind}|^{-1} \cdot \sum_{(i,j) \in O^l_{ind}} (i, j) \quad (2)$$

We use these component centers \hat{c} as estimates for the object centers c .

3.2 Object Class and Bounding Box

In order to recover bounding boxes we do not only need the object centers, but also the object classes and bounding box dimensions. To achieve this we output two additional maps M^c and M^b as predictions of our network. M^c is defined as:

$$M^c_{(i,j)} = \begin{cases} \Lambda_{(i,j)}, & \text{if } M^e_{(i,j)} > 0 \\ \Lambda_{background}, & \text{otherwise} \end{cases} \quad (3)$$

where $\Lambda_{background}$ is the class label indicating background and $\Lambda_{(i,j)}$ is the class label associated with the center c that is closest to (i, j) . We define our estimate for the class of component o^l by a majority vote of all values $\hat{M}^c_{(i,j)}$ for all $(i, j) \in O^l_{ind}$, where \hat{M}^c is the estimate of M^c . Finally, we define the bounding box map M^b as follows:

$$M^b_{(i,j)} = \begin{cases} (y^l, x^l), & \text{if } M^c_{(i,j)} > 0 \\ (0, 0), & \text{otherwise} \end{cases} \quad (4)$$

where y^l and x^l are the width and height of the bounding box for component o^l . Based on this we define our bounding box estimation as the average of all estimations for label l :

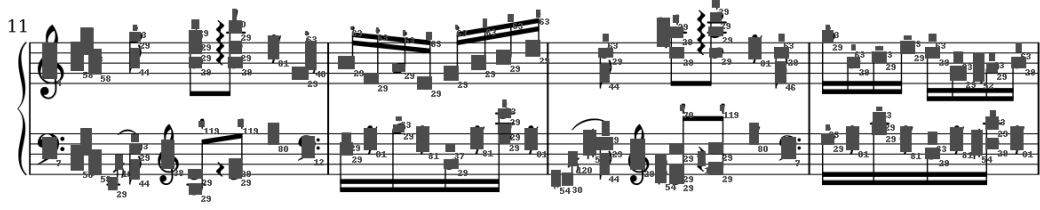
$$(\hat{y}^l, \hat{x}^l) = |O^l_{ind}|^{-1} \cdot \sum_{(i,j) \in O^l_{ind}} \hat{M}^b_{(i,j)} \quad (5)$$

3.3 Network Architecture and Losses

As mentioned above we use a deep neural network to predict the dense output maps M^e , M^c and M^b (see Fig. 1). The base neural network for this prediction can be any fully convolutional network with the same input and output dimensions. We use a ResNet-101 [12] (a special case of a Highway Net [27]) in conjunction with the elaborate RefineNet [16] upsampling architecture. For the estimators defined above it is crucial to have the highest spacial prediction resolution possible. Our network has three output layers, all of which are an 1 by 1 convolution applied to the last feature map of the RefineNet.

3.3.1 Energy prediction

We predict a quantized and one-hot encoded version of M^e , called M^{eo} , by applying a 1 by 1 convolution of depth E_{max} to the last feature map of the base network. The loss of the prediction \hat{M}^{eo} , $loss^e$, is defined as the cross-entropy between M^{eo} and \hat{M}^{eo} .



a) Example result from *DeepScores* with detected bounding boxes as overlays. The tiny numbers are class labels from the dataset introduced with the overlay. This system is roughly one fourth of the size of a typical *DeepScores* input we process at once.



b) Example result from *MUSCIMA++* with detected bounding boxes and class labels as overlays. This system is roughly one half of the size of a typical processed *MUSCIMA++* input. The images are random picks amongst inputs with many symbols.

Figure 3. Detection results for *DeepScores* and *MUSCIMA++* examples, drawn on crops from corresponding input images.

3.3.2 Class prediction

We again use the corresponding one-hot encoded version M^{co} and predict it using an 1 by 1 convolution, with the depth equal to the number of classes, on the last feature map of the base network. The cross-entropy $loss^c$ is calculated between M^{co} and \hat{M}^{co} . Since it is not the goal of this prediction to distinguish between foreground and background, all the loss stemming from locations with $M^e = 0$ will get masked out.

3.3.3 Bounding box prediction

M^b is predicted in its initial form using an 1 by 1 convolution of depth 2 on the last feature map of the base network. The bounding box loss $loss^b$ is the mean-squared difference between M^b and \hat{M}^b . For $loss^b$, the components stemming from background locations will be masked out analogous to $loss^c$.

3.3.4 Combined prediction

We want to jointly train in all tasks, therefore we define a total loss $loss^{tot}$ as:

$$loss^{tot} = w_1 * \frac{loss^e}{v^e} + w_2 * \frac{loss^c}{v^c} + w_3 * \frac{loss^b}{v^b} \quad (6)$$

where the v are running means of the corresponding losses and the scalars w are hyper-parameters of the DWD network. We purposefully use very short extraction heads of one convolutional layer; by doing so we force the base network to do all three tasks simultaneously. We expect this leads to the base network learning a meaningful representation of music notation, from which it can extract the solutions of the three above defined tasks.

4. EXPERIMENTS AND RESULTS

4.1 Used Datasets

For our experiments we use two datasets: *DeepScores* [29] and *MUSCIMA++* [11].

DeepScores is currently the largest publicly available dataset of musical sheets with ground truth for various machine learning tasks, consisting of high-quality pages of written music, rendered at 400 dots per inch. The dataset has 300,000 full pages as images, containing tens of millions of objects, separated in 123 classes. We randomly split the set into training and testing, using 200k images for training and 50k images each for testing and validation. The dataset being so large allows efficient training of large convolutional neural networks, in addition to being suitable for transfer learning [32].

MUSCIMA++ is a dataset of handwritten music notation for musical symbol detection. It contains 91,255 symbols spread into 140 pages, consisting of both notation primitives and higher-level notation objects, such as key signatures or time signatures. It features 105 object classes. There are 23,352 notes in the dataset, of which 21,356 have a full notehead, 1,648 have an empty notehead, and 348 are grace notes. We randomly split the dataset into training, validation, and testing, with the training set consisting of 110 pages, while validation and testing each consists of 15 pages.

4.2 Network Training and Experimental Setup

We pre-train our network in two stages in order to achieve reasonable results. First we train the ResNet on music symbol classification using the *DeepScores* classification dataset [29]. Then, we train the ResNet and RefineNet jointly on semantic segmentation data also available from *DeepScores*. After this pre-training stage we are able to use the network on the tasks defined above in Sec. 3.3.

Since music notation is composed of hierarchically organized sub-symbols, there does not exist a canonical way to define a set of atomic symbols to be detected (e.g., individual numbers in time signatures vs. complete time signatures). We address this issue using a fully data-driven approach by detecting atomic classes as they are provided by the two datasets.

Class	AP@ $\frac{1}{2}$	Class	AP@ $\frac{1}{4}$
rest16th	0.8773	tuplet6	0.9252
noteheadBlack	0.8619	keySharp	0.9240
keySharp	0.8185	rest16th	0.9233
tuplet6	0.8028	noteheadBlack	0.9200
restQuarter	0.7942	accidentalSharp	0.8897
rest8th	0.7803	rest32nd	0.8658
noteheadHalf	0.7474	noteheadHalf	0.8593
flag8thUp	0.7325	rest8th	0.8544
flag8thDown	0.6634	restQuarter	0.8462
accidentalSharp	0.6626	accidentalNatural	0.8417
accidentalNatural	0.6559	flag8thUp	0.8279
tuplet3	0.6298	keyFlat	0.8134
noteheadWhole	0.6265	flag8thDown	0.7917
dynamicMF	0.5563	tuplet3	0.7601
rest32nd	0.5420	noteheadWhole	0.7523
flag16thUp	0.5320	fClef	0.7184
restWhole	0.5180	restWhole	0.7183
timeSig8	0.5180	dynamicPiano	0.7069
accidentalFlat	0.4949	accidentalFlat	0.6759
keyFlat	0.4685	flag16thUp	0.6621

Table 1. AP with overlap 0.5 and overlap 0.25 for the twenty best detected classes of the *DeepScores* dataset.

We rescale every input image to the desired interline value. We use 10 pixels for *DeepScores* and 20 pixels for *MUSCIMA++*. Other than that we apply no preprocessing. We do not define a subset of target objects for our experiments, but attempt to detect all classes for which there is ground truth available. We always feed single images to the network, i.e. we only use batch size = 1. During training we crop the full page input (and the ground truth) to patches of 960 by 960 pixels using randomized coordinates. This serves two purposes: it saves GPU memory and performs efficient data augmentation. This way the network never sees the exact same input twice, even if we train for many epochs. For all of the results described below we train individually on $loss^e$, $loss^c$ and $loss^b$ and then refine the training using $loss^{tot}$. It turns out that the prediction of M^e is the most fragile to effects introduced by training on the other losses, therefore we retrain on $loss^e$ again after training on the individual losses in the order defined above, before moving on to $loss^{tot}$. All the training is done using the RMSProp optimizer [28] with a learning rate of 0.001 and a decay rate of 0.995.

Since our design is invariant to how many objects are present on the input (as long as their centers do not overlap) and we want to obtain bounding boxes for full pages at once, we feed whole pages to the network at inference time. The maximum input size is only bounded by the memory of the GPU. For typical pieces of sheet music this is not an issue, but pieces that use very small interline values (e.g. pieces written for conductors) result in very large inputs due to the interline normalization. At about 10.5 million pixels even a Tesla P40 with 24 gigabytes runs out of memory.

4.3 Results and Discussion

Table 1 shows the average precision (AP) for the twenty best detected classes with an overlap of the detected

Class	AP@ $\frac{1}{2}$	Class	AP@ $\frac{1}{4}$
half-rest	0.8981	whole-rest	0.9762
flat	0.8752	ledger-line	0.9163
natural	0.8531	half-rest	0.8981
whole-rest	0.8226	flat	0.8752
notehead-full	0.8044	natural	0.8711
sharp	0.8033	stem	0.8377
notehead-empty	0.7475	staccato-dot	0.8302
stem	0.7426	notehead-full	0.8298
quarter-rest	0.6699	sharp	0.8121
8th-rest	0.6432	tenuto	0.7903
f-clef	0.6395	notehead-empty	0.7475
numeral-4	0.6391	duration-dot	0.7285
letter-c	0.6313	numeral-4	0.7158
letter-c	0.6313	8th-flag	0.7055
8th-flag	0.6051	quarter-rest	0.6849
slur	0.5699	letter-c	0.6643
beam	0.5188	letter-c	0.6643
time-signature	0.4940	8th-rest	0.6432
staccato-dot	0.4793	beam	0.6412
letter-o	0.4793	f-clef	0.6395

Table 2. AP with overlap 0.5 and overlap 0.25 for the twenty best detected classes from *MUSCIMA++*.

bounding box and ground truth of 50% and 25%, respectively. We observe that in both cases there are common symbol classes that get detected very well, but there is also a steep fall off. The detection rate outside the top twenty continues to drop and is almost zero for most of the rare classes. We further observe that there is a significant performance gain for the lower overlap threshold, indicating that the bounding-box regression is not very accurate.

Fig. 3 shows an example detection for qualitative analysis. It confirms the conclusions drawn above. The rarest symbol present, an arpeggio, is not detected at all, while the bounding boxes are sometimes inaccurate, especially for large objects (note that stems, bar-lines and beams are not part of the *DeepScores* alphabet and hence do not constitute missed detections). On the other hand, staccato dots are detected very well. This is surprising since they are typically hard to detect due to their small size and the context-dependent interpretation of the symbol shape (compare the dots in dotted notes or F-clefs). We attribute this to the opportunity of detecting objects in context, enabled by training on larger parts of full raw pages of sheet music in contrast to the classical processing of tiny, pre-processed image patches or glyphs.

The results for the experiments on *MUSCIMA++* in Tab. 2 and Fig. 3b show a very similar outcome. This is intriguing because it suggests that the difficulty in detecting digitally rendered and handwritten scores might be smaller than anticipated. We attribute this to the fully data-driven approach enabled by deep learning instead of hand-crafted rules for handling individual symbols. It is worth noting that ledger-lines are detected with very high performance (see AP@ $\frac{1}{4}$). This explains the relatively poor detection of note-heads on *MUSCIMA++*, since they tend to overlap.

Fig. 4 shows an estimate for a class map with its corresponding input overlaid. Each color corresponds to one class. This figure proves that the network is learning a sensible representation of music notation: even though it is

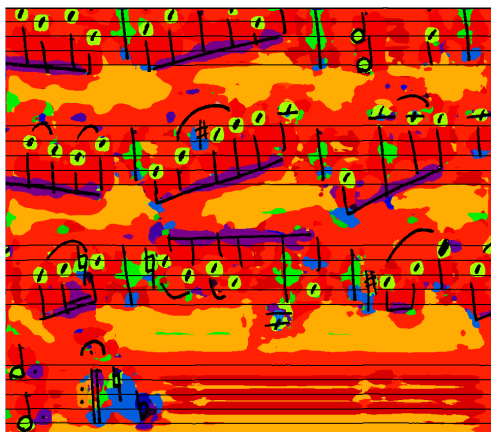


Figure 4. Estimate of a class map \hat{M}^c for every input pixel with the corresponding *MUSCIMA++* input overlaid.

only trained to mark the centers of each object with the correct colors, it learns a primitive segmentation mask. This is best illustrated by the (purple) segmentation of the beams.

5. CONCLUSIONS AND FUTURE WORK

We have presented a novel method for object detection that is specifically tailored to detect many tiny objects on large inputs. We have shown that it is able to detect common symbols of music notation with high precision, both in digitally rendered music as well as in handwritten music, without a drop in performance when moving to the “more complicate” handwritten input. This suggests that deep learning based approaches are able to deal with handwritten sheets just as well as with digitally rendered ones, additionally to their benefit of recognizing objects in their context and with minimal preprocessing as compared to classical OMR pipelines. Pacha et al. [19] show that higher detection rates, especially for uncommon symbols, are possible when using R-CNN on small snippets (cp. Fig. 5). Despite their higher scores, it is unclear how recognition performance is affected when results of overlapping and potentially disagreeing snippets are aggregated to full page results. A big advantage of our end-to-end system is the complete avoidance of error propagation in longer recognition pipeline of independent components like classifiers, aggregators, etc [14]. Moreover, our full-page end-to-end approach has the advantages of speed (compared to a sliding window patch classifier), change of domain (we use the same architecture for both the digital and handwritten datasets) and is easily integrated into complete OMR frameworks.

Arguably the biggest problem we faced is that symbol classes in the dataset are heavily unbalanced. In the *DeepScores* dataset in particular, the class *notehead* contains more than half of all the symbols in the entire dataset, while the top 10 classes contain more than 85% of the symbols. Considering that we did not do any class-balancing whatsoever, this imbalance had its effect in training. We

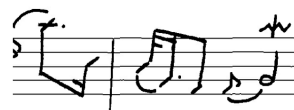


Figure 5. Typical input snippet used by Pacha et al. [19]



Figure 6. Evolution of $loss^b$ (on the ordinate) of a sufficiently trained network, when training for another 8000 iterations (on the abscissa).

observe that in cases where the symbol is common, we get a very high average precision, but it quickly drops when symbols become less common. Furthermore, it is interesting to observe that the neural network actually forgets about the existence of these rarer symbols: Fig. 6 depicts the evolution of $loss^b$ of a network that is already trained and gets further trained for another 8,000 iterations. When faced with an image containing rare symbols, the initial loss is larger than the loss on more common images. But to our surprise, later during the training process, the loss actually increases when the net encounters rare symbols again, giving the impression that the network is actually treating these symbols as outliers and ignoring them.

Future work will thus concentrate on dealing with the catastrophic imbalance in the data to successfully train DWD to detect all classes. We believe that the solution lies in a combination of data augmentation and improved training regimes (i.e. sample pages containing rare objects more often, synthesizing mock pages filled with rare objects etc.).

Additionally, we plan to investigate the ability of our method beyond OMR on natural images. Initially we will approach canonical datasets like *PASCAL VOC* [7] and *MS-COCO* [17] that have been at the front-line of object recognition tasks. However, images in those datasets are not exactly natural, and for the most part they are simplistic (small images, containing a few large objects). Recently, researchers have been investigating the ability of state-of-the-art recognition systems on more challenging natural datasets, like *DOTA* [31], and unsurprisingly, the results leave much to be desired. The *DOTA* dataset shares a lot of similarities with musical datasets, with images being high resolution and containing hundreds of small objects, making it a suitable benchmark for our DWD method to recognize tiny objects.

Acknowledgements This work is financially supported by CTI grant 17963.1 PFES-ES “DeepScore”. The authors are grateful for the collaboration with ScorePad AG.

6. REFERENCES

- [1] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2858–2866, 2017.
- [2] David Bainbridge and Tim Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35.2:95 – 121, 2001.
- [3] Arnau Baro, Pau Riba, and Alicia Fornés. Towards the recognition of compound music notes in handwritten music scores. In *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26, 2016*, pages 465–470, 2016.
- [4] Serge Beucher et al. The watershed transformation applied to image segmentation. *SCANNING MICROSCOPY-SUPPLEMENT*, pages 299–299, 1992.
- [5] Jorge Calvo-Zaragoza, Jose J. Valero-Mas, and Antonio Pertusa. End-to-end optical music recognition using neural networks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 472–477, 2017.
- [6] Kwon-Young Choi, Bertrand Coüasnon, Yann Ricquebourg, and Richard Zanibbi. Bootstrapping samples of accidentals in dense piano scores for cnn-based detection. In *12th International Workshop on Graphics Recognition, 14th IAPR International Conference on Document Analysis and Recognition, GREC@ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, pages 19–20, 2017.
- [7] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.
- [8] Kuniyiko Fukushima and Sei Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*, 15(6):455–469, 1982.
- [9] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. <https://github.com/facebookresearch/detectron>, 2018.
- [10] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 580–587, 2014.
- [11] Jan Hajic and Pavel Pecina. The MUSCIMA++ dataset for handwritten optical music recognition. In *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, pages 39–46, 2017.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.
- [13] Jan Hajic Jr. and Pavel Pecina. Detecting noteheads in handwritten scores with convnets and bounding box regression. *CoRR*, abs/1708.01806, 2017.
- [14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [15] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.
- [16] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5168–5177, 2017.
- [17] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, pages 740–755, 2014.
- [18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, pages 21–37, 2016.
- [19] Alexander Pacha, Kwon-Young Choi, Bertrand Coüasnon, Yann Ricquebourg, and Richard Zanibbi. Handwritten music object detection: Open issues and baseline results. In *Proceedings of the 13th IAPR International Workshop on Document Analysis Systems, Vienna, April 2018*.
- [20] Alexander Pacha and Horst Eidenberger. Towards self-learning optical music recognition. In *16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017, Cancun, Mexico, December 18-21, 2017*, pages 795–800, 2017.
- [21] Ana Rebelo, G. Capela, and Jaime S. Cardoso. Optical recognition of music symbols - A comparative study. *IJDAR*, 13(1):19–31, 2010.

- [22] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6517–6525, 2017.
- [23] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 91–99, 2015.
- [24] Florence Rossant and Isabelle Bloch. Robust and adaptive OMR system including fuzzy modeling, fusion of musical rules, and possible error detection. *EURASIP J. Adv. Sig. Proc.*, 2007, 2007.
- [25] Antonio Javier Gallego Sánchez and Jorge Calvo-Zaragoza. Staff-line removal with selectional auto-encoders. *Expert Syst. Appl.*, 89:138–148, 2017.
- [26] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.
- [27] Rupesh Kumar Srivastava, Klaus Greff, and Juer-gen Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.
- [28] Tijmen Tieleman and Geoffrey E. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning 4.2*, pages 26–31, 2012.
- [29] Lukas Tuggener, Ismail Elezi, Jurgen Schmidhuber, Marcello Pelillo, and Thilo Stadelmann. Deepscores - a dataset for segmentation, detection and classification of tiny objects. *International Conference on Pattern Recognition*, 2018. Preprint arXiv:1804.00525 [cs.CV], March 2018.
- [30] Kesheng Wu, Ekow Otoo, and Kenji Suzuki. Optimizing two-pass connected-component labeling algorithms. *Pattern Anal. Appl.*, 12(2):117–135, February 2009.
- [31] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge J. Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. DOTA: A large-scale dataset for object detection in aerial images. *CoRR*, abs/1711.10398, 2017.
- [32] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3320–3328, 2014.

Session C

Source separation, symbolic, emotion

DEEP NEURAL NETWORKS WITH VOICE ENTRY ESTIMATION HEURISTICS FOR VOICE SEPARATION IN SYMBOLIC MUSIC REPRESENTATIONS

Reinier de Valk
Jukedeck Ltd.
reinier@jukedeck.com

Tillman Weyde
Department of Computer Science
City, University of London
t.e.veyde@city.ac.uk

ABSTRACT

In this study we explore the use of deep feedforward neural networks for voice separation in symbolic music representations. We experiment with different network architectures, varying the number and size of the hidden layers, and with dropout. We integrate two voice entry estimation heuristics that estimate the entry points of the individual voices in the polyphonic fabric into the models. These heuristics serve to reduce error propagation at the beginning of a piece, which, as we have shown in previous work, can seriously hamper model performance.

The models are evaluated on the 48 fugues from Johann Sebastian Bach’s *The Well-Tempered Clavier* and his 30 inventions—a dataset that we curated and make publicly available. We find that a model with two hidden layers yields the best results. Using more layers does not lead to a significant performance improvement. Furthermore, we find that our voice entry estimation heuristics are highly effective in the reduction of error propagation, improving performance significantly. Our best-performing model outperforms our previous models, where the difference is significant, and, depending on the evaluation metric, performs close to or better than the reported state of the art.

1. INTRODUCTION

In the domain of symbolic music representation, the term *voice separation* denotes the identification of individual lines (*voices*) in polyphonic music. More formally, it can be defined as “the task of separating a musical work consisting of multi-note sonorities into independent constituent voices” [3]. With regard to the term *voice* itself, whose meaning is left ambiguous in the above definition, a distinction can be made between (i) a voice as a monophonic sequence of successive, non-overlapping notes, and (ii) a voice as a perceptually independent, but not necessarily monophonic, sequence of notes or multi-note simultaneities [3]. The former definition corresponds to the

music-theoretical notion of a voice (also *part*) [3, 9], while the latter corresponds to the music-psychological notion of an *auditory stream* [2]. For certain genres of music—e.g., piano sonatas or string quartets—it is more appropriate to think of the polyphonic fabric as consisting of multiple streams that may or may not be (partly) monophonic.

Voice separation, especially in monotimbral polyphonic music (e.g., harpsichord or lute music) for more than three concurrent voices, has been recognised as a difficult task even for professional musicians [15, 16, 30]. From a music information retrieval (MIR) perspective, voice separation is considered a challenge that has not yet been addressed satisfactorily. It is, however, an important task: an adequate identification of the individual voices is a prerequisite for tackling several open MIR and musicological problems, such as automatic transcription [1], pattern retrieval [11, 26, 29], and melodic querying [24, 36].

Over the past decade, deep neural networks (DNNs) have been successfully applied to various computer vision, speech recognition, and natural language processing tasks, and, increasingly, to MIR tasks [5]. Consisting of multiple processing layers, DNNs can learn representations of data with multiple levels of abstraction [25], which makes them better suited than their shallow counterparts to model complex input-output relationships. Despite their successful application to a number of MIR tasks, DNNs have not yet been used for voice separation.

The main contributions of this paper are:

- the implementation and evaluation of DNNs for voice separation in symbolic music representations;
- the implementation and evaluation of improved *voice entry estimation heuristics*;
- the creation of a public benchmark dataset for voice separation, which currently does not exist.

We show that a model that combines a DNN with the voice entry estimation heuristics performs close to or better than the reported state of the art.

In what follows, in Section 2, related work is discussed. In Section 3, the model and the integrating framework are presented, and in Section 4, the evaluation method is explained. Section 5 is dedicated to the voice entry estimation heuristics, and Section 6 to the dataset. In Section 7, the experimental results are discussed, and in Section 8, conclusions and directions for future work are presented.



© Reinier de Valk, Tillman Weyde. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Reinier de Valk, Tillman Weyde. “Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

2. RELATED WORK

The existing models addressing the task of voice separation can be divided into two categories: rule-based models and machine learning models. A characteristic that the models in both categories share is that they almost all lean heavily on at least one of two perceptual principles fundamental in auditory stream segregation, coined the *Pitch Proximity Principle* and the *Principle of Temporal Continuity* in [16]. These principles dictate that the closer two notes are to one another in terms of pitch or time, respectively, the more likely they are perceived as belonging to the same voice.

2.1 Rule-based models

The rule-based models form the largest category, containing a wide array of approaches. In [34], a preference rule system for contrapuntal analysis is presented. *Preference rules* are criteria by which a possible analysis is evaluated. Dynamic programming techniques are used to limit the amount of possible analyses to be evaluated. In later work [35], a probabilistic model of polyphonic music analysis, incorporating a stream segregation component that builds on the earlier work, is introduced. Inspired by [34] is the algorithm presented in [27], which consists of a voice configuration unit generating well-formed local solutions, and a note assignment unit calculating a preferred solution.

In [4], a *contig* mapping approach is presented, in which the music is divided into segments where a constant number of voices is active, the *contigs*. Starting from the *contigs* where the number of voices active equals the nominal number of voices, the optimal connections to the neighbouring *contigs* are determined. Gradually branching out, this process is repeated until all *contigs* are connected. A modified version of this approach is proposed in [17], where the connection of *contigs* that share a boundary at which the number of voices increases is prioritised. The idea is that the distinctiveness (in terms of pitch distance) of the new voice will prevent it from being connected incorrectly to one of the voices active in the smaller-size *contig*. A further improvement of the approach is described in [14], where, taking into account more context information, additional criteria that underly the *contig* connection policy are proposed. The criteria are weighted using a genetic algorithm with mutation and crossover operators.

In [33], voice separation is modelled as a clustering problem. Using an agglomerative single-link clustering algorithm, in an iterative process that starts from an initial distribution in which each note is a cluster, all clusters are combined into larger clusters until n simultaneous clusters, the voices, remain. In [10], the music is modelled as a directed graph. The goal is to create a set of disjoint paths, the voices, through the graph. To this end, the graph is divided into segments, which are analysed through constraint satisfaction optimisation. Using a sequence alignment algorithm, the analyses are then connected.

Two models stand out as they allow for non-monophonic voices. In the local optimisation approach proposed in [21], a piece is partitioned into slices that are processed iteratively, assigning the notes to voices. A

stochastic local search algorithm is used to find assignments that minimise a parametric cost function assessing the assignments; weighting the parameters in a certain way can result in non-monophonic assignments. In the Voice Integration/Segregation Algorithm (VISA) as proposed in [19, 20] and later refined in [31], *vertical integration*—concurrent notes with the same onset and duration merging perceptually into a single sonority—is considered to be prior to *horizontal integration*—successive notes close in pitch and time merging perceptually into a single voice. VISA thus first identifies concurrent notes that merge into single sonorities, and, using a bipartite matching algorithm, then assigns the sonorities to separate streams.

2.2 Machine learning models

In [23], VoiSe, a system for separating voices in both implicit and explicit polyphony, is presented. The system consists of two components: a same-voice predicate implemented as a learned decision tree, which determines whether or not two notes belong to the same voice, and a hard-coded algorithm that maps notes to voices.

A probabilistic, Markov chain-like, system is proposed in [18]. Based on pitch information only, the system learns how likely a note is to occur for a voice, as well as how likely a transition between two notes is to occur. The system is inspired by [4] in that the music is processed in a similar manner—starting at chords in which all voices are present. Another probabilistic approach is described in [6], where the music is represented as a sequence of chords, and a discrete hidden Markov model (HMM) is used to determine the most likely sequence of mappings to voices (the hidden states) for the chords (the observations). A similar, although more sophisticated, approach using an HMM is proposed in [28]. This model explicitly allows notes within a single voice to overlap. This not only makes preprocessing (quantisation) redundant, but also enables application to data generated from live performance.

In [6], the task of voice separation is modelled both as a multi-class classification problem (see also [7]), where the music is represented as a sequence of notes, which are assigned to voices (the classes), and as a regression problem, where the music is represented as a sequence of chords, for which mappings to voices are rated. Standard single-hidden layer feedforward neural networks are used as the classifier and regressor, respectively. In [13], too, the music is represented as a sequence of chords, and a single-hidden layer feedforward neural network is used to greedily assign each chord note to the voice that maximises a trained assignment probability.

3. PROBLEM FORMULATION, MODEL, AND FRAMEWORK

As in [6, 7], in this paper we formulate the task of voice separation as a multi-class classification problem, where each note in a piece is assigned to one of v voices (the classes). We assume that a voice is always monophonic

(see Section 1), and that the number of voices in a piece is equal to its nominal number of voices, which we infer from the size of its largest chord. (These assumptions do not always hold true, but in pure contrapuntal music one generally finds only few exceptions. We resolve such cases by removing the offending notes from the dataset; this is discussed in Section 6.) Furthermore, for practical reasons we set the maximum value of v to 5. This enables us to process pieces containing up to five voices, which currently suffices. The maximum number of voices determines the number of classes and hence the size of the neural network’s output layer; for the sake of efficiency, it should thus be kept as small as possible.

3.1 Model

We use the open source TensorFlow machine learning library¹ (version 1.6.0) to implement a multi-layer deep feedforward neural network that uses the rectified linear unit activation function for all $L - 1$ hidden layers and the softmax activation function for the output layer, and that has five output neurons, each representing a class. Given that our dataset is relatively small, we use batch training, where we check the performance on the validation set (comprising every fifth training example) every 10 epochs as early stopping strategy, and store the earliest best-performing model. We use Xavier initialisation [12] for the weights and initialisation with zeros for the biases, the Adam optimisation algorithm [22] to minimise the cross-entropy loss, and dropout [32] to prevent overfitting. We set the learning rate to 0.01 and the number of training epochs to 600, values we observed to work well. Three further hyperparameters are optimised using a grid search (see Section 7): the dropout *keep probability*, the number of hidden layers, and the size of the hidden layers.

3.2 Framework

We integrate the model in our previously developed framework for data preprocessing, feature extraction, and cross-validated training and evaluation [6], implemented in Java.² In this framework, the music is represented as a sequence of notes, by default ordered by (i) onset time (low to high) and (ii) pitch (low to high). When evaluating the model, this sequence is processed in linear fashion, where for each note a feature vector is calculated that is given as input to the model, which then makes a class decision—thus assigning the note to a voice.

3.2.1 Feature vector

Each note is represented by a 33-dimensional feature vector, containing properties of that note in its polyphonic context. The features are handcrafted and can be divided into four categories of increasing scope: (i) note-level features, encoding individual properties of the note; (ii) note-chord features, encoding the note’s position in the chord; (iii) chord-level features, encoding properties shared by all

notes in the chord; and (iv) polyphonic embedding features, encoding the note’s polyphonic relation to the notes in the previous as well as the current chord. All feature values (except certain default values) are scaled to fall in the range [0, 1]. An overview is presented in Table 1; more detail is provided in [6].

Index	Feature	Description
0	<i>pitch</i>	pitch, as a MIDI number
1	<i>duration</i>	duration, in whole notes
2	<i>isOrnamentation</i>	true (1) if a 16th note or shorter, false (0) if not
3	<i>indexInChord</i>	index (pitch-based) in the chord
4	<i>pitchDistBelow</i>	distance to note below
5	<i>pitchDistAbove</i>	distance to note above
6	<i>chordSize</i>	number of chord notes
7	<i>metricPosition</i>	metric position in the bar
8	<i>numNotesNext</i>	number of notes (onsets) in the next chord
9-12	<i>intervals</i>	intervals in the chord
13-17	<i>pitchProx</i>	for each voice v , the pitch proximity to the adjacent left note in v
18-22	<i>interOnsetProx</i>	idem, inter-onset
23-27	<i>offsetOnsetProx</i>	idem, offset-onset
28-32	<i>voicesOccupied</i>	for each voice v , whether it is currently occupied (1) or not (0)

Table 1. The feature vector, containing note-level (0-2), note-chord (3-5), chord-level (6-12), and polyphonic embedding features (13-32). Pitch distances and intervals are measured in semitones; proximities are inverted distances.

4. EVALUATION

We evaluate the models using k -fold cross-validation. Because it is not desirable that identical or highly similar samples extracted from one piece end up in both the training and the test set, we partition a dataset along its individual pieces rather than randomly. k thus equals the number of pieces in a dataset; each piece in it serves as test set once.

4.1 Evaluation metrics

We use four metrics to assess model performance. *Accuracy* is a per-note metric that measures the proportion of notes that have been assigned to the correct voice:

$$\text{acc} = \frac{|C|}{|N|}, \quad (1)$$

where C is the set of notes assigned to the correct voice, and N the set of all notes.

Soundness and *completeness* are complementary metrics that measure transitions between note pairs. We use the definitions provided in [23]. If f is an assigned

¹ <https://www.tensorflow.org/>

² https://www.github.com/reinierdevalk/voice_separation/

voice and g a correct voice, then a pair of adjacent notes (n_t, n_{t+1}) in f is considered *sound* if $g(n_t) = g(n_{t+1})$ holds. (Note that, according to this definition, f and g need not be the same voice.) Extending the definition, we take soundness to be the proportion of sound pairs in *all* voices:

$$\text{snd} = \frac{|S|}{|P|}, \quad (2)$$

where S is the set of sound pairs, and P the set of all pairs in all assigned voices f . Similarly, a pair of adjacent notes (n_t, n_{t+1}) in correct voice g is considered *complete* if assigned voice $f(n_t) = f(n_{t+1})$ holds. We take completeness to be the proportion of complete pairs in all voices:

$$\text{cmp} = \frac{|C|}{|P|}, \quad (3)$$

where C is the set of complete pairs, and P the set of all pairs in all correct voices g .³

Average voice consistency (AVC), coined in [4], measures, “on average, the proportion of notes from the same voice that have been assigned ... to the same voice”. The *voice consistency* (VC) for voice v is calculated as follows:

$$\text{VC}(v) = \frac{1}{|S(v)|} \max_{u \in V} \{ |n \in S(v) : vN(n) = u| \}, \quad (4)$$

where $S(v)$ is the set of notes assigned to v , V the set of all voices, and $vN(n)$ the correct voice for note n . The AVC, then, is the average VC over all voices:

$$\text{AVC} = \frac{1}{|V|} \sum_{v \in V} \text{VC}(v). \quad (5)$$

The per-fold percentages for each metric m are weighted by the number of notes (or note pairs) in the piece for the fold, so that the average values over all folds are always per-note (or per-pair):

$$\text{avg}(m) = \frac{\sum_{i=1}^k (m_i \cdot |N_i|)}{\sum_{i=1}^k |N_i|}, \quad (6)$$

where k is the number of folds, and N the set of notes in a piece.

4.2 Evaluation modes

We use two evaluation modes: *test mode* and *application mode*. In test mode, the feature vectors are calculated using the correct voice information for the preceding notes. Test mode serves a gauging function in that it reflects the *optimal* model performance on unseen data. In application mode, the feature vectors are calculated using the model-generated voice information. This mode corresponds to the real-world application scenario where no correct voice information is available, and where all voice decisions must be based on previous decisions—it thus reflects the *expected* model performance on unseen data. In application mode, model performance can suffer from *error propagation*.

³ The definitions are equal to those given for *precision* and *recall* in [10], metrics used in [13, 14, 18, 27, 28]. The terms appear to be used interchangeably.

5. VOICE ENTRY ESTIMATION HEURISTICS

Error propagation is the phenomenon in which an incorrect voice assignment influences the voice decision for the following notes negatively. Given the accuracy in test (acc_T) and application mode (acc_A), the proportion of misassignments due to error propagation, q , is calculated as follows:

$$q = \frac{\text{acc}_T - \text{acc}_A}{1 - \text{acc}_A}. \quad (7)$$

In previous work [6, 7], depending on the dataset we observed q values up to 0.87, indicating that model performance is indeed seriously hampered by error propagation.

Although error propagation can occur throughout a piece, it tends to be particularly strong in thinly-textured openings of pieces, where the model may start ‘on the wrong foot’. This often leads to a chain of misassignments. To address this problem, we propose a preprocessing step that applies two heuristics, h1 and h2 (improving on [8]). They estimate which notes belong to the *new voices* at each *density increase*, that is, each point where the maximal number of simultaneous notes so far increases. h1 and, partly, h2 are based on the prior assumptions that (i) voices tend to move in small steps, and that when new voice(s) enter, (ii) none of the already active voices has a rest, and (iii) none of the voices is involved in voice crossing.

```

01 function estimate(list notes) returns list
02   density increases  $d := [d_1, \dots, d_m]$ 
03   available voices  $av := [1, \dots, d_m]$ 
04   add  $av$  to new list  $fw$ 
05   for  $i$  from  $m$  to 2:
06     if h1: find lowest-cost configuration
07       at  $\text{pos}(d_i)$ 
08     if h2: find pattern at  $\text{pos}(d_i)$ 
09     remove new voices from  $av$ 
10     prepend  $av$  to  $fw$ 
11      $av := \text{copy}(av)$ 
12   return  $fw$ 

```

Figure 1. Algorithm outline. Underlined concepts are explained in the main text.

h1 and h2 share a similar overall algorithmic structure, as shown in Figure 1. The algorithm takes as input the sequence of notes representing a piece (see Section 3.2), and returns, for each density increase (including the opening), a vector of voice assignments for the first chord of the increased density. If the voices enter successively, h2 is called; if not, or if h2 fails, h1 is called. The voice assignments returned remain fixed when the DNN is applied.

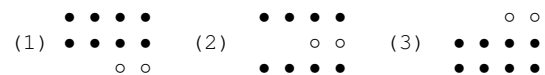


Figure 2. Chord configurations ($n = 2$). Columns represent chords; rows represent layers.

h1 is the more generic heuristic. It determines the new voices by calculating, at each density increase starting at the last, the lowest-cost *configuration*. A configuration organises the last n chords (i.e., ordered sequences of

itches) of density d_{i-1} and the first n chords of density d_i into horizontal layers, as shown in Figure 2. The cost for a configuration is calculated as follows:

$$\sum_{j=1}^n \sum_{k=1}^{d_{i-1}} \sum_{l=1}^n |p_{j,k} - p_{l,k}|, \quad (8)$$

where $p_{j,k}$ is the pitch at the k th $2n$ -sized layer in the j th chord of density d_{i-1} , and $p_{l,k}$ the pitch at the k th $2n$ -sized layer in the l th chord of density d_i . The positions of the remaining n -sized layers in the lowest-cost configuration, then, determine the new voices.

h2 caters specifically to imitative pieces, and attempts to determine the new voices by finding, at each density increase starting at the last, a match for the *pattern* (as defined by the first n notes of the piece, the opening motif’s head) in the first n chords of density d_i . The first matching criterion is rhythmic sequence; if this yields multiple matches, melodic contour (up, same, down) is added as a second matching criterion. If a single match is thus found, the new voice is identified; if multiple matches are still found, the lowest-cost configuration (as in h1) is used to disambiguate. If no match is found, which can happen if the prior assumptions do not hold true, the new voice is assumed to enter below the existing voice(s). If at more than half of the density increases no match is found, h2 fails.

6. DATASET

The models are evaluated on the 48 fugues from Johann Sebastian Bach’s *The Well-Tempered Clavier* (BWV 846-893), containing one two-, 26 three-, 19 four-, and two five-voice pieces, as well as his 30 inventions (BWV 772-801), containing 15 two- and 15 three-voice pieces (also known as *sinfonias*). The dataset, in MIDI format, was originally retrieved from the MuseData repository of the Center for Computer Assisted Research in the Humanities,⁴ and has been slightly modified. First, all *in-voice chords*—instances where a voice is non-monophonic—were reduced to single notes, and all temporarily added extra voices were removed. Figure 3 shows an example of both. Second, because of liberties in performance or rounding errors leading to note overlap within a voice, occasionally some quantisation was required. This was achieved by adjusting each offending left note’s offset to equal its adjacent right note’s onset. These first two modifications are necessary in order for the data to comply with the assumptions that underly our modelling approach (a voice is always monophonic, and the number of voices in a piece is equal to its nominal number of voices—see Section 3). Third, to create a more equal distribution of training and test data in cross-validation, the two-voice fugue was split into two parts, and the two five-voice fugues were split into four (BWV 849) and two (BWV 867) parts. Fourth, for a number of pieces starting with an anacrusis, some padding with rests was required to ensure a correct metrical alignment. Fifth, where necessary, time signature or key signature information was corrected or added.

⁴ <http://www.musedata.org/>



Figure 3. *The Well-Tempered Clavier*, Fugue 17 in A \flat major (BWV 886), closing bars. Temporarily added extra voice, chromatically descending from G $_3$ to E \flat_3 (lower staff), and in-voice chord (upper staff, final chord).

Thus, a total of 206 notes were pruned from the original 53230 notes in the fugues, and a total of five notes from the original 19872 notes in the inventions—yielding a dataset containing 72891 notes. We publish this dataset as a curated benchmark dataset for voice separation,⁵ that enables the comparison of results in a rigorous manner, and that thus facilitates reproducible research [37].

7. EXPERIMENTAL RESULTS AND DISCUSSION

In a first experiment, we performed a grid search to optimise three hyperparameters: the number of hidden layers (HL), the size of the hidden layers (HLS), and the value of the dropout keep probability (KP). We explored a small hyperparameter space determined in earlier experimentation, consisting of four HL values (2, 3, 4, and 5), four HLS values (25, 33, 50, and 66), and three KP values (0.75, 0.875, and 0.9375). The grid search was performed on the 19 four-voice fugues; as the deciding metric, accuracy in test mode was used (metrics in test mode are more stable indicators of model performance; see Section 4.2). For each HL value, we selected the best-performing model, which we then trained and evaluated on all 48 fugues and all 30 inventions. This was done separately on the different subsets (two-voice, three-voice, etc.); the performance on all fugues or inventions is the per-note (or per-pair) average over their subsets as calculated using Equation (6). Table 2 shows that on the fugues, the two-layer model yields the highest performance in both test and application mode. On the inventions, the results are less clear—although the two more shallow models seem to perform better here too. Overall, however, the results are fairly similar, indicating a limited effect of the number of layers.

Focussing on the best model (HL = 2; HLS = 66, KP = 0.875), in a second experiment, we then investigated the effect of using a deep(er) neural network, as well as the effect of the integration of the voice entry estimation heuristics. To this end, we compared four models: the single-hidden layer neural network as described in [6, 7] (N), the same model with the heuristics integrated (N/h), the two-layer model (D), and the two-layer model with the heuristics integrated (D/h). The heuristics were not used in test mode, as error propagation does not occur there. Table 3 shows that D always outperforms N, and that N/h and D/h always outperform N and D, respectively. A test for

⁵ <https://www.github.com/reinierdevalk/data/>

HL	HLS	KP	Test				Application			
			acc	snd	cmp	AVC	acc	snd	cmp	AVC
2	66	0.875	98.34	97.11	97.26	98.27	90.72	96.43	96.39	90.76
3	66	0.75	98.36	97.12	97.24	98.27	89.96	96.30	96.30	90.05
4	50	0.75	98.32	97.07	97.23	98.25	89.97	96.36	96.36	90.12
5	50	0.75	98.27	96.98	97.13	98.20	89.96	96.38	96.35	90.23
2	66	0.875	99.09	98.52	98.58	99.06	96.64	98.14	98.09	96.49
3	66	0.75	99.17	98.64	98.62	99.13	96.53	98.22	98.21	96.35
4	50	0.75	99.20	98.64	98.66	99.15	96.54	98.17	98.13	96.34
5	50	0.75	99.00	98.40	98.39	98.96	96.44	97.95	97.93	96.28

Table 2. Experiment 1. Best-performing models per HL value, 48 fugues (top) and 30 inventions (bottom). Values are averages over the different subsets (see Section 7 and Equation (6)); all values are percentages.

Model	Test				Application					q
	acc	snd	cmp	AVC	acc	snd	cmp	AVC	F_1	
N	97.86	96.54	96.70	97.78	86.36	95.46	95.33	86.73	95.39	0.84
N/h					90.44	95.86	95.69	90.62	95.78	0.77
D	98.34	97.11	97.26	98.27	87.69	96.26	96.20	87.43	96.23	0.86
D/h					90.72	96.43	96.39	90.76	96.41	0.82
[28]								88.23	97.00	
[17]								89.21		
[10]									92.5	

Table 3. Experiment 2 and 3. N, N/h, D, and D/h models, 48 fugues (top); [10, 17, 28] models, 48 fugues (bottom). Values are averages over the different subsets (see Section 7 and Equation (6)); all values except q are percentages. The F_1 score is the harmonic mean of soundness and completeness.

statistical significance (we used the one-tailed Wilcoxon signed-rank test with $p < 0.05$ as the significance criterion) reveals that these performance differences are always significant. We thus conclude that both using a deep(er) neural network and integrating the heuristics yield a significant performance improvement. Furthermore, the q values show that the heuristics indeed reduce error propagation—but the effect is weaker in case of the D model, where error propagation is also slightly worse. Finally, we note that integrating the heuristics leads to a strong improvement in terms of accuracy and AVC. The improvement in terms of soundness and completeness—which are by definition less affected by error propagation—, on the other hand, is only small.

Additionally, we compared the performance of our overall best model (D/h) on the 48 fugues with the performances reported for the three voice separation models that, to our knowledge, represent the current state of the art, and that have also been evaluated on the 48 fugues. As Table 3 shows, D/h outperforms the [17] and [10] models. It also outperforms the [28] model in terms of AVC, but not in terms of F_1 score—which may be because the latter model is specifically optimised for that metric, whereas D/h is optimised for accuracy.

It should be noted, finally, that a strict comparison with the state of the art is problematic due to the heterogeneity of datasets and metrics used. We address this by making our dataset publicly available as a benchmark dataset (see Section 6).

8. CONCLUSIONS AND FUTURE WORK

In this paper, we present the implementation and evaluation of DNNs for voice separation in symbolic music representations as well as the implementation and evaluation of two voice entry estimation heuristics. We evaluate the models on 78 keyboard works by Johann Sebastian Bach, which we publish as a curated benchmark dataset for comparing voice separation models. We observe that both the use of deep(er) neural networks for the task and the integration of the heuristics into the models improve performance significantly. The best model outperforms our previous models, and performs close to or better than the reported state of the art.

A first analysis of the results reveals that the model has difficulties processing musically challenging passages, containing, for example, voice crossings or reduced textures. Furthermore, despite the success of the voice entry estimation heuristics, error propagation remains problematic. An in-depth analysis of the results, planned for future work, is required to gain better insight into these matters. Possible explanations are that the model is not given enough context information, and that it does not have any memory. We therefore also plan to encode a larger polyphonic window into the features as to increase the context information, and we plan to experiment with other types of DNNs, such as recurrent neural networks, which allow information to persist, or long short-term memory models, which are capable of learning long-time dependencies.

9. REFERENCES

- [1] E. Benetos, S. Dixon, D. Giannoulis, Automatic music transcription: Challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.
- [2] A. S. Bregman and J. Campbell. Primary auditory stream segregation and perception of order in rapid sequences of tones. *Journal of Experimental Psychology*, 89(2):244–249, 1971.
- [3] E. Cambouropoulos. Voice and stream: Perceptual and computational modeling of voice separation. *Music Perception*, 26(1):75–94, 2008.
- [4] E. Chew and X. Wu. Separating voices in polyphonic music: A contig mapping approach. In U. K. Wiil, editor, *Computer Music Modeling and Retrieval: Second international symposium, CMMR 2004*, pages 1–20. Springer, Berlin, 2005.
- [5] K. Choi, G. Fazekas, K. Cho, and M. Sandler. A tutorial on deep learning for music information retrieval. *arXiv:1709.04396v1 [cs.CV]*, 2017.
- [6] R. de Valk. *Structuring lute tablature and MIDI data: Machine learning models for voice separation in symbolic music representations*. PhD thesis, City University, London, 2015.
- [7] R. de Valk and T. Weyde. Bringing ‘Musicque into the tabletur’: Machine-learning models for polyphonic transcription of 16th-century lute tablature. *Early Music*, 43(4):563–576, 2015.
- [8] R. de Valk and T. Weyde. Voice entry estimation heuristics to reduce error propagation in voice separation models. In *Proc. of the 18th International Society for Music Information Retrieval Conference, Suzhou, China, Late-Breaking/Demo session*, 2017.
- [9] W. Drabkin. Part (ii). In Stanley Sadie, editor, *The new Grove dictionary of music and musicians*, volume 19, page 164. Macmillan, London, 2nd edition, 2001.
- [10] B. Duane and B. Pardo. Streaming from MIDI using constraint satisfaction optimization and sequence alignment. In *Proc. of the International Computer Music Conference, Montreal, QC, Canada*, 2009.
- [11] M. Giraud, R. Groult, and F. Levé. Subject and counter-subject detection for analysis of the *Well-Tempered Clavier* fugues. In M. Aramaki, M. Barthet, R. Kronland-Martinet, and S. Ystad, editors, *From sounds to music and emotions: 9th international symposium, CMMR 2012*, pages 422–438. Springer, Berlin, 2013.
- [12] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of the 13th International Conference on Artificial Intelligence and Statistics, Sardinia, Italy*, pages 249–256, 2010.
- [13] P. Gray and R. Bunescu. A neural greedy model for voice separation in symbolic music. In *Proc. of the 17th International Society for Music Information Retrieval Conference, New York, NY, USA*, pages 782–788, 2016.
- [14] N. Guimard-Kagan, M. Giraud, R. Groult, and F. Levé. Improving voice separation by better connecting contigs. In *Proc. of the 17th International Society for Music Information Retrieval Conference, New York, NY, USA*, pages 164–170, 2016.
- [15] D. Huron. Voice denumerability in polyphonic music of homogeneous timbres. *Music Perception*, 6(4):361–382, 1989.
- [16] D. Huron. Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception*, 19(1):1–64, 2001.
- [17] A. Ishigaki, M. Matsubara, and H. Saito. Prioritized contig combining to segregate voices in polyphonic music. In *Proc. of the 8th Sound and Music Computing Conference, Padua, Italy*, 2011.
- [18] A. Jordanous. Voice separation in polyphonic music: A data-driven approach. In *Proc. of the International Computer Music Conference, Belfast, Ireland*, 2008.
- [19] I. Karydis, A. Nanopoulos, A. Papadopoulos, E. Cambouropoulos, and Y. Manolopoulos. Horizontal and vertical integration/segregation in auditory streaming: A voice separation algorithm for symbolic musical data. In *Proc. of the 4th Sound and Music Computing Conference, Lefkada, Greece*, pages 299–306, 2007.
- [20] I. Karydis, A. Nanopoulos, A. N. Papadopoulos, and E. Cambouropoulos. VISA: The voice integration/segregation algorithm. In *Proc. of the 8th International Conference on Music Information Retrieval, Vienna, Austria*, pages 445–448, 2007.
- [21] J. Kilian and H. H. Hoos. Voice separation—A local optimization approach. In *Proc. of the 3rd International Conference on Music Information Retrieval, Paris, France*, pages 39–46, 2002.
- [22] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980v9 [cs.LG]*, 2017.
- [23] P. B. Kirlin and P. E. Utgoff. VoiSe: Learning to segregate voices in explicit and implicit polyphony. In *Proc. of the 6th International Conference on Music Information Retrieval, London, UK*, pages 552–557, 2005.
- [24] I. Knopke and F. Jürgensen. A system for identifying common melodic phrases in the masses of Palestrina. *Journal of New Music Research*, 38(2):171–181, 2009.
- [25] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

- [26] D. Lewis, T. Crawford, and D. Müllensiefen. Instrumental idiom in the 16th century: Embellishment patterns in arrangements of vocal music. In *Proc. of the 17th International Society for Music Information Retrieval Conference, New York, NY, USA*, pages 524–530, 2016.
- [27] S. T. Madsen and G. Widmer. Separating voices in MIDI. In *Proc. of the 7th International Conference on Music Information Retrieval, Victoria, BC, Canada*, pages 57–60, 2006.
- [28] A. McLeod and M. Steedman. HMM-based voice separation of MIDI performance. *Journal of New Music Research*, 45(1):17–26, 2016.
- [29] D. Meredith, K. Lemström, and G. A. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4):321–345, 2002.
- [30] N. Orio. Music retrieval: A tutorial and review. *Foundations and Trends in Information Retrieval*, 1(1):1–90, 2006.
- [31] D. Rafailidis, E. Cambouropoulos, and Y. Manolopoulos. Musical voice integration/segregation: VISA revisited. In *Proc. of the 6th Sound and Music Computing Conference, Porto, Portugal*, pages 42–47, 2009.
- [32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(Jun):1929–1958, 2014.
- [33] W. M. Szeto and M. H. Wong. Stream segregation algorithm for pattern matching in polyphonic music databases. *Multimedia Tools and Applications*, 30(1):109–127, 2006.
- [34] D. Temperley. *The cognition of basic musical structures*. MIT Press, Cambridge, MA, 2001.
- [35] D. Temperley. A unified probabilistic model for polyphonic music analysis. *Journal of New Music Research*, 38(1):3–18, 2009.
- [36] G. Velarde, T. Weyde, and D. Meredith. An approach to melodic segmentation and classification based on filtering with the haar-wavelet. *Journal of New Music Research*, 42(4):325–345, 2013.
- [37] M. D. Wilkinson et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(160018), 2016.

MUSIC SOURCE SEPARATION USING STACKED HOURGLASS NETWORKS

Sungheon Park

Taehoon Kim

Kyogu Lee

Nojun Kwak

Graduate School of Convergence Science and Technology, Seoul National University, Korea

{sungheonpark, kcjs55, kglee, nojunk}@snu.ac.kr

ABSTRACT

In this paper, we propose a simple yet effective method for multiple music source separation using convolutional neural networks. Stacked hourglass network, which was originally designed for human pose estimation in natural images, is applied to a music source separation task. The network learns features from a spectrogram image across multiple scales and generates masks for each music source. The estimated mask is refined as it passes over stacked hourglass modules. The proposed framework is able to separate multiple music sources using a single network. Experimental results on MIR-1K and DSD100 datasets validate that the proposed method achieves competitive results comparable to the state-of-the-art methods in multiple music source separation and singing voice separation tasks.

1. INTRODUCTION

Music source separation is one of the fundamental research areas for music information retrieval. Separating singing voice or sounds of individual instruments from a mixture has grabbed a lot of attention in recent years. The separated sources can be further used for applications such as automatic music transcription, instrument identification, lyrics recognition, and so on.

Recent improvements on deep neural networks (DNNs) have been blurring the boundaries between many application domains, including computer vision and audio signal processing. Due to its end-to-end learning characteristic, deep neural networks that are used in computer vision research can be directly applied to audio signal processing area with minor modifications. Since the magnitude spectrogram of an audio signal can be treated as a 2D single-channel image, convolutional neural networks (CNNs) have been successfully used in various music applications, including the source separation task [1, 8]. While very deep CNNs are typically used in computer vision literature with very large datasets [4, 25], CNNs used

for audio source separation so far have relatively shallow architectures.

In this paper, we propose a novel music source separation framework using CNNs. We used stacked hourglass network [18] which was originally proposed to solve human pose estimation in natural images. The CNNs take spectrogram images of a music signal as inputs, and generate masks for each music source to separate. An hourglass module captures both holistic features from low resolution feature maps and fine details from high resolution feature maps. The module outputs 3D volumetric data which has the same width and height as those of the input spectrogram. The number of output channels equals the number of music sources to separate. The module is stacked for multiple times by taking the results of the previous module. As passing multiple modules, the results are refined and intermediate supervision helps faster learning in the initial state. We used a single network to separate multiple music sources, which reduces both time and space complexity for training as well as testing.

We evaluated our framework on a couple of source separation tasks: 1) separating singing voice and accompaniments, and 2) separating bass, drum, vocal, and other sounds from music. The results show that our method outperforms existing methods on MIR-1K dataset [5] and achieves competitive results comparable to state-of-the-art methods on DSD100 dataset [30] despite its simplicity.

The rest of the paper is organized as follows. In Section 2, we briefly review the literature of audio source separation focusing on DNN based methods. The proposed source separation framework and the architecture of the network are explained in Section 3. Experimental results are provided in Section 4, and the paper is concluded in Section 5.

2. RELATED WORK

Non-negative matrix factorization (NMF) [12] is one of the most widely-used algorithms for audio source separation. It has been successfully applied to monaural source separation [32] and singing voice separation [29, 38]. However, despite its generality and flexibility, NMF is inferior to recently proposed DNN-based methods in terms of performance and time complexity.

Simple deep feed-forward networks consisting of multiple fully-connected layers showed reasonable performance for supervised audio source separation tasks [27]. Wang *et*



© Sungheon Park, Taehoon Kim, Kyogu Lee, Nojun Kwak. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Sungheon Park, Taehoon Kim, Kyogu Lee, Nojun Kwak. "Music Source Separation Using Stacked Hourglass Networks", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

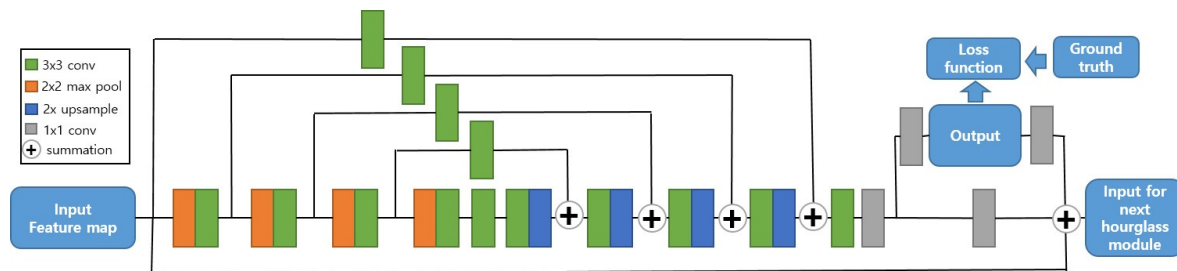


Figure 1. Structure of the hourglass module used in this paper. We follow the structure proposed in [17] except that the number of feature maps are set to 256 for all convolutional layers.

al. [34] used DNNs to learn an ideal binary mask which boils the source separation problem down to a binary classification problem. Simpson *et al.* [24] proposed a convolutional DNN to predict a probabilistic binary mask for singing voice separation. Recently, a fully complex-valued DNN [13] is proposed to integrate phase information into the magnitude spectrograms. Deep NMF [11] combined DNN and NMF by designing non-negative deep network and its back-propagation algorithm.

Since an audio signal is time series data, it is natural to use a sequence model like recurrent neural networks (RNNs) for music source separation tasks to learn temporal information. Huang *et al.* [6] proposed an RNN framework that jointly optimizes masks of foreground and background sources, which showed promising results for various source separation tasks. Other approaches include a recurrent encoder-decoder that exploits gated recurrent unit [15] or discriminative RNN [33].

CNNs are also an effective tool for audio signal analysis when the magnitude spectrogram is used as an input. Fully convolutional networks (FCNs) [14] are initially proposed for semantic segmentation in the computer vision area, which is also effective for solving human pose estimation [18, 35] or super-resolution [2]. FCNs usually contain downsampling and upsampling layers to learn meaningful features at multiple scales. Strided convolution or pooling is used for downsampling, while transposed convolution or nearest neighbor interpolation is mainly used for upsampling. It is proven that FCNs are also effective in signal processing. Chandna *et al.* [1] proposed encoder-decoder style FCN for monoaural audio source separation. Recently, singing voice separation using an U-Net architecture [8] showed impressive performance. U-Net [22] is a FCN which consists of a series of convolutional layers and upsampling layers. There is a skip connection which connects the convolutional layers of the same resolution. They trained vocal and accompaniment parts separately on different networks. Miron *et al.* [16] proposed the method that separates multiple sources using a single CNN. They used score-filtered spectrograms as inputs and generated masks for each source via an encoder-decoder CNN. Multi-resolution FCN [3] was proposed for monoaural audio source separation. Recently proposed CNN architecture [26] based on DenseNet [7] achieved state-of-the-art performance on DSD100 dataset.

3. METHOD

3.1 Network Architecture

The stacked hourglass network [18] was originally proposed to solve human pose estimation in RGB images. It is an FCN consisting of multiple hourglass modules. The hourglass module is similar to U-Net [22], of which feature maps at lower (coarse) resolution are obtained by repeatedly applying convolution and pooling operations. Then, the feature maps at the lowest resolution are upsampled via nearest neighbor interpolation with a preceding convolutional layer. Feature maps at the same resolution in the downsampling and the upsampling steps are connected with an additional convolutional layer. The hourglass module captures features at different scales by repeating pooling and upsampling with convolutional layers at each resolution. In addition, multiple hourglass modules are stacked to make the network deeper. As more hourglass modules are stacked, the network learns more powerful and informative features which refine the estimation results. Loss functions are applied at the end of each module. This intermediate supervision improves training speed and performance of the network.

The structure of a single hourglass module used in this paper is illustrated in Fig 1. Considering the efficiency and the size of the network, we adopt the hourglass module used in [17] which is a smaller network than the originally proposed one in [18]. A notable difference is that the residual blocks [4] used in [18] are replaced with a single convolutional layer. This light-weight structure showed competitive performance to the original network in human pose estimation with much smaller number of parameters. In the module, there are four downsampling and upsampling steps. All convolutional layers in downsampling and upsampling steps have filter size of 3×3 . The 2×2 max pooling is used to halve the size of the feature maps, and the nearest neighbor interpolation is used to double the size of the feature maps in the upsampling steps. We fixed the size of the maximum feature maps in convolutional layers to 256 which is different from [17]. After the last upsampling layer, a single 3×3 convolution and two 1×1 convolution is performed to generate network outputs. Then, an 1×1 convolution is applied to the outputs to match the number of channels to that of the input feature maps. Another 1×1 convolution is also applied to the feature maps

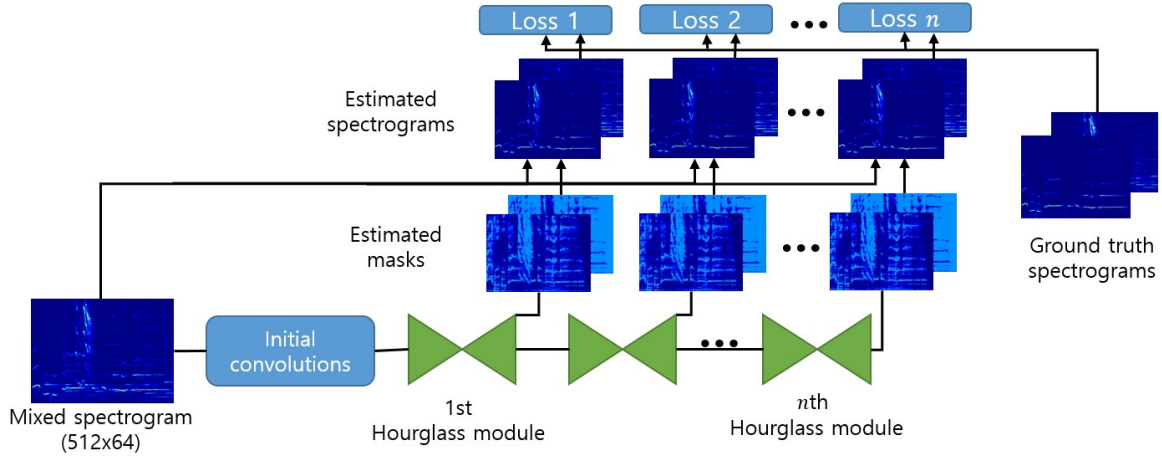


Figure 2. Overall music source separation framework proposed in this paper. Multiple hourglass modules are stacked, and each module outputs masks for each music source. The masks are multiplied with the input spectrogram to generate predicted spectrograms. Differences between the estimated spectrograms and the ground truth ones are used as loss functions of the network.

which used for output generation. Finally, the two feature maps that passed the respective 1×1 convolution and the input of the hourglass module is added together, and the resulting feature map is used as an input to the next hourglass module.

In the network used in this paper, input image firstly passes through initial convolutional layers that consist of a 7×7 convolutional layer and four 3×3 convolutional layers where the number of output feature maps for each layer is 64, 128, 128, 128, and 256 respectively. To make the output mask and the input spectrogram have the same size, we did not use the pooling operations in the initial convolutional layers before the hourglass module. The feature maps generated from the initial layers are fed to the first hourglass module. The proposed overall music source separation framework is depicted in Fig. 2.

3.2 Music Source Separation

As shown in Fig. 2, to apply the stacked hourglass network to music source separation, we aim to train the network to output soft masks for each music source given the magnitude spectrogram of the mixed source. Hence, the output dimension of the network is $H \times W \times C$ where H and W are the height and width of the input spectrogram respectively, and C is the number of music sources to separate. The magnitude spectrogram of separated music source is obtained by multiplying the mask and the input spectrogram. Our framework is scalable in that it requires almost no additional operation as the number of sources increases.

The input for the network is the magnitude of spectrogram obtained from Short-Time Fourier Transform (STFT) with a window size of 1024 and a hop size of 256. The input source is downsampled to 8kHz to increase the duration of spectrograms in a batch and to speed up training. For each sample, magnitude spectrograms of mixed and

separated sources are generated, which are divided by the maximum value of the mixed spectrogram for data normalization. The spectrograms have 512 frequency bins and the width of the spectrogram depends on the duration of the music sources. For all the music sources, the width of the spectrogram is at least 64. Thus, we fix the size of an input spectrogram to 512×64 . Hence, the size of the feature maps at the lowest resolution is 32×4 . Starting time index is randomly chosen when the input batches are created.

Following [22], we designed the loss function as an $L_{1,1}$ norm of the difference between the ground truth spectrogram and the estimated spectrogram. More concretely, given an input spectrogram \mathbf{X} , i th ground truth music source \mathbf{Y}_i , and the generated mask for the i th source in the j th hourglass module $\hat{\mathbf{M}}_{ij}$, the loss for the i th source is defined as

$$\mathcal{J}(i, j) = \|\mathbf{Y}_i - \mathbf{X} \odot \hat{\mathbf{M}}_{ij}\|_{1,1}, \quad (1)$$

where \odot denotes element-wise multiplication of the matrix. $L_{1,1}$ norm is calculated as the sum of absolute values of matrix elements. The loss function of the network becomes

$$\mathcal{J} = \sum_{i=1}^C \sum_{j=1}^D \mathcal{J}(i, j), \quad (2)$$

where D is the number of hourglass modules stacked in the network. We directly used the output of the last 1×1 convolutional layer as the mask, which is different from [22] where they used the sigmoid activation to generate masks. While it is natural to use the sigmoid function to restrict the value of the mask to $[0, 1]$, we empirically found that not applying the sigmoid function boosts the training speed and improves the performance. Since sigmoid activations vanish the gradient of the inputs that have large absolute values, they may diminish the effect of intermediate supervision.

We have stacked hourglass modules up to four and provide analysis of the effect of stacking multiple modules in Section 4. The network is trained using Adam optimizer [10] with a starting learning rate of 10^{-4} and a batch size of 4. We trained the network for 15,000 and 150,000 iterations for MIR-1K dataset and DSD100 dataset respectively, and the learning rate is decreased to 2×10^{-5} when 80% of the training is finished. No data augmentation is applied during training. The training took 3 hours for MIR-1K dataset and 31 hours for DSD100 dataset using a single GPU when the biggest model is used. For the singing voice separation task, C is set to 2 which corresponds to *vocal* and *accompaniments*. For the music source separation task in DSD100 dataset, $C = 4$ is used where each output mask corresponds to *drum*, *bass*, *vocal*, and *others*. While it can be advantageous in terms of performance to train a network for a single source individually, it is computationally expensive to train a deep CNN for each source. Therefore, we trained a single network for each task.

In the test phase, the magnitude spectrogram of the input source is cropped to network input size and fed to the network sequentially. The output of the last hourglass module is used for testing. We set the negative values of output masks to 0 in order to avoid negative magnitude values. The masks are multiplied by the normalized magnitude spectrogram of the test source and unnormalized to generate spectrograms of separated sources. We did not change the phase spectrogram of the input source, and it is combined with the estimated magnitude spectrogram to retrieve signals for separated sources via inverse STFT.

4. EXPERIMENTS

We evaluated performance of the proposed method on MIR-1K and DSD100 datasets. For quantitative evaluation, we measured signal-to-distortion ratio (SDR), source-to-interference ratio (SIR), and source-to-artifacts ratio (SAR) based on BSS-EVAL metrics [31]. Normalized SDR (NSDR) [20] is also measured for the singing voice separation task which measures improvement between the mixture and the separated source. The values are obtained using mir-eval toolbox [21]. Global NSDR (GNSDR), global SIR (GSIR), and global SAR (GSAR) are calculated as a weighted mean of NSDR, SIR, and SAR respectively whose weights are length of the source. The separated sources generated from the network are upsampled to the original sampling rate of the dataset and compared with ground truth sources for all experiments.

4.1 MIR-1K dataset

MIR-1K dataset is designed for singing voice separation research. It contains a thousand song clips extracted from 110 Chinese karaoke songs at a sampling rate of 16kHz. Following the previous works [6, 37], we used one male and one female (*abjones* and *amy*) as a training set which contains 175 clips in total. The remaining 825 clips are used for evaluation. For the baseline CNN, we trained the FCN that has U-Net [22]-like structure and evaluated its

Singing voice			
Method	GNSDR	GSIR	GSAR
MLRR [37]	3.85	5.63	10.70
DRNN [6]	7.45	13.08	9.68
ModGD [23]	7.50	13.73	9.45
U-Net [8]	7.43	11.79	10.42
SH-1stack	10.29	15.51	12.46
SH-2stack	10.45	15.89	12.49
SH-4stack	10.51	16.01	12.53

Accompaniments			
Method	GNSDR	GSIR	GSAR
MLRR [37]	4.19	7.80	8.22
U-Net [8]	7.45	11.43	10.41
SH-1stack	9.65	13.90	12.27
SH-2stack	9.64	13.69	12.39
SH-4stack	9.88	14.24	12.36

Table 1. Quantitative evaluation of singing voice separation on MIR-1K dataset.

performance. We followed the structure of [8], in which singing voice and accompaniments are trained on different networks. For the stacked hourglass networks, both singing voice and accompaniments are obtained from a single network.

The evaluation results on test sets are shown in Table 1. We trained the networks with varying number of stacked hourglass modules 1, 2, and 4. It is proven that our stacked hourglass network (SH) significantly outperforms existing methods in all evaluation criteria. Our method gains 3.01 dB in GNSDR, 2.28 dB in GSIR, and 1.83 dB in GSAR compared to the best results of the existing methods. It is also proven that the structure of the stacked hourglass module is more efficient and beneficial than U-Net [8] for music source separation. U-Net has 9.82 million parameters while single stack hourglass network has 8.99 million parameters considering only convolutional layers. Even with the absence of batch normalization, smaller number of parameters, and multi-source separation in a single network, the stacked hourglass network showed superior performance to U-Net. While the network with a single hourglass module shows outstanding source separation performance, even better results are provided when multiple hourglass modules are stacked. This indicates that SH network does not overfit even when the network gets deeper despite small amount of the training data. Our method provides good performance on separating both singing voice and accompaniments with a single forward step.

Qualitative results of our method and comparison with U-Net are shown in Fig. 3. The estimated log spectrograms of singing voice and accompaniments from SH-4stack and U-Net and the ground truth log spectrograms are provided. It can be seen that our method captures fine details and harmonics compared to the U-Net. The voice spectrogram from U-Net has more artifacts in the time slot of 0~1 and 4~5 compared to the result of SH-4stack. On the other

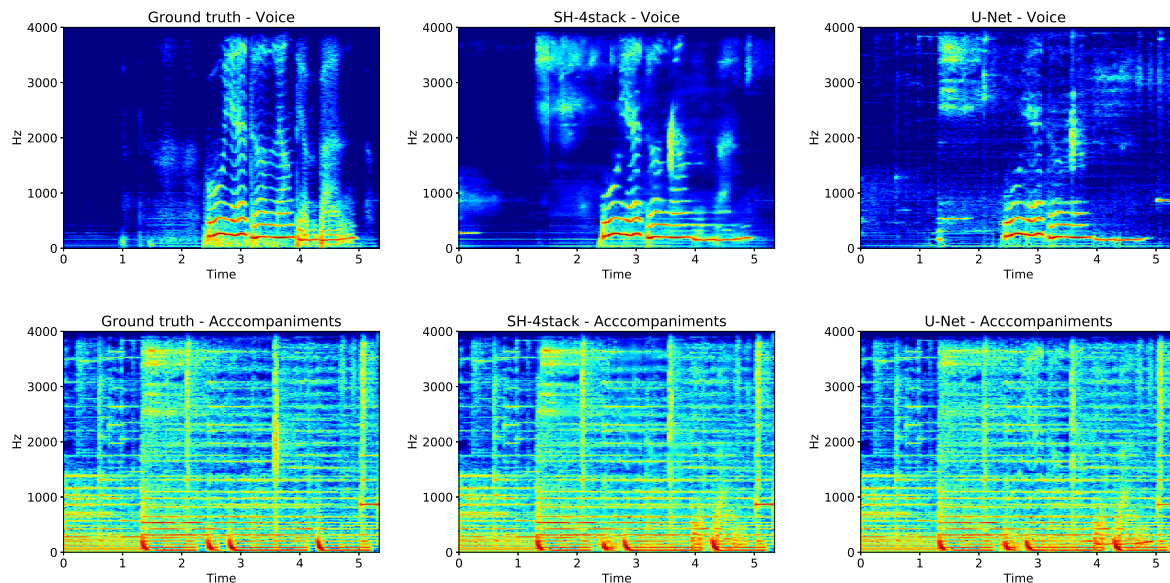


Figure 3. Qualitative comparison of our method (SH-4stack) and U-Net for singing voice and accompaniments separation on *annar_3_05* in MIR-1K dataset. Ground truth and estimated spectrograms are displayed in a log-scale. Our method is superior in capturing fine details compared to U-Net.

hand, harmonics from voice signals can be clearly seen in the spectrogram of SH-4stack. For accompaniments spectrogram, it is observed that U-Net contains voice signals around the time slot of 3.

4.2 DSD100 dataset

DSD100 dataset consists of 100 songs that are divided into 50 training sets and 50 test sets. For each song, four different music sources, bass, drums, vocals, and other as well as their mixtures are provided. The sources are stereophonic sound with a sampling rate of 44.1kHz. We converted all sources to monophonic and performed single channel source separation using stacked hourglass networks. We used a 4-stacked hourglass network (SH-4stack) for the experiments.

The performance of music source separation using stacked hourglass network is provided in Table 2. We measured SDR of the separated sources for all test songs and report median values for comparison with existing methods. The methods that use single channel inputs are compared to our method. While the stacked hourglass network gives second-best performance following the state-of-the-art methods [26] for drums and vocals, it shows poor performance for separating bass and other. This is mainly due to the similarity between bass and guitar sound in other sources, which confuses the network especially when trained together in a single network. Since the losses for all sources are summed up with equal weights, the network tends to be trained to improve the separation performance of vocal and drum, which is easier than separating bass and other sources.

Next, we trained the stacked hourglass network for a

Method	Bass	Drums	Other	Vocals
dNMF [36]	0.91	1.87	2.43	2.56
DeepNMF [11]	1.88	2.11	2.64	2.75
BLEND [28]	2.76	3.93	3.37	5.13
MM-DenseNet [26]	3.91	5.37	3.81	6.00
SH-4stack	1.77	4.11	2.36	5.16

Table 2. Median SDR values for music source separation on DSD100 dataset.

Method	Vocals	Accompaniments
DeepNMF [11]	2.75	8.90
wRPCA [9]	3.92	9.45
NUG [19]	4.55	10.29
BLEND [28]	5.23	11.70
MM-DenseNet [26]	6.00	12.10
SH-4stack	5.45	12.14

Table 3. Median SDR values for singing voice separation on DSD100 dataset.

singing voice separation task. The three sources except vocals are mixed together to form accompaniments source. The median SDR values for each source are reported in Table 3. Our method achieved best result for accompaniments separation and second-best for vocal separation. Separation performance of vocals is improved compared to the music source separation setting. It can be inferred that the stacked hourglass network provides better results as number of sources are smaller and the separating sources are more distinguishable from each other.

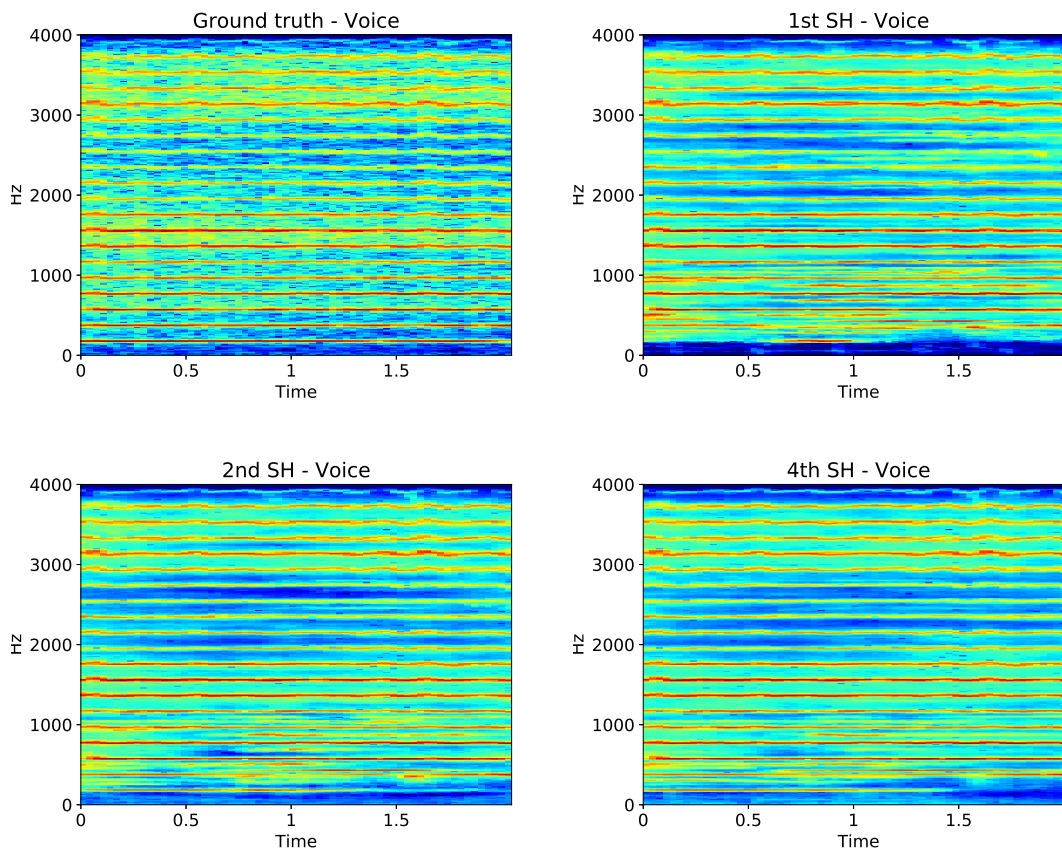


Figure 4. Examples showing the effectiveness of stacking multiple hourglass modules. Ground truth and estimated spectrograms of the part of the song *Schoolboy Fascination* in DSD100 dataset are shown. SDR values of the source generated from the spectrograms obtained from first, second, fourth hourglass module are 10.90, 12.50, 13.30 respectively. Especially, it is observed that the estimated spectrogram captures fine details of spectrogram at low frequency range (0~500 Hz) as more hourglass modules are stacked.

Lastly, we investigate how the stacked hourglass network improves the output masks as they pass through the hourglass modules within the network. The example illustrated in Fig. 4 shows the estimated voice spectrogram of first, second, and fourth hourglass module with the ground truth spectrogram from one of the test sets of DSD 100 dataset. It is observed that the estimated spectrogram becomes more similar to the ground truth as it is generated from a deeper side of the network. In the result of the fourth hourglass module, spectrograms at low frequency are clearly recovered compared to the result of the first hourglass module. The artifacts in the range of 2000~3000 Hz are also removed. Although it is hard to recognize the difference in the spectrogram image, the difference of SDR between the source estimated from the first hourglass module and the last hourglass module is about 2.4dB which is a significant performance gain.

5. CONCLUSION

In this paper, we proposed music source separation algorithm using stacked hourglass networks. The network suc-

cessfully captures features at both coarse and fine resolution, and it produces masks that are applied to the input spectrograms. Multiple hourglass modules refines the estimation results and outputs the better results. Experimental results has proven the effectiveness of the proposed framework for music source separation. We implemented the framework in its simplest form, and there is a lot of room for performance improvements including data augmentation, regularization of CNNs, and ensemble learning of multiple models. Designing a loss function that considers correlation of different sources may further improves the performance.

6. ACKNOWLEDGEMENT

This work was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (2017M3C4A7077582).

7. REFERENCES

- [1] Pritish Chandna, Marius Miron, Jordi Janer, and Emilia Gómez. Monoaural audio source separation using deep

- convolutional neural networks. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 258–266. Springer, 2017.
- [2] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.
- [3] Emad M Grais, Hagen Wierstorf, Dominic Ward, and Mark D Plumbley. Multi-resolution fully convolutional neural networks for monaural audio source separation. *arXiv preprint arXiv:1710.11473*, 2017.
- [4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [5] C. L. Hsu and J. S. R. Jang. On the improvement of singing voice separation for monaural recordings using the mir-1k dataset. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(2):310–319, Feb 2010.
- [6] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Joint optimization of masks and deep recurrent neural networks for monaural source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(12):2136–2147, 2015.
- [7] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014.
- [8] Andreas Jansson, Eric Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep u-net convolutional networks. *18th International Society for Music Information Retrieval Conference, Suzhou, China*, 2017.
- [9] Il-Young Jeong and Kyogu Lee. Singing voice separation using rpca with weighted $L_{\{1\}}$ -norm. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 553–562. Springer, 2017.
- [10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [11] Jonathan Le Roux, John R Hershey, and Felix Weninger. Deep nmf for speech separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 66–70. IEEE, 2015.
- [12] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.
- [13] Yuan-Shan Lee, Chien-Yao Wang, Shu-Fan Wang, Jia-Ching Wang, and Chung-Hsien Wu. Fully complex deep neural network for phase-incorporating monaural source separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 281–285. IEEE, 2017.
- [14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
- [15] Stylianos Ioannis Mimilakis, Konstantinos Drossos, Tuomas Virtanen, and Gerald Schuller. A recurrent encoder-decoder approach with skip-filtering connections for monaural singing voice separation. *CoRR*, abs/1709.00611, 2017.
- [16] Marius Miron, Jordi Janer, and Emilia Gómez. Monaural score-informed source separation for classical music using convolutional neural networks. In *18th International Society for Music Information Retrieval Conference, Suzhou, China*, 2017.
- [17] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2274–2284, 2017.
- [18] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.
- [19] Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent. Multichannel music separation with deep neural networks. In *Signal Processing Conference (EUSIPCO), 2016 24th European*, pages 1748–1752. IEEE, 2016.
- [20] Alexey Ozerov, Pierrick Philippe, Frédéric Bimbot, and Rmi Gribonval. Adaptation of bayesian models for single-channel source separation and its application to voice/music separation in popular songs. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(5):1564–1578, 2007.
- [21] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. mir_eval: A transparent implementation of common mir metrics. In *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.
- [22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

- [23] Jilt Sebastian and Hema A Murthy. Group delay based music source separation using deep recurrent neural networks. In *Signal Processing and Communications (SPCOM), 2016 International Conference on*, pages 1–5. IEEE, 2016.
- [24] Andrew JR Simpson, Gerard Roma, and Mark D Plumbley. Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 429–436. Springer, 2015.
- [25] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.
- [26] Naoya Takahashi and Yuki Mitsufuji. Multi-scale multi-band densenets for audio source separation. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2017 IEEE Workshop on*, pages 21–25. IEEE, 2017.
- [27] Stefan Uhlich, Franck Giron, and Yuki Mitsufuji. Deep neural network based instrument extraction from music. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 2135–2139. IEEE, 2015.
- [28] Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 261–265. IEEE, 2017.
- [29] Shankar Vembu and Stephan Baumann. Separation of vocals from polyphonic audio recordings. In *ISMIR*, pages 337–344. Citeseer, 2005.
- [30] Emmanuel Vincent, Shoko Araki, Fabian Theis, Guido Nolte, Pau Bofill, Hiroshi Sawada, Alexey Ozerov, Vikram Gowreesunker, Dominik Lutter, and Ngoc QK Duong. The signal separation evaluation campaign (2007–2010): Achievements and remaining challenges. *Signal Processing*, 92(8):1928–1936, 2012.
- [31] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing*, 14(4):1462–1469, 2006.
- [32] Tuomas Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE transactions on audio, speech, and language processing*, 15(3):1066–1074, 2007.
- [33] Guan-Xiang Wang, Chung-Chien Hsu, and Jen-Tzung Chien. Discriminative deep recurrent neural networks for monaural speech separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 2544–2548. IEEE, 2016.
- [34] Yuxuan Wang, Arun Narayanan, and DeLiang Wang. On training targets for supervised speech separation. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(12):1849–1858, 2014.
- [35] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.
- [36] Felix Weninger, Jonathan Le Roux, John R Hershey, and Shinji Watanabe. Discriminative nmf and its application to single-channel source separation. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.
- [37] Yi-Hsuan Yang. Low-rank representation of both singing voice and music accompaniment via learned dictionaries. In *ISMIR*, pages 427–432, 2013.
- [38] Xiu Zhang, Wei Li, and Bilei Zhu. Latent time-frequency component analysis: A novel pitch-based approach for singing voice separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 131–135. IEEE, 2015.

THE NORTHWESTERN UNIVERSITY SOURCE SEPARATION LIBRARY

Ethan Manilow, Prem Seetharaman, Bryan Pardo

Northwestern University

{ethanmanilow@u., prem@u., pardo@}northwestern.edu

ABSTRACT

Audio source separation is the process of isolating individual sonic elements from a mixture or auditory scene. We present the Northwestern University Source Separation Library, or `nussl` for short. `nussl` (pronounced ‘nuzzle’) is an open-source, object-oriented audio source separation library implemented in Python. `nussl` provides implementations for many existing source separation algorithms and a platform for creating the next generation of source separation algorithms. By nature of its design, `nussl` easily allows new algorithms to be benchmarked against existing algorithms on established data sets and facilitates development of new variations on algorithms. Here, we present the design methodologies in `nussl`, two experiments using it, and use `nussl` to showcase benchmarks for some algorithms contained within.

1. INTRODUCTION

Audio source separation is the process of isolating individual sonic elements from a mixture or auditory scene. The underdetermined case is where there are fewer mixture channels (e.g. a stereo recording) than sources (a string quartet). Examples of underdetermined source separation include extracting a single speaker from a single-mic recording of a crowded cocktail party, extracting a singer from a rock band recording, or removing an extraneous car horn from a field recorded interview. Applications of source separation include end-user tools for extracting vocals (e.g., Audionamix ADX Trax), upmixing vintage recordings to stereo or 5.1 surround sound, and as a pre-processing step for speech recognition [15] and other audio tasks.

There have been many approaches taken to source separation in the underdetermined case. These include Non-negative Matrix Factorization (NMF) [37, 38], harmonic/percussive separation [7], deep learning [11, 13, 14, 16, 21, 25], pitch tracking [5, 34], spatialization [8, 32], repeating vs non-repeating elements [29, 30, 36], low-rank vs sparse decomposition [12], and common fate [24, 39, 44].

The research community has centered around a collection of common data sets to benchmark results from these different approaches. Perhaps the best known are the data sets used for the recurring Signal Separation Evaluation Campaign (SiSEC) [19, 42]. SiSEC previously used DSD100 [20], and now uses both DSD100 and MedleyDB [1], calling the combined data set MUSDB18 [28]. Other common data sets include iKala [2], MIR-1K [3], TIMIT [10], and WSJ0 [9]. The community also typically uses the signal quality measures provided by BSS-Eval [6, 42] (SDR, SIR, and SAR) when reporting results.

Though there is some debate about this [4], it can be argued that using common data sets and evaluation measures strengthens research through standardizing metrics by making new and existing research directly comparable. While the source separation community has common data sets and common evaluation measures, there exists no such common code repository for actual implementations of proposed algorithms.

Vandewalle *et al.* [41] argue that in the computational sciences, implementation details are crucial to reproducing the results of academic papers, despite being routinely omitted from publications. They establish six degrees of reproducibility, scored from 0 (lowest) to 5 (highest). A score of 5 is defined as “The results can be easily reproduced by an independent researcher with at most 15 min of user effort, requiring only standard, freely available tools.” A 0 indicates research completely unreproducible by an independent researcher.

The ubiquity of code repositories like Github has allowed many researchers to share their code, but using Github is not a guarantee of easy reproducibility. A recent seminar¹ convened to reproduce results from six MIR papers (including two source separation papers) and concluded that not a single paper, *despite including code*, scored better than “Can be reproduced, requiring considerable effort” using the reproducibility scorecard by Vandewalle *et al.* [41]. Of the two source separation papers, both scored “Could be reproduced, requiring extreme effort.”

This work aims to provide a common platform for researchers to contribute their source separation algorithms to fill the implementation gap and promote reproducibility within the source separation research community. Furthermore, this work strives to make every algorithm in the proposed framework achieve the highest reproducibility rating using the Vandewalle *et al.* scorecard: reproducible results



© Ethan Manilow, Prem Seetharaman, Bryan Pardo. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Ethan Manilow, Prem Seetharaman, Bryan Pardo. “The Northwestern University Source Separation Library”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ <https://github.com/audiolabs/APSRR-2016>

in under 15 minutes. The Northwestern University Source Separation Library (`nussl`) is the culmination of that effort.

In this paper we will explore `nussl` and introduce some core aspects of its design methodology, provide an outline about how to add a new algorithm to `nussl` and benchmark many of the source separation algorithms in `nussl`. We also leverage the flexibility of the `nussl` framework to implement and test novel combinations of existing source separation algorithms. We examine how source separation algorithms interact with methods such as overlap and add, which apply the same source separation algorithm to overlapping windows in the mixture and recombine the sources afterwards, rather than applying them to the entire mixture. More information about `nussl` can be obtained at the project’s online documentation.² A companion website is also provided for this paper.³

2. RELATED WORK

The biennial Signal Separation Evaluation Campaign (SiSEC) [19,42] is an open call for members of the MIR research community to submit source separation algorithms to be run and evaluated on a common dataset. While the dataset is widely distributed and used, not all of the code submissions from previous campaigns have been made available to scrutinize. Additionally, SiSEC offers no standard API to adhere to, and only a minimal framework to work with. We have submitted many algorithms within `nussl` to the most recent SiSEC campaign.

Other source separation libraries have been presented in the past, as well. The *Flexible Audio Source Separation Toolbox* (FASST) [23,35]⁴ was written in MATLAB and C++, but did not have a process for outside submissions. `untwist` [33] is an open source Python source separation library, but it is based on a different design framework than `nussl`, implements a different set of algorithms than `nussl`, and has no built-in interfaces for common evaluation metrics, data sets, or loading pre-trained models.

3. DESIGN FRAMEWORK OF NUSSL

`nussl` is built with extensibility in mind. It would be impossible to provide implementations for every source separation algorithm upon the announcement of this library. As such `nussl` is built to an API so that the community can easily add their own algorithms, models, datasets and have them automatically work with every other aspect of `nussl`.

Under the hood, `nussl` uses many common Python tools for signal processing and machine learning, such as `librosa`, `numpy`, `scipy`, `scikit-learn`, `mir_eval`, `musdb`, and `museval`, so developing with `nussl` should be familiar to any MIR researcher working in Python.

```
1 import nussl
2
3 # Load audio
4 signal = nussl.AudioSignal('path/to/mix.wav')
5
6 # Run REPET for foreground/background separation
7 algorithm = nussl.Repet(sig)
8 algorithm.run()
9 fg, bg = algorithm.make_audio_signals()
10
11 # Save results to wav files
12 fg.write_audio_to_file('fg.wav')
13 bg.write_audio_to_file('bg.wav')
```

Figure 1: Using `nussl` to run a single algorithm (REPET [30] for foreground/background separation) on a single mixture. In a recent seminar on reproducibility, REPET scored “could be reproduced, requiring extreme effort.” `nussl` aims to improve the reproducibility score of multiple source separation algorithms, including REPET.

In the next sections, we provide a high-level overview of some of the more important aspects of the `nussl` API. For more information, please see our full online documentation.

3.1 AudioSignal

The main entry point to `nussl` for end-users and algorithm developers is through the `AudioSignal` object. The `AudioSignal` object has methods for reading and writing audio, padding or truncating the audio, adding and subtracting audio signals from one another, checking and altering properties of the audio, computing invertible signal transforms (e.g. short time Fourier transform), and much more. `AudioSignal` can read all of the most common audio codecs. Once in memory, audio is represented as a 2-dimensional (channels and time series within a channel) `numpy` array of pulse-code modulated (PCM) samples.

All source separation algorithms in `nussl` accept as their first argument an `AudioSignal` object. Each algorithm copies the content of the audio object, performs separation on that copy and returns a set of new `AudioSignal` objects, one per source, leaving the original `AudioSignal` object unchanged.

3.2 Source Separation Algorithms

All source separation algorithms in `nussl` are encapsulated in classes that are derived from `SeparationBase`. For each class, the constructor does minimal set up, the `run()` method does the computation required for the source separation, and the `make_audio_signals()` method returns `AudioSignal` objects containing the estimated signals. An example of this whole process is shown in Figure 1.

3.2.1 MaskSeparationBase vs SeparationBase

Source separation algorithms in `nussl` are segregated into two categories: those that produce a mask and apply it

² <https://interactiveaudiolab.github.io/nussl>

³ <https://interactiveaudiolab.github.io/demos/nussl.html>

⁴ Related Python library: <https://github.com/wslight/pyfasst/>

Algorithms in nussl			
Repetition	Other Fore/Background	Spatialization	Composite
Repet [31]	Harmonic/Percussive (HPSS) [7]	DUET [32]	Overlap/Add
RepetSim [29]	Melody Masking (Melodia [34])	PROJET [8]	Algorithm Picker [22]
2DFT [36]	Component Analysis	Benchmarking	Neural Networks
Matrix Decomposition	ICA [18]	High/Low Pass Filter	Deep Clustering [11, 21]
NMF w/ MFCC Clustering [38]	RPCA [12]	Ideal Mask	

Table 1: Source Separation algorithms by category currently implemented in nussl.

to a representation (e.g. a spectrogram) built from the waveform, and those that do separation via other means (e.g. time domain methods such as independent component analysis). The former group of algorithms are derived from the `MaskSeparationBase` base class, which is a subclass of the `SeparationBase` base class. The `run()` method in `MaskSeparationBase`-derived algorithms are expected to return mask objects (see Section 3.2.2). Some algorithms inherit directly from `SeparationBase` and have no requirement about what their `run()` method returns. With `MaskSeparationBase` separation classes, it is easy to switch between running an algorithm with a binary or soft mask.

3.2.2 Masks

Masks are encapsulated by the `MaskBase` base class. `SoftMask` and `BinaryMask` are the two classes that derive from `MaskBase`. `MaskBase`-derived objects have a `numpy` array that contains the data, and utilities for applying masks to `AudioSignal` objects. `SoftMask` objects are applied using a classical approach:

$$\hat{S}_{\omega,t}^{(i)} = \frac{v_{\omega,t}^{(i)}}{\sum_{i=0}^N v_{\omega,t}^{(i)}}$$

Here, $v_{\omega,t}^{(i)}$ is the estimate of source i at frequency ω and time t , $\hat{S}_{\omega,t}^{(i)}$ is the value of the mask for that source at that time and frequency, and N is the total number of sources. The `BinaryMask` objects simply put a 1 when a source estimate dominates all other source estimates in a time-frequency bin and a 0 elsewhere. More masking types (e.g. consistent Wiener filtering [17]) can be implemented by subclassing `MaskBase`.

3.3 Evaluation

nussl also has a common interface to evaluate the estimates from source separation algorithms using established metrics, such as BSS-Eval [6] using implementations from `mir_eval` [26] or `museval` [40]. nussl also has methods for comparing binary masks to an ideal binary mask using accuracy, precision, recall, and F-Score [43]. Similar to the rest of nussl, all of the evaluation metrics are encapsulated by the `EvaluationBase` base class so that all of its child classes are built to a common API.

```

1 import nussl
2
3 mlk = 'path/to/MIR-1K'
4
5 # List of algorithms to test
6 sep_classes = [nussl.RepetSim, nussl.Melodia]
7
8 # Loop through all of MIR-1K
9 for mix, vox, acc in nussl.datasets.mir1k(mlk):
10     mix.to_mono(overwrite=True)
11
12     for alg in sep_classes:
13
14         # Run the algorithm
15         a = alg(mix)
16         a.run()
17         est = a.make_audio_signals()
18
19         # Evaluate results
20         gt = [acc, vox] # Ground truth
21         bss = nussl.evaluate.BssEval(mix, gt, est)
22         scores = bss.evaluate()
    
```

Figure 2: Running two algorithms on *all* of MIR-1K and evaluating using BSS-Eval.

3.4 Data Sets

Although nussl does not ship with any data sets, it does provide “hooks” for interfacing with common data sets. The hooks are basic utilities for reading the audio files into `AudioSignal` objects. The user first points nussl to the top-level directory of the downloaded data set. The utilities can then iterate through every audio file, a subset of files, or shuffle the order in which they are read. There structure of the directories is assumed to be Data sets that nussl can currently interface with include iKala [2], MIR-1K [3], MUSDB18 [28] (using `musdb`), and DSD100 [20]. An example of running multiple algorithms on the entirety of MIR-1K and evaluating the results using BSS-Eval is shown in Figure 2.

3.5 Modelers and Deep Learning Models

nussl also contains a section for generic modeling and matrix manipulation classes. Classes in this section are not source separation algorithms, but are used by the algorithms in nussl. An example is the Non-negative Matrix Factorization (NMF) class, `NMF`. This receives a non-negative `numpy` matrix as input, factorizes it into a template matrix and an activation matrix, and outputs the two results to be used by a separation algorithm. It does not input or output audio, spectrograms, or masks. For those util-

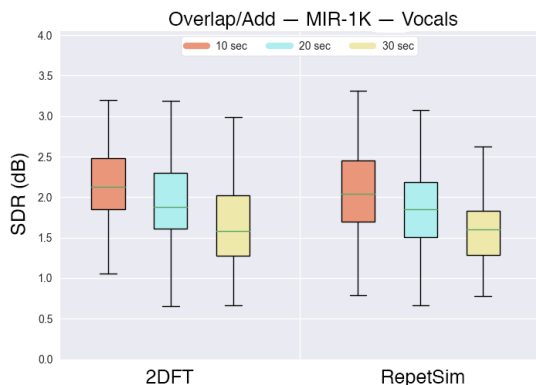


Figure 3: SDR evaluations for vocal estimates using OverlapApp on the MIR-1K data set. Three different window sizes are shown: 10 sec (red), 20 sec (blue), 30 sec (yellow). Hops are half of the window size.

ities, a wrapper separation class is needed, like `NMF_MFCC`, which clusters the templates using mel-frequency cepstral coefficients. Other classes train deep learning models for separation use. Classes in this section have a more lax API because of their heterogeneous nature.

`nussl` currently supports deep learning models written in `PyTorch`, but does not ship with any pre-trained models, only the code to train with. Similar to frameworks like `PyTorch`, `nussl` offers a way to download pre-trained models from `nussl` servers for algorithms which require them. Developers can see what models exist on our servers and download a models via utilities built into `nussl`. There also exists a process for contributors to upload their own pre-trained models (see Section 4.2 for more details).

4. ALGORITHMS IN NUSSL

4.1 Currently in `nussl`

At the time of this writing, the source separation algorithms are implemented in `nussl` to the API specification are presented in Table 1, by category. The algorithms currently in `nussl` provide a good starting point for future benchmark work, and we hope to expand the set of offered algorithms to include many more state-of-the-art approaches.

4.2 Adding new algorithms to `nussl`

The process of adding new source separation algorithms into `nusnussl` is similar to other open source projects, in many ways. A researcher who wishes to add an algorithm must clone the Github repository, make a new branch for their algorithm, add their code, push to Github, and then create a pull request. At this point, the `nussl` contributing process deviates from that standard open-source process.

After the new code passes the style and error checks, the researcher must provide benchmark files for tests. These

can be created by using standard metrics on a set of example files. For example, when adding a new algorithm, a researcher could provide `BSS-Eval` metrics on a few songs from MIR-1K dataset. If an implementation existed elsewhere prior to being incorporated into `nussl`, then a copy of the original implementation will be requested to benchmark against. Authors of new algorithms, must also provide a reference to a paper or other documentation which outlines the algorithm in more detail. Additionally, any large supplemental materials that are needed for the algorithm (such as pre-trained neural network models) must be provided so that they can be distributed through `nussl`'s API as outlined in Section 3.5.

All of this is outlined in more detail on the contributions section of the `nussl` Github page and documentation.

5. EXAMPLE USES OF NUSSL

Because all of the algorithms and supporting infrastructure in `nussl` are built to an API, this allows a very simple way to find novel combinations of multiple source separation algorithms and evaluate them on a variety of data sets under different evaluation metrics. In this section, we will showcase two novel experiments using `nussl` and present results from these experiments.

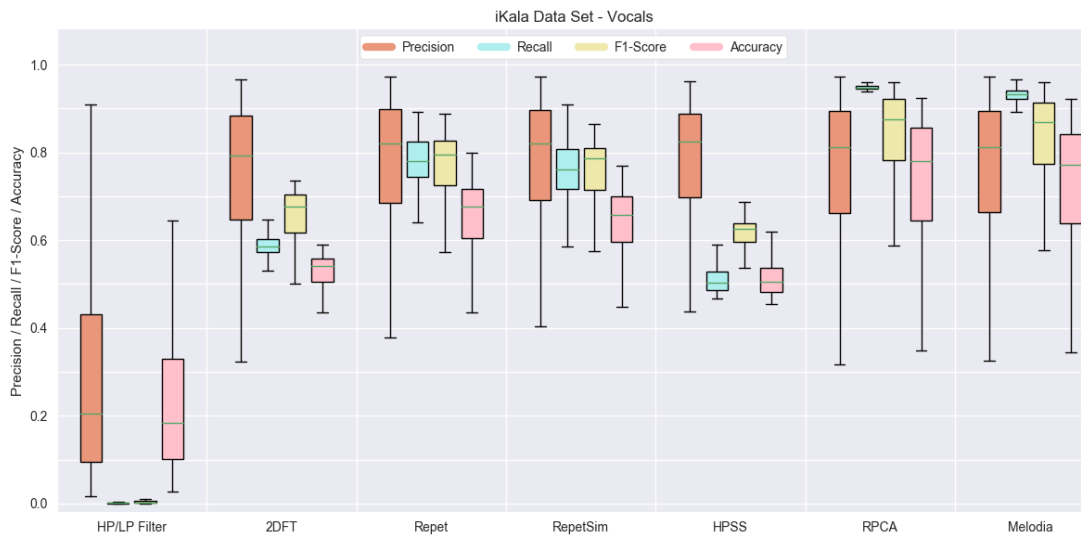
5.1 Cascading algorithms

The `nussl` API facilitates combining several different algorithms. To illustrate this point, we reproduce and expand upon work demonstrated by Rafii *et al.* [27] in combining rhythm-based and pitch-based approaches to source separation.

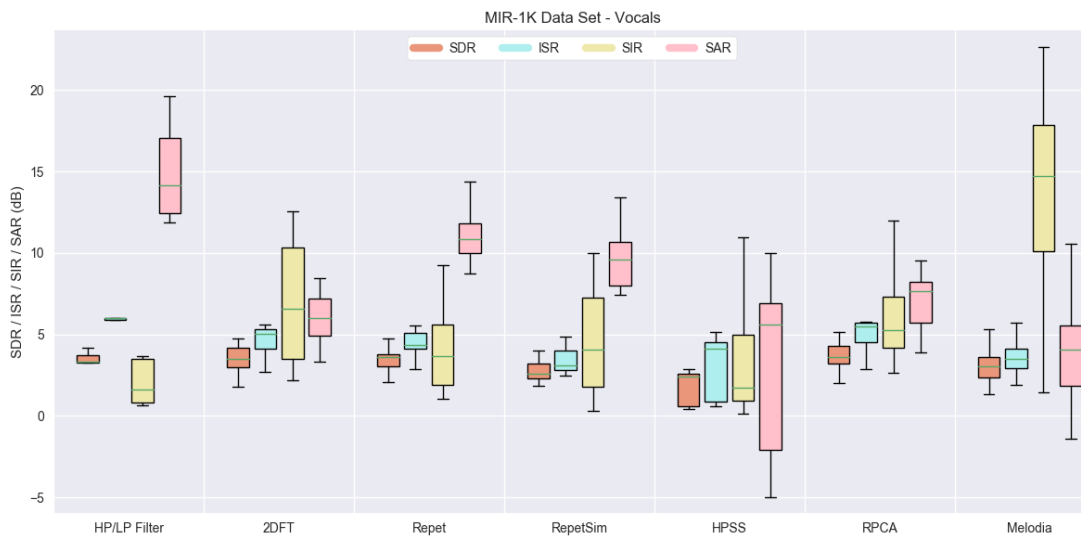
Rafii *et al.* present two methods for cascading algorithms: Parallel, where the background and foreground masks created by each algorithm are combined after the algorithms run on the mixture; and, Series, where the foreground estimation of algorithm A is fed in as the “mixture” to algorithm B. The mask estimates, in each case, are combined using weighted Weiner Filtering.

For this experiment, we use four background/foreground algorithms, RepetSim, Separation via 2DFT, RCPA, and Melodic masking with Melodia. We chose each pair from the set of algorithms and resulting in a total of 16 combinations. Based on values reported by Rafii *et al.*, for running in Parallel we set $w_B = 1.0$ and $w_M = 0.3$ as the weights of the background and foreground masks, respectively. We set the weight parameter $w = 0.5$ for running in Series. All algorithms created soft masks. We evaluated results using `BSS-Eval` on the undivided MIR-1K data set.⁵ Mean SDR values (with 1 standard deviation) are shown in Figure 5 for vocals. We find that series configurations outperform parallel configurations overall, and RepetSim is best as a second algorithm run especially when it is also the first.

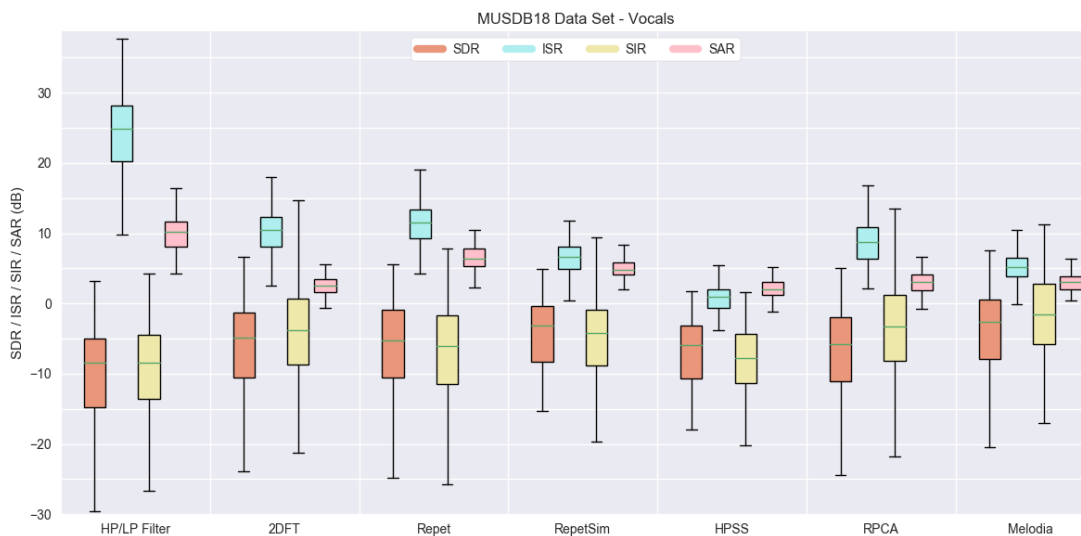
⁵ MIR-1K has 110 tracks of mean duration 72.7 ± 17.3 seconds, that are divided into 1000 smaller tracks of 8.0 ± 1.8 seconds. The divided tracks are too small to capture multiple repetitions.



(a) Benchmarks for the iKala data set. Algorithms apply binary masks to the mixtures and the results were evaluated using precision, recall, F-Score, and accuracy (precision is red, recall is blue, F-Score is yellow, and accuracy is pink).



(b) Benchmarks for the MIR-1K data set. Algorithms apply binary masks to the mixtures and the results were evaluated using BSS-Eval (SDR is red, ISR is blue, SIR is yellow, and SAR is pink).



(c) Benchmarks for the MUSDB18 data set. Algorithms apply soft masks to the mixtures and the results were evaluated using BSS-Eval (SDR is red, ISR is blue, SIR is yellow, and SAR is pink).

Figure 4: Illustrative benchmarks for a set of algorithms and configurations in `nussl`.

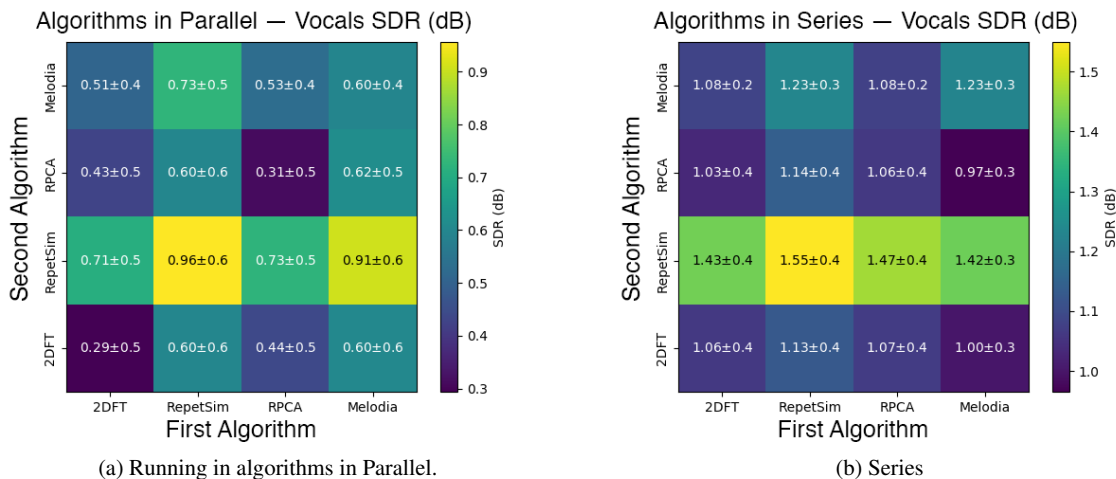


Figure 5: Mean SDR for vocal estimations from cascading pairs of algorithms.

5.2 Combining algorithms with Overlap/Add

In addition to classes that run one type of algorithm (like HPSS, etc), `nussl` also contains composite algorithm classes, *i.e.* those that run other algorithms in `nussl`. One such simple example is the `OverlapAdd` class, which does the overlap add method when running an algorithm.

In this experiment, we ran two repetition-based algorithms, `RepetSim` and `Separation via 2DFT`, wrapped in the `Overlap/Add` class to do vocal extraction. We tested three different window lengths, 10, 20, and 30 seconds, with hop length at half of the window length and using Hamming windows. We ran this experiment on the undivided MIR-1K data set⁶ and evaluated the estimates using BSS-Eval. Results from this experiment show that smaller windows lead to better vocal separation performance, according to SDR. These results are shown in Figure 3.

6. BENCHMARKS

In this section, we provide a selection of benchmarks for a set of algorithms in `nussl`. We benchmarked all algorithms that explicitly perform vocal separation with deterministic output source ordering (*i.e.* for an output array of sources, accompaniment is always index 0 and vocals is always index 1). We ran the algorithms on the iKala, MIR-1K, and MUSDB18 data sets. We ran REPET, REPET-SIM, Separation by 2DFT, HPSS, Masking from Pitch Tracking (using Melodia as the pitch tracker), RPCA, High/Low Pass filtering (cutoff at 100Hz).

For brevity, we only report one evaluation type for each data set here. We aim not to be complete, but rather showcase what `nussl` is capable of. For iKala, we show precision/recall/F-Score/accuracy computed from output binary masks, Figure 4a. For MIR-1K, we show BSS-Eval metrics computed from estimates using binary masks, Figure 4b. And for MUSDB18, we show BSS-Eval metrics computed from estimates using soft masks, Figure 4c.

⁶ We excluded two signals that were shorter than the largest window sizes.

All algorithms were run using the default parameter values for the algorithm in `nussl`'s implementation. Specifics of all of the parameters are contained in the project's documentation website.

7. FUTURE WORK AND CONCLUSION

In the future, we hope to expand upon `nussl` in a number of ways. First, while `nussl` is currently focused on musical source separation (the expertise of its authors), we would like to expand it to include source separation methods for speech. This would also necessitate adding hooks for speech data sets (like TIMIT and WSJ0) and adding pre-trained models for speech. Second, we would like to add an extensible API for spectral transformations. Currently, the STFT is at the core of `AudioSignal`, but in the future, it should be abstracted so that it is easy to run any algorithm on a CQT, Mel-Spaced STFT, etc.

Finally, and importantly, we would like buy-in from the MIR and audio community. The aim of `nussl` is to become the community's central repository for audio source separation. This goal is impossible without the support and contributions of the research community. We encourage interested participants to read the guidelines for contributing on this project's documentation page and get involved.

We have presented the Northwestern University Source Separation Library (`nussl`), an open-source, object-oriented audio source separation library implemented in Python. `nussl` implements many popular source separation algorithms, and a low barrier API for end-users and developers alike. We have demonstrated its design framework, including its ability to interface with common data sets and evaluation metrics. We also showcased two novel experiments using the API and a set of benchmarks. This project is actively seeking submissions from eager researchers and avid open source developers. Readers can find more information at interactiveaudiolab.github.io/nussl. This work was supported by USA National Science Foundation Award 1420971.

8. REFERENCES

- [1] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research. *15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, pages 155–160, 2014.
- [2] Tak-Shing Chan, Tzu-Chun Yeh, Zhe-Cheng Fan, Hung-Wei Chen, Li Su, Yi-Hsuan Yang, and Roger Jang. Vocal Activity Informed Singing Voice Separation with the iKala Dataset. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 718–722, April 2015.
- [3] Chao-Ling Hsu and J.-S.R. Jang. On the Improvement of Singing Voice Separation for Monaural Recordings Using the MIR-1K Dataset. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(2):310–319, February 2010.
- [4] Chris Drummond and Nathalie Japkowicz. Warning: Statistical Benchmarking is Addictive. Kicking the Habit in Machine Learning. *Journal of Experimental & Theoretical Artificial Intelligence*, 22(1):67–80, March 2010.
- [5] Jean-Louis Durrieu, Bertrand David, and Gaël Richard. A Musically Motivated Mid-Level Representation for Pitch Estimation and Musical Audio Source Separation. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1180–1191, October 2011.
- [6] Cédric Févotte, Rémi Gribonval, and Emmanuel Vincent. Bss_eval toolbox user guide—revision 2.0. page 19, 2005.
- [7] Derry FitzGerald. Harmonic/Percussive Separation Using Median Filtering. *Proceedings of the 13th International Conference on Digital Audio Effects*, 2010.
- [8] Derry FitzGerald, Antoine Liutkus, and Roland Badeau. PROJET — Spatial Audio Separation Using Projections. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 36–40, March 2016.
- [9] John Garofalo, David Graff, Doug Paul, and David Pallett. Continuous speech recognition (csr-i) wall street journal (wsj0) news, complete. *Linguistic Data Consortium, Philadelphia*, 1993.
- [10] John S Garofalo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continuous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.
- [11] John R. Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. Deep Clustering: Discriminative Embeddings for Segmentation and Separation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 31–35. IEEE, March 2016.
- [12] Po-Sen Huang, Scott Deeann Chen, Paris Smaragdis, and Mark Hasegawa-Johnson. Singing-Voice Separation from Monaural Recordings Using Robust Principal Component Analysis. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 57–60. IEEE, March 2012.
- [13] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Deep Learning for Monaural Speech Separation. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1562–1566. IEEE, May 2014.
- [14] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Singing-Voice Separation from Monaural Recordings using Deep Recurrent Neural Networks. *15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, pages 477–482, 2014.
- [15] Yusuf Isik, Jonathan Le Roux, Zhuo Chen, Shinji Watanabe, and John R Hershey. Single-Channel Multi-Speaker Separation Using Deep Clustering. *arXiv preprint arXiv:1607.02173*, 2016.
- [16] Minje Kim and Paris Smaragdis. Bitwise Neural Networks. In *International Conference on Machine Learning (ICML) Workshop on Resource-Efficient Machine Learning*, Lille, France, Jul 2015.
- [17] Jonathan Le Roux and Emmanuel Vincent. Consistent Wiener Filtering for Audio Source Separation. *IEEE Signal Processing Letters*, 20(3):217–220, March 2013.
- [18] Te-Won Lee. *Independent Component Analysis: Theory and Applications*. Springer, New York; London, 2011. OCLC: 752483521.
- [19] Antoine Liutkus, Fabian-Robert Stöter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave. The 2016 Signal Separation Evaluation Campaign. In *Latent Variable Analysis and Signal Separation - 13th International Conference, LVA/ICA 2017, Grenoble, France, February 21-23, 2017, Proceedings*, pages 323–332, 2017.
- [20] Antoine Liutkus, Fabian-Robert Stöter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave. The 2016 Signal Separation Evaluation Campaign. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 323–332. Springer, 2017.
- [21] Yi Luo, Zhuo Chen, John R. Hershey, Jonathan Le Roux, and Nima Mesgarani. Deep Clustering and Conventional Networks for Music Separation:

- Stronger Together. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 61–65. IEEE, March 2017.
- [22] Ethan Manilow, Prem Seetharaman, Fatemeh Pishdadian, and Bryan Pardo. Predicting Algorithm Efficacy for Adaptive Multi-Cue Source Separation. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 274–278, Oct 2017.
- [23] Alexey Ozerov and Emmanuel Vincent. Using the FASST source separation toolbox for noise robust speech recognition. In *International Workshop on Machine Listening in Multisource Environments (CHiME 2011)*, Florence, Italy, Sept 2011.
- [24] Fatemeh Pishdadian, Bryan Pardo, and Antoine Liutkus. A Multi-resolution Approach to Common Fate-based Audio Separation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 566–570. IEEE, March 2017.
- [25] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Joint Optimization of Masks and Deep Recurrent Neural Networks for Monaural Source Separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(12):2136–2147, December 2015.
- [26] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. MIR.EVAL: A Transparent Implementation of Common MIR Metrics. In *15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, pages 367–372, 2014.
- [27] Zafar Rafii, Zhiyao Duan, and Bryan Pardo. Combining Rhythm-Based and Pitch-Based Methods for Background and Melody Separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12):1884–1893, December 2014.
- [28] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation, December 2017.
- [29] Zafar Rafii and Bryan Pardo. Music/Voice Separation Using the Similarity Matrix. *13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, pages 583–588, 2012.
- [30] Zafar Rafii and Bryan Pardo. REpeating Pattern Extraction Technique (REPET): A Simple Method for Music/Voice Separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(1):73–84, Jan 2013.
- [31] Zafar Rafii and Bryan Pardo. REpeating Pattern Extraction Technique (REPET): A Simple Method for Music/Voice Separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(1):73–84, January 2013.
- [32] Scott Rickard. The DUET blind source separation algorithm. In *Blind Speech Separation*, pages 217–241. Springer, 2007.
- [33] Gerard Roma, Emad M Grais, Andrew JR Simpson, Iwona Sobieraj, and Mark D Plumbley. Untwist: A New Toolbox for Audio Source Separation. *Extended abstracts for the Late-Breaking Demo Session of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, 2016.
- [34] Justin Salamon and Emilia Gomez. Melody Extraction From Polyphonic Music Signals Using Pitch Contour Characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, August 2012.
- [35] Yann Salaün, Emmanuel Vincent, Nancy Bertin, Nathan Souvira-Labastie, Xabier Jaureguiberry, Dung T Tran, and Frédéric Bimbot. The Flexible Audio Source Separation Toolbox Version 2.0. In *ICASSP*, 2014.
- [36] Prem Seetharaman, Fatemeh Pishdadian, and Bryan Pardo. Music/Voice Separation Using the 2D Fourier Transform. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 36–40, Oct 2017.
- [37] Paris Smaragdis and J.C. Brown. Non-Negative Matrix Factorization for Polyphonic Music Transcription. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684)*, pages 177–180. IEEE, 2003.
- [38] Martin Spiertz and Volker Gnann. Source-Filter Based Clustering for Monaural Blind Source Separation. *Proceedings of the 12th International Conference on Digital Audio Effects*, 2009.
- [39] Fabian-Robert Stoter, Antoine Liutkus, Roland Badeau, Bernd Edler, and Paul Magron. Common Fate Model for Unison Source Separation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 126–130. IEEE, March 2016.
- [40] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. The 2018 signal separation evaluation campaign. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 293–305. Springer, 2018.
- [41] Patrick Vandewalle, Jelena Kovacevic, and Martin Vetterli. Reproducible Research in Signal Processing.

- IEEE Signal Processing Magazine*, 26(3):37–47, May 2009.
- [42] Emmanuel Vincent, Shoko Araki, Fabian Theis, Guido Nolte, Pau Bofill, Hiroshi Sawada, Alexey Ozerov, Vikram Gowreesunker, Dominik Lutter, and Ngoc Q.K. Duong. The Signal Separation Evaluation Campaign (2007-2010): Achievements and Remaining Challenges. *Signal Processing*, 92(8):1928–1936, August 2012.
 - [43] DeLiang Wang. On ideal binary mask as the computational goal of auditory scene analysis. In *Speech separation by humans and machines*, pages 181–197. Springer, 2005.
 - [44] Guy Wolf, Stephane Mallat, and Shihab Shamma. Rigid Motion Model for Audio Source Separation. *IEEE Transactions on Signal Processing*, 64(7):1822–1831, April 2016.

IMPROVING BASS SALIENCY ESTIMATION USING LABEL PROPAGATION AND TRANSFER LEARNING

Jakob Abeßer¹

Stefan Balke²

Meinard Müller²

¹ Semantic Music Technologies Group, Fraunhofer IDMT, Ilmenau, Germany

² International Audio Laboratories Erlangen, Germany

`jakob.abesser@idmt.fraunhofer.de`

ABSTRACT

In this paper, we consider two methods to improve an algorithm for bass saliency estimation in jazz ensemble recordings which are based on deep neural networks. First, we apply label propagation to increase the amount of training data by transferring pitch labels from our labeled dataset to unlabeled audio recordings using a spectral similarity measure. Second, we study in several transfer learning experiments, whether isolated note recordings can be beneficial for pre-training a model which is later fine-tuned on ensemble recordings. Our results indicate that both strategies can improve the performance on bass saliency estimation by up to five percent in accuracy.

1. INTRODUCTION

Recent developments in the field of machine learning, in particular deep learning, stimulated a significant performance boost in various Music Information Retrieval (MIR) tasks [7] such as audio tagging [23], audio source separation [27], and automatic music transcription (AMT) [15]. One major challenge in training deep neural networks (DNNs) that generalize well to unseen data lies in the large amount of required labeled training data, which is often not available.

In this context, semi-supervised learning strategies can help to solve this data problem. A first approach is to apply *transfer learning*, i. e., training a network on a related classification task and fine-tune the model parameters for the target task with the (usually smaller) amount of training data at hand [10, 18]. Both training steps are fully supervised and therefore require labeled datasets. A second approach is *label propagation*, where labels from labeled feature vectors are propagated to unlabeled feature vectors if some pre-defined similarity measure exceeds a particular threshold. Label propagation can help to significantly increase the amount of available training data.

In this paper, we focus on the task of estimating the pitch salience of the bass instrument in jazz ensemble

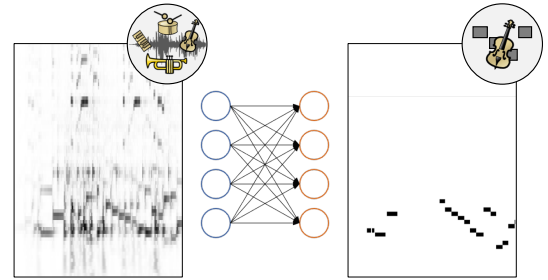


Figure 1. Flowchart summarizing the main idea of training a deep neural network to learn a mapping function from a constant-Q spectrogram of a jazz ensemble recording (left) towards a bass pitch saliency representation (right) using a deep neural network.

recordings. In general, pitch saliency refers to a likelihood measure of an instrument playing certain pitch frequencies at given times. Figure 1 illustrates the DNN-based approach that we use. Given a time-frequency representation of an audio recording of a jazz ensemble, the goal is to estimate a bass salience representation in a frame-wise fashion. As outlined in [1], these frame-wise estimates of the bass salience can then be aggregated using beat annotations to obtain a beat-wise pitch representation, which is a musically meaningful approximation of the commonly played walking bass lines in jazz music.

As the main contributions of this paper, we investigate transfer learning and label propagation strategies for improving fully-connected deep neural networks for the task of bass saliency estimation, as shown in Figure 2. Both techniques aim to compensate the lack of available labeled data for the task of bass salience estimation. For label propagation, the core idea is to enrich an unlabeled dataset with labels from a labeled dataset. For transfer learning, we investigate whether training models on music data of lower timbral complexity (e. g., isolated instrument tones) is beneficial for transferring them to complex mixture recordings.

The remainder of this paper is structured as follows. In Section 2, we review related work. In Section 3, we introduce the underlying datasets used throughout our experiments and propose additional data augmentation steps. Section 4 introduces the feature extraction approach, DNN architecture, and the evaluation methodology. In Section 5,



© Jakob Abeßer, Stefan Balke, Meinard Müller. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Jakob Abeßer, Stefan Balke, Meinard Müller. “Improving Bass Saliency Estimation using Label Propagation and Transfer Learning”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

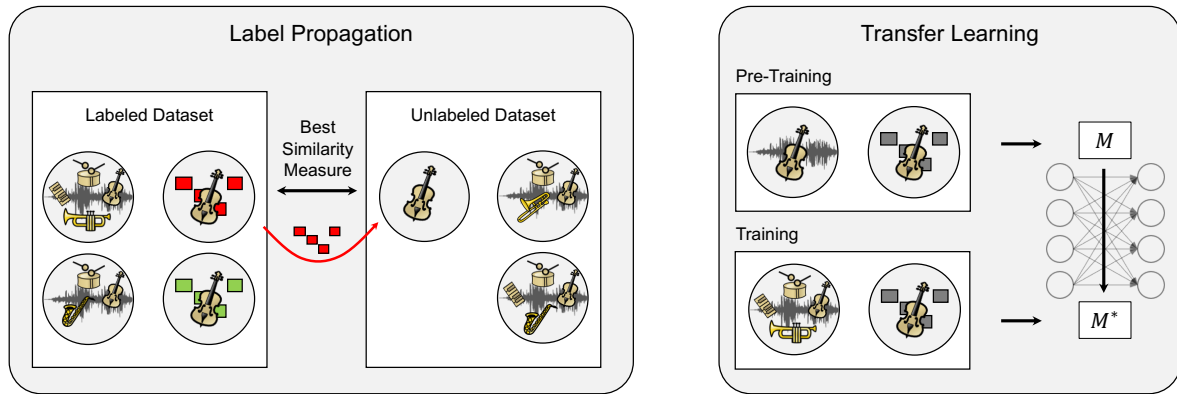


Figure 2. Illustration of label propagation and transfer learning. In label propagation (left), an unlabeled audio dataset is enriched with frame-wise pitch labels from a labeled dataset. In transfer learning (right), a DNN is first pre-trained on a dataset with lower complexity (isolated bass recordings) and then trained further on jazz ensemble recordings.

we present experiments towards hyperparameter optimization (Section 5.1), label propagation (Section 5.2), as well as transfer learning (Section 5.3). Finally, Section 6 concludes our work and gives perspectives towards future work.

2. RELATED WORK

Saliency representations are a common intermediate representation in many automatic music transcription (AMT) systems prior to the formation of note events. Most previous approaches for bass saliency estimation rely on hand-crafted algorithms rather than on automatically learnt mappings. For instance, Goto derives pitch saliency values from predominant peaks in a spectral representation based on instantaneous frequency values [13]. Ryyänen and Klapuri compute a saliency measure for a given pitch from a weighted sum over the spectral magnitude values at its harmonic frequencies [24]. Salamon et al. apply harmonic summation based on a logarithmic frequency representation combined with instantaneous frequency estimation methods [25].

In [1], the mapping from a constant-Q spectrogram to a bass saliency function is automatically learnt using fully-connected deep neural networks. The authors also investigated a semi-supervised learning step where parts of predicted pitch saliency estimations on unlabeled audio data were added to the training data based on a sparsity criterion. The modeling strategy was inspired by Balke et al. [2], who used a similar approach to estimate a saliency representation of the predominant melody instrument in jazz music recordings. Bittner et al. [4] proposed a fully convolutional neural network (CNN) to extract a saliency representations from different constant-Q transforms used as input for both multiple fundamental frequency estimation and melody tracking.

Models with state-of-the-art performance in related disciplines such as image processing (mostly CNN-based models) are rarely trained from scratch due to the large amount of required training data. Instead, only the last

Dataset	Usage	Labels	# Feature Vectors	Duration [h]
ISO ⁺	Training	✓	448,626	5.79
WJD ⁺	Training	✓	305,507	3.94
WJD ⁻	Training	-	500,000	6.45
WJD ⁺ -TEST	Test	✓	8,318	0.1

Table 1. Summary of the datasets. The number of feature vectors after data augmentation and voiced frame selection as well as the corresponding duration in hours is given in the last two columns. For the WJD⁻ dataset, 500,000 feature vectors were randomly selected due to memory limitations on the hardware in use for the experiments.

layers of existing “general purpose” classification models (such as the ImageNet model [9]) are fine-tuned for related classification tasks using smaller amounts of training data [21]. Similarly, in the field of MIR, Choi et al. [8] used a pre-trained CNN-based feature extractor trained on music tagging data for related music classification and regression tasks. However, for the task of AMT, no such general-purpose model was established so far.

3. DATASETS

The spectral characteristics of the targeted upright bass tones are affected by different factors of variation such as the pitch, the loudness, as well as the overlap with tones from simultaneously playing instruments. In our considered datasets, we use different sets of upright bass recordings that try to address these variations. All considered audio files used in this paper include an acoustic upright bass played with the plucked (pizzicato) plucking style—as opposed to using a bow—as this is the common playing style for jazz bass players. Table 1 gives an overview of the datasets used, which we discuss in the following.

3.1 Isolated Upright Bass Recordings (ISO⁺)

The ISO⁺ dataset is a collection of isolated chromatic note recordings. The recordings stem from various commercial and non-commercial upright bass sample datasets: Adam

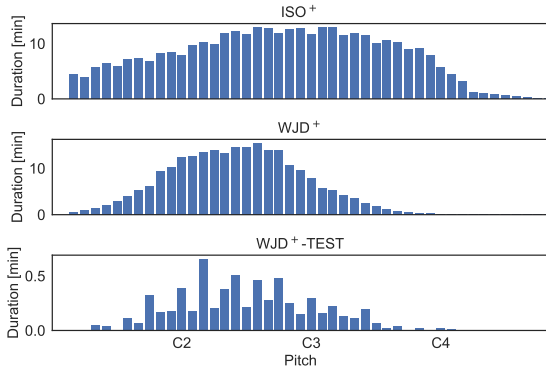


Figure 3. Pitch histogram over labeled datasets ISO^+ , WJD^+ , and WJD^+-TEST after data augmentation. Total duration of all notes in minutes is shown for each pitch.

Monroe’s Upright Bass Sample Library, Meatbass, Trillian, Steinberg Halion Symphonic Orchestra, and SWAM Double Bass. Furthermore, we collected recordings from the Real World Computing Music Database (RWC) [14], the McGill University Master Samples [11] and the Iowa Classical Instrument Samples¹.

3.2 Jazz Ensemble Recordings (WJD)

The Weimar Jazz Database (WJD) contains 456 manually transcribed solos from famous jazz recordings [22]. For a subset of 40 of these recordings, excerpts of walking bass lines using the Sonic Visualiser software [6]. All of the selected recordings are typical jazz ensembles that consist of upright bass, drums, piano, as well as melody instruments such as trumpet or saxophone. We use 30 of these annotated recordings for training (WJD^+) and 10 for testing (WJD^+-TEST). The remaining 416 recordings from the WJD are denoted by WJD^- . These recordings come without bass pitch annotations and will be used in the label propagation experiment detailed in Section 5.2.

3.3 Data Augmentation

On the datasets that we use for supervised training (ISO^+ and WJD^+), we generated 15 augmented versions from each original audio file by combining three time-stretching settings (stretch factors 0.9, 1, and 1.1) and five pitch-shifting settings (shifts between -2 and +2 semitones) using the software package sox².

For all labeled datasets, we discard all non-voiced frames. Furthermore, we only keep the spectral frames from the first 75 % of the note duration as especially higher harmonics from upright bass tones decay much faster than the fundamental frequency contours. In order to make the final results comparable to [1], data augmentation is not applied to the test set WJD^+-TEST (compare Section 3).

Figure 3 illustrates the pitch distribution over the three labeled datasets after applying data augmentation. While the WJD^+ and WJD^+-TEST datasets similarly include

Hyperparameter	Search Space	Importance
Magnitude scaling	{ linear , logarithmic}	0.015
# hidden layers	$n \in \{3, 4, 5, 6\}$	0.020
Hidden layer size	$H = 2^h, h \in \{7, 8, 9, 10\}$	0.040
Learning rate	$\alpha = 10^r, r \in [-3, -6], (-4.27)$	0.485
Batch normalization	{no, yes}	0.017
ℓ_2 weight regularization	$\lambda \in \{0, 10^{-4}, 10^{-3}, 10^{-2}\}$	0.038
Dropout ratio	$d \in [0, 0.5], (0.06)$	0.385

Table 2. Search space for hyperparameter optimization. Optimal parameter set for $a_{\text{val, opt}} = 0.62$ is given in bold font. Feature importance values in a random forest regression model are shown in the last column.

notes up to C4, the isolated tones in ISO^+ cover a wider pitch range and distribute among the pitches in a more balanced fashion.

4. METHODOLOGY

4.1 Feature Extraction

Audio files are resampled to 22.05 kHz before constant-Q magnitude spectrograms are computed with a hopsize of 1024 samples (46.4 ms) and a frequency resolution of 12 bins per octave between 34.65 Hz (MIDI pitch 25, note D^b1) and 1567 Hz (MIDI pitch 91, note G6) using the librosa Python library [19]. Hence, the input vectors have the dimensionality of 67. In contrast to [1], we extend the frequency range by a small margin in the low frequency range and by two octaves in the upper frequency range in order to incorporate overtone frequencies of higher bass notes. In Section 5.1 we evaluate, to which extent a logarithmic compression of the magnitude spectrogram is beneficial for bass saliency.

4.2 Deep Neural Network Architecture

Throughout this paper, we use a fully-connected network architecture for the given task of bass saliency estimation, see Table 2 for an overview of parameters. The model is a cascade of n hidden layers of size H with optional intermediate layers for batch normalization (prior to the ReLU activation function) [16] and dropout (dropout ratio d) [26]. In contrast to [1], we do not use frame stacking here as we aim to directly compare local feature vectors later in the label propagation step described in Section 5.2. The model instead processes individual spectrogram frames as input and predicts the corresponding pitch saliency vector. Furthermore, all hidden layers have an optional ℓ_2 weight regularization (with regularization parameter λ) [12]. For each model training, we use 500 training epochs, a batch size of 250, early stopping with a patience of 25 epochs based on the validation accuracy, and the categorical cross-entropy as loss function. The keras³ Python library is used for all experiments in this paper.

Score annotations are converted into frame-wise binary pitch activities, which are used as targets for the model training. In the annotated datasets used in this paper, bass lines are strictly monophonic. In the final layer, we use a

¹ <http://theremin.music.uiowa.edu/MIS.html>

² <http://sox.sourceforge.net>

³ <https://www.keras.io>

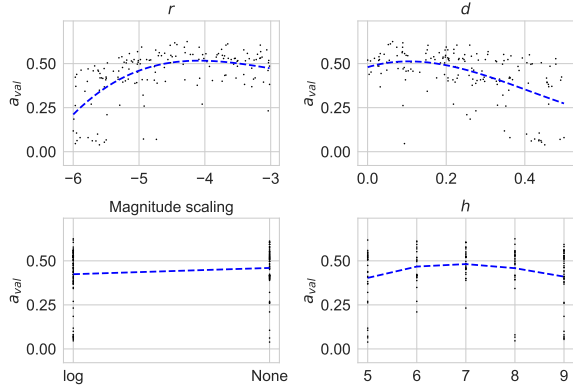


Figure 4. Validation accuracy a_{val} and hyperparameter values for learning rate exponent r , dropout ratio d , magnitude scaling, and layer size exponent h over all random parameter configurations (black dots). Cubic regression lines (blue dashed lines) show trends in the data. Optimal values for the other hyperparameters are given in Table 2.

sigmoid instead of a softmax activation function to be able to model the activity of all pitches independently. This also allows us to model polyphonic parts or rests within bass lines. However, in order to compare our results with [1], we only focus on bass saliency estimation from voiced frames in this paper and leave bass voicing detection open for future work.

As pitch range for the targets, we use [26, 69] (notes D1 to G4) whereas in [1], a slightly smaller pitch range [28, 67] (notes E1 to F4) was used. The dimensionality of the target vectors is 44.

4.3 Evaluation

We derive pitch estimates by looking at the highest output value of the final sigmoid layer. For the evaluation, we use the standard evaluation measures *Raw Pitch Accuracy* (denoted as a) and *Raw Chroma Accuracy* (denoted as a_{12}) as used in the MIREX Audio Melody Extraction task. For the definition of these measures, we refer to [20]. During training, we randomly split the training dataset(s) into training and validation dataset based on a 80:20 split. Accuracy values a_{train} , a_{val} , and a_{test} are computed on the training, validation, and test set, respectively.

5. EXPERIMENTS

5.1 Hyperparameter Optimization

A systematic grid search of possible hyperparameter combinations is not feasible for deep neural networks due to the high computational costs. In our approach, we train 160 models with different combinations of hyperparameters. These combinations are randomly sampled from the hyperparameters given in Table 2. The best hyperparameter combination is then retrieved by testing the model performance on the validation set (a_{val}). Figure 4 shows the validation set accuracy for the different hyperparameter

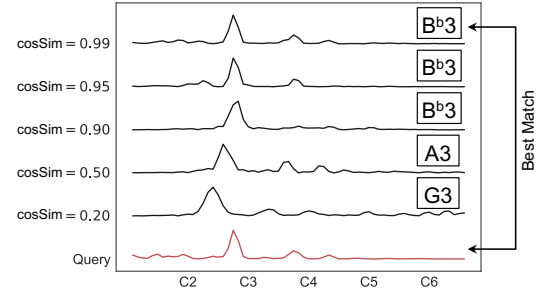


Figure 5. Label propagation example: given a query constant-Q spectrogram frame with unknown pitch (bottom, red), candidates with different cosSim similarity values are shown (above, black). The pitch label B^b3 will be transferred from the most similar candidate shown on top ($s = 0.99$).

configurations. To get an intuition about the influence of the different hyperparameters, we follow an approach previously presented in [17]. In that approach, a random forest regression model [5] is fitted to a_{val} over all parameter configurations. From the random forest regression model, we can obtain the relative importance of all hyperparameters, see Table 2 (third column) for the results.

As previously found in [17], the learning rate exponent r is by far the most important hyperparameter (0.485) with optimal values around 10^{-4} , as shown in Figure 4. Interestingly, as indicated in Table 2, the dropout ratio d also has a high relative importance (0.385) and an optimal value only slightly above zero.

5.2 Label Propagation

A first approach to enrich the available amount of training data is to use label propagation. We derive pitch labels for feature vectors in the unlabeled WJD^- dataset by transferring labels from their most similar counterparts in WJD^+ dataset. To this end, we compute a similarity score s_i for the i -th feature vector in the WJD^- database $x_i^{\text{WJD}^-} \in \mathbb{R}^{67}$ by maximizing its cosine similarity (cosSim) towards all feature vectors in the WJD^+ database as

$$s_i = \max_k \cos\text{Sim}(x_i^{\text{WJD}^-}, x_k^{\text{WJD}^+}). \quad (1)$$

An example is shown in Figure 5. Given a query spectrogram frames (bottom), we show five example spectrogram frames with different similarity values. The most similar frame ($\cos\text{Sim} = 0.99$) shows an almost identical overtone structure, which motivates the transfer of its pitch label.

As shown in Figure 6, most feature vectors in the unlabeled WJD^- dataset have very similar counterparts in the WJD^+ dataset, which is somewhat intuitive as both datasets originate from the Weimar Jazz Database (WJD). We derive three similarity thresholds $\tau_{25} = 0.914$, $\tau_{50} = 0.940$, and $\tau_{75} = 0.96$ from the 25th, 50th, and 75th percentile of the distribution over s . The label-enriched WJD^- dataset is denoted as $(\text{WJD}^-)^+$ in the following.

We use the best model architecture obtained via hyperparameter optimization (see Section 5.1) and train

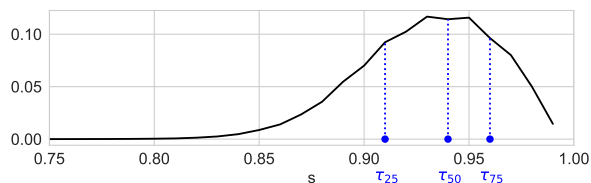


Figure 6. Histogram over best-match cosine-similarity values for mapping feature vectors from the unlabeled dataset WJD^- to the labeled dataset WJD^+ . Similarity thresholds τ_{25} , τ_{50} , and τ_{75} are derived from the respective percentiles of the distribution over s .

models from different training sets. For that purpose, we combine the full WJD^+ dataset with feature vectors of the $(WJD^-)^+$ dataset based on the criterion $\tau_- \leq s_i \leq \tau_+$. We test different pairs (τ_-, τ_+) using combinations of the percentile-based thresholds τ_{25} , τ_{50} , and τ_{75} as well as $\tau_0 = 0$ and $\tau_{100} = 1$ as shown in the lower subplot of Figure 7.

Feature vectors with lower similarity scores more likely introduce label noise to the mixed training dataset. Since the WJD^+ dataset only contains voiced frames, even unvoiced frames in the WJD^- will be mapped to voiced frames. This is a drawback due to the given dataset configurations. Another possible reason for low similarity scores are notes played by other instruments in the ensemble such as the piano or the soloist. However, voiced frames from the WJD^- database with a lower similarity can provide novel information for the classification task, which can help to improve the existing model. At the same time, feature vectors with high similarity scores can be redundant without providing much novel information for the pitch saliency estimation task. In contrast to [1], label propagation is performed based on feature vector similarity and not based on predictions of existing models.

From the results shown in Figure 7, we make the following observations for all configurations. First, we observe that difference between a_{train} and a_{val} (overfitting) remains almost constant across different configurations. Also, the raw chroma accuracy $a_{\text{test},12}$ is consistently about 0.07 higher than the raw pitch accuracy a_{test} , which indicates that octave errors make up only a small fraction of the remaining pitch estimation errors.

Using the WJD^+ dataset alone or combined with the most similar feature vectors in WJD^- (configurations 0:0 and 75:100), we observe that the training and validation accuracies are clearly higher than the test accuracy. The reason is that these configurations correspond to the best parameter settings found in the hyperparameter optimization step (compare Section 5.1), where maximizing a_{val} was the main objective. Due to their small size, the data distribution in the WJD^+ and WJD^+ -TEST datasets presumably is only similar to a certain degree although both are taken from the Weimar Jazz Database.

In contrast, by adding feature vectors from the WJD^- dataset with lower similarity values and higher novelty (configurations 0:25, 0:50, and 0:75), the modeling task

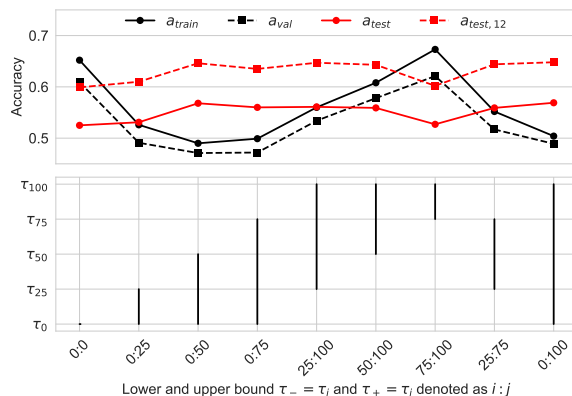


Figure 7. Label propagation results for different dataset configurations (see Section 5.2). Training accuracy a_{train} , validation set accuracy a_{val} , test set accuracy a_{test} , and chroma pitch accuracy $a_{\text{test},12}$ are shown.

becomes harder and the training and validation accuracies decrease. Interestingly, the models' ability to generalize to the test set improves and a_{test} increases. The relatively high difference between validation and test accuracy of up to 0.09 indicates that the small test set size needs to be increased in future work, as both, test and validation set, should come from the same distribution.

For the configurations 0:50 and 0:100, we observe the highest test accuracy of around $a_{\text{test}} = 0.57$. This result is notable as by using label propagation, we are able to train a model which achieves a performance comparable to the highest test accuracy reported in [1] without requiring additional temporal context information using frame stacking. Therefore, label propagation seems a promising approach to improve the model performance.

5.3 Transfer Learning

State-of-the-art music transcription algorithms based on spectral decomposition algorithms such as Non-Negative Matrix Factorization (NMF) are commonly initialized with isolated instrument tones, e. g., for learning spectral note templates [3]. We aim to investigate to which extent a similar strategy can be used to improve neural networks for pitch saliency estimation tasks. As an alternative, we want to find out if it is instead better to train the networks solely on more complex instrumental mixtures (ensemble recordings, see Section 3.2) as these are more similar to the final test data.

We compare three training scenarios in our experiment. First, we train the model solely using the isolated bass tones (ISO^+ dataset) to evaluate the generalization potential of the trained model towards mixture signals in the test set. Secondly, we apply transfer learning, i. e., we pre-train an initial model for bass saliency estimation using isolated bass tracks (ISO^+ dataset) and then fine-tune the model in a second training step on the WJD^+ dataset. In a third scenario, we mix and shuffle the WJD^+ and ISO^+ datasets and perform a single training step. The model trained only

on the WJD^+ dataset serves as baseline for comparison. We train for 750 epochs for both training steps but apply early stopping as detailed in Section 4.2 if possible.

The results are shown in Table 3. Accuracy values are computed on a macro-level by averaging across all spectrogram frames of the test set files. We observe that a pitch saliency model, which is only trained on isolated tones of the target instrument, is not capable to generalize to more complex mixtures as it performs poorly on the test set ($a_{\text{test}} = 0.095$). Combining pre-training on the isolated note database with fine-tuning on the mixture dataset improves the performance by around four percent on the test set accuracy ($a_{\text{test}} = 0.542$) compared to a baseline model, which is only trained on the mixture recordings. The best configuration improves on the test set accuracy by 6 percent ($a_{\text{test}} = 0.561$) compared to the baseline model. It does not involve a pre-training step but uses a mix of isolated and mixed recordings (ISO^+ and WJD^+) for training instead.

The results of the transfer learning experiments suggest that combining training data with different levels of complexity, i. e., different amount of instrumental overlap, can be useful to improve DNN-based models for pitch saliency estimation in ensemble recordings. By using a mixture of both isolated and mixed recordings in one training step, it appears as if the neural network learns best to “focus” on the targeted instrument. Future work could address a different order in the training process, i. e., first training on the mixture tracks and then fine-tuning the model on the isolated note recordings.

6. CONCLUSION

We investigated strategies for label propagation and transfer learning in order to improve bass saliency estimation using deep neural networks. We could show that unlabeled feature vectors from datasets with a similar spectral distribution as the target scenario can be mapped towards labeled datasets to derive pitch labels. By combining labeled datasets and unlabeled datasets through label propagation, we were able to improve the model’s accuracy by around six percent compared to a baseline model. Similarly, we could show that by combining isolated note recordings of the targeted instrument with mixture recordings as training set, we gain around five percent in accuracy. This joint training slightly outperformed our considered transfer learning strategy with two successive training steps. Future work could deal with strategies on how to combine frame-stacking (compare [1]), label propagation, and transfer learning.

For a systematic bias–variance analysis of the given modeling task, it remains challenging to define human level performance as we only focus on frame-wise pitch estimation here. Human experts, i. e., musicians or musicologists, are capable to generate near-perfect note-wise transcriptions. This related task, however, involves listening to longer audio excerpts and allows to include additional cues from the metric structure, tone duration, and local harmony.

Pre-Training	Training	a_{train}	a_{val}	a_{test}	$a_{\text{test},12}$
-	WJD^+ (baseline)	0.665	0.614	0.508	0.589
-	ISO^+	0.514	0.538	0.095	0.234
ISO^+	WJD^+	0.652	0.603	0.542	0.614
-	ISO^+ & WJD^+	0.507	0.531	0.561	0.655

Table 3. Performance comparison with and without transfer learning. All experiments were evaluated using the WJD^+ -TEST dataset.

7. ACKNOWLEDGEMENTS

This work has been supported by the German Research Foundation (MU 2686/11-1, AB 675/2-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

8. REFERENCES

- [1] Jakob Abeßer, Stefan Balke, Klaus Frieler, Martin Pfeleiderer, and Meinard Müller. Deep learning for jazz walking bass transcription. In *Proceedings of the AES International Conference on Semantic Audio*, pages 202–209, Erlangen, Germany, 2017.
- [2] Stefan Balke, Christian Dittmar, Jakob Abeßer, and Meinard Müller. Data-driven solo voice enhancement for jazz music retrieval. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 196–200, New Orleans, USA, 2017.
- [3] Emmanouil Benetos and Tillman Weyde. An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 701–707, Málaga, Spain, 2015.
- [4] Rachel M. Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan P. Bello. Deep salience representations for F0 tracking in polyphonic music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 63–70, Suzhou, China, 2017.
- [5] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [6] C. Cannam, C. Landone, and M. Sandler. Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the ACM Multimedia 2010 International Conference*, pages 1467–1468, Firenze, Italy, October 2010.
- [7] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark Sandler. A Tutorial on Deep Learning for Music Information Retrieval. *ArXiv e-prints*, September 2017.

- [8] Keunwoo Choi, György Fazekas, Mark B. Sandler, and Kyunghyun Cho. Transfer learning for music classification and regression tasks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 141–149, Suzhou, China, 2017.
- [9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, Miami, FL, USA, 2009.
- [10] Aleksandr Diment and Tuomas Virtanen. Transfer learning of weakly labelled audio. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 6–10, New Paltz, NY, USA, 2017.
- [11] Tuomas Eerola and Rafael Ferror. Instrument library (MUMS) revised. *Music Perception*, 25(3):253–255, 2008.
- [12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [13] Masataka Goto. A real-time music-scene-description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication (ISCA Journal)*, 43(4):311–329, 2004.
- [14] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Music genre database and musical instrument sound database. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 229–230, Baltimore, Maryland, USA, 2003.
- [15] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and Frames: Dual-Objective Piano Transcription. *ArXiv e-prints*, October 2017.
- [16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on International Conference on Machine Learning (ICML)*, pages 448–456, 2015.
- [17] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 475–481, New York City, USA, 2016.
- [18] Anurag Kumar, Maksim Khadkevich, and Christian Fügen. Knowledge Transfer from Weakly Labeled Audio using Convolutional Neural Network for Sound Events and Scenes. *ArXiv e-prints*, November 2017.
- [19] Brian McFee, Colin Raffel, Dawen Liang, Daniel P. W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the Scientific Computing with Python conference (Scipy)*, Austin, Texas, 2015.
- [20] MIREX. Audio melody extraction task. Website http://www.music-ir.org/mirex/wiki/2016:Audio_Melody_Extraction, last accessed 03/29/2018, 2016.
- [21] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- [22] Martin Pfeleiderer, Klaus Frieler, Jakob Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart. *Inside the Jazzomat. New perspectives for jazz research*. Schott Campus, Mainz, Germany, 2017.
- [23] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik M. Schmidt, Andreas F. Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, pages 1–5, Long Beach, CA, USA, 2015.
- [24] Matti Ryynänen and Anssi P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.
- [25] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, 2012.
- [26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.
- [27] Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 261–265, New Orleans, USA, 2017.

IMPROVING PEAK-PICKING USING MULTIPLE TIME-STEP LOSS FUNCTIONS

Carl Southall, Ryan Stables and Jason Hockman

DMT Lab, Birmingham City University, Birmingham, United Kingdom

{carl.southall, ryan.stables, jason.hockman}@bcu.ac.uk

ABSTRACT

The majority of state-of-the-art methods for music information retrieval (MIR) tasks now utilise deep learning methods reliant on minimisation of loss functions such as cross entropy. For tasks that include framewise binary classification (e.g., onset detection, music transcription) classes are derived from output activation functions by identifying points of local maxima, or peaks. However, the operating principles behind peak picking are different to that of the cross entropy loss function, which minimises the absolute difference between the output and target values for a single frame. To generate activation functions more suited to peak-picking, we propose two versions of a new loss function that incorporates information from multiple time-steps: 1) *multi-individual*, which uses multiple individual time-step cross entropies; and 2) *multi-difference*, which directly compares the difference between sequential time-step outputs. We evaluate the newly proposed loss functions alongside standard cross entropy in the popular MIR tasks of onset detection and automatic drum transcription. The results highlight the effectiveness of these loss functions in the improvement of overall system accuracies for both MIR tasks. Additionally, directly comparing the output from sequential time-steps in the multi-difference approach achieves the highest performance.

1. INTRODUCTION

At present, the state-of-the-art systems for many music information retrieval (MIR) tasks utilise deep learning models. Within the domain of dynamic time-series MIR tasks such as onset detection and music transcription, solutions are achieved through a binary classification of each time-step t . A binary classification output is typically limited to a range of $[0,1]$ using a non-linear function (e.g., sigmoid, softmax). For classification purposes the output is subsequently rounded to either 0 or 1. However, in framewise binary classification tasks using this approach has proven to be less effective [7]. In the example presented in Figure 1, a framewise output activation function \tilde{y} is shown in

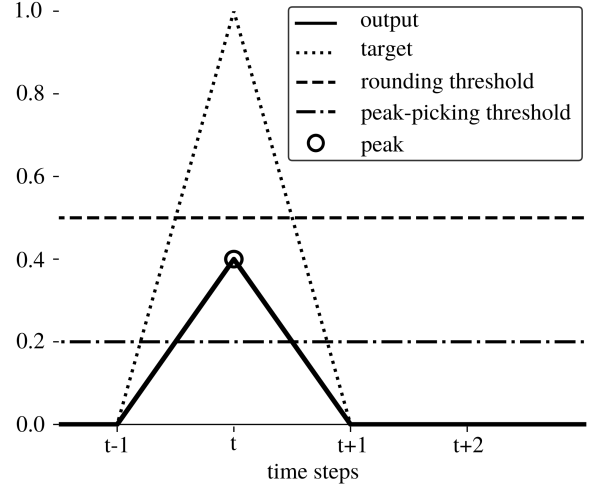


Figure 1. A true positive is missed using the rounding approach, but is successfully selected through peak picking (circled point). The solid line denotes the output, the dotted line the target, the dashed line the 0.5 rounding threshold and the dash-dotted line the peak-picking threshold.

which the values ideally associated with a class label (i.e., value) of 1 do not exceed 0.5. While \tilde{y} clearly shows the presence of an event as a *peak*, it would be identified as a false negative ($\tilde{y}^t < 0.5$).

1.1 Peak Picking

To overcome the problem posed in Figure 1, the majority of framewise binary classification systems utilise peak picking, which differentiates between classes by identifying local maxima. Multiple peak-picking approaches have been proposed in the literature [1, 4, 12, 16] and follow a general process as shown in Figure 1. Here, a point is selected as a peak if it is the maximum value within a local window and above a threshold τ . In [16] the threshold is determined by calculating the mean of a window, controlled using δ , a user determined constant λ and maximum and minimum values ($tmax$ and $tmin$).

$$\tau^t = \text{mean}(\tilde{y}^{t-\delta} : \tilde{y}^{t+\delta}) * \lambda \quad (1)$$

$$\tau^t = \begin{cases} tmax, & \tau > tmax \\ tmin, & \tau < tmin \end{cases} \quad (2)$$



© Carl Southall, Ryan Stables and Jason Hockman. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Carl Southall, Ryan Stables and Jason Hockman. “Improving Peak-picking Using Multiple Time-step Loss Functions”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

An onset classification vector O is achieved by determining if each time-step of \tilde{y} is the maximum value within the surrounding number of frames, set using Ω , and above the threshold τ :

$$O^t = \begin{cases} 1, & \tilde{y}^t == \max(\tilde{y}^{t-\Omega} : \tilde{y}^{t+\Omega}) \quad \& \quad \tilde{y}^t > \tau^t \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

1.2 Loss Functions

The overall loss (often referred to as the *cost*) \mathcal{L} represents the error of a system within a single value. It is calculated by comparing the difference between the desired ground truth y and the actual output \tilde{y} [10]. Within audio based time-step classification tasks it is calculated by taking the mean of the individual time-step losses l^t :

$$\mathcal{L} = \frac{1}{T} \sum_{t=1}^T l^t. \quad (4)$$

\mathcal{L} is a component of back propagation (and truncated back propagation) which is used to calculate the gradients \mathcal{G} used in updating the trainable parameters of the model Θ with learning rate μ .

$$\Theta \leftarrow \Theta - \mu \cdot \mathcal{G} \quad (5)$$

Commonly used loss functions for calculating l^t include mean squared error (MS) (eq. 6) and cross entropy (CE) (eq. 7) [5].

$$l_{ms}^t\{y^t, \tilde{y}^t\} = (y^t - \tilde{y}^t)^2 \quad (6)$$

$$l_{ce}^t\{y^t, \tilde{y}^t\} = y^t \log(\tilde{y}^t) + (1 - y^t) \log(1 - \tilde{y}^t) \quad (7)$$

Both of these loss functions are suited to differentiating between binary classes using rounding as they aim to minimise the absolute difference between the targets y and the output \tilde{y} . In the majority of MIR tasks CE is more suited than MS due to its greater penalization of large errors [14, 16, 22].

1.3 Motivation

In the peak-picking process, multiple frames are utilized in both the calculation of a threshold as well as the peak selection. However, in the MS and CE calculations only the current time-step t is used in measuring the difference between the target y and output \tilde{y} . In order for the loss to reflect peak salience (i.e., the clarity of the local maxima) and to ensure that the output activation function is suitable for peak-picking, then multiple time-steps should be included within the loss function calculation. To this end, we propose two versions of a new loss function which not only measures the absolute difference between y and \tilde{y} , but also allows for peak salience to be maintained. We then evaluate the worth of these functions within the tasks of onset detection and automatic drum transcription (ADT).

The remainder of this paper is structured as follows: Section 2 presents the proposed loss functions and Section

3 gives an overview of the evaluation. The results and discussion are presented in Section 4 and the conclusions and future work are presented in Section 5.

2. METHOD

For a loss function to represent an understanding of peak salience, it must include at least three points: $\tilde{y}^{t-1} : \tilde{y}^{t+1}$. We propose combining CE and a peak salience measure into a single loss function termed peakiness cross entropy (PCE):

$$l_{pce}^t = \frac{1}{2}(\gamma l_{ce}^t\{y^t, \tilde{y}^t\} + (1 - \gamma)(l_p^t + l_f^t)), \quad (8)$$

where the first part of the equation is the standard cross entropy (CE) of the current time-step t . The second part of the function is a peak salience measure that consists of two variables: l_p , which focuses on the previous time-step and l_f , which focuses on the future ($t + 1$) time step. γ is used to control the weighting between standard CE and the peakiness measure. We propose two methods for achieving l_p and l_f : a combination of multiple individual time-step calculations and a direct comparison of the differences between multiple time-steps.

2.1 Multi-individual

The multi-individual (MI) method calculates l_p and l_f as individual time step cross entropies of previous and future time-steps:

$$l_p^t = l_{ce}^t\{y^{t-1}, \tilde{y}^{t-1}\} \quad (9)$$

$$l_f^t = l_{ce}^t\{y^{t+1}, \tilde{y}^{t+1}\}. \quad (10)$$

This ensures that updates to \tilde{y}^t do not cause greater negative updates to \tilde{y}^{t-1} and \tilde{y}^{t+1} .

2.2 Multi-difference

Although MI utilizes multiple time-steps it does not compare absolute differences between sequential time-steps. To achieve this, we propose an additional calculation of l_p and l_f , termed multi-difference (MD), which measures the absolute differences between sequential time-steps of the target y and the output \tilde{y} . The first version of MD (MMD), utilizes MS. The second version (WMD) utilizes an updated version of the CE equation, termed weighted cross entropy (WCE):

$$l_{wce}^t\{y^t, \tilde{y}^t\} = (1 - \phi)y^t \log(\tilde{y}^t) + \phi(1 - y^t) \log(1 - \tilde{y}^t), \quad (11)$$

which allows the strength of each half of the equation to be controlled using the weighting parameter ϕ . The first half of the WCE equation (henceforth referred to as WCE-FN) aims to reduce false negatives by producing a loss value proportional to the difference between sequential time-steps of y^t and \tilde{y}^t . The second half of the WCE equation (hereafter termed WCE-FP) aims to suppress false positives

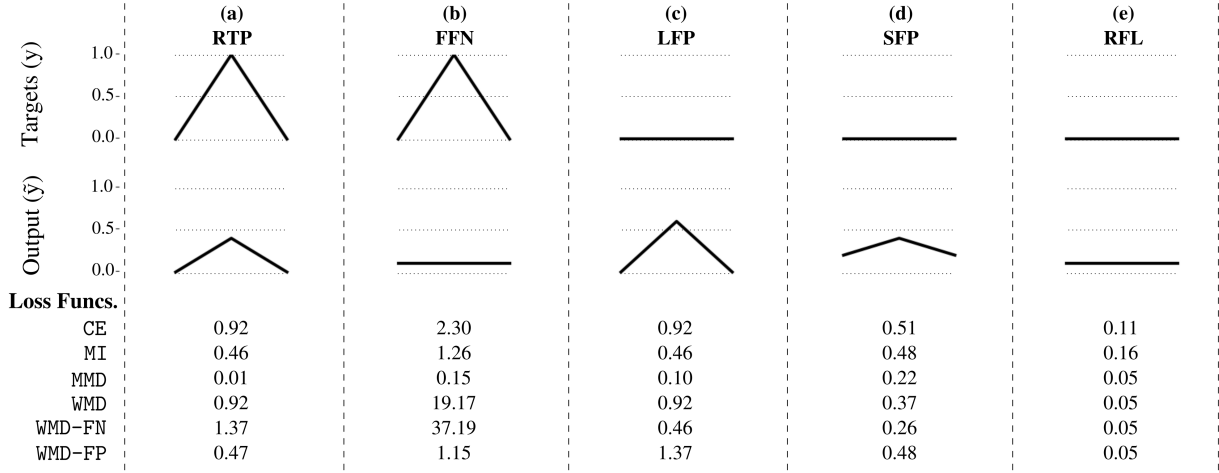


Figure 2. Example activation function scenarios with corresponding loss values output from each loss function. From left to right: a raised true positive (RTP), a flat line false negative (FFN), a large false positive (LFP), a small raised false positive (SFP) and a raised flat line (RFL).

as it outputs a larger value if there is a large undesirable difference between sequential frames. l_p^t and l_f^t in MMD and WMD are calculated respectively using:

$$l_p^t = \begin{cases} l_{wce}^t \{|y^t - y^{t-1}|, |\tilde{y}^t - \tilde{y}^{t-1}|\}, & WMD, \\ l_{ms}^t \{|y^t - y^{t-1}|, |\tilde{y}^t - \tilde{y}^{t-1}|\}, & MMD, \end{cases} \quad (12)$$

$$l_f^t = \begin{cases} l_{wce}^t \{|y^t - y^{t+1}|, |\tilde{y}^t - \tilde{y}^{t+1}|\}, & WMD, \\ l_{ms}^t \{|y^t - y^{t+1}|, |\tilde{y}^t - \tilde{y}^{t+1}|\}, & MMD. \end{cases} \quad (13)$$

Truncated back propagation is used to calculate the gradients for all loss functions. The presented implementation utilises the automatic differentiation functions built into the Tensorflow¹ library for this purpose.

2.3 Example Loss Function Scenarios

Figure 2 presents five example activation function scenarios. The loss values achieved by CE, MI, MMD, WMD and the two separated halves of WMD: WMD-FN ($\phi = 0$) and WMD-FP ($\phi = 1$), are presented with $\gamma = 0.5$. The targets are presented at the top and the output activation function on the bottom. It is worth noting that all of the loss functions that utilize CE can be directly compared but MMD is relative to itself (i.e., the MMD values might seem small relative to the other loss values but not relative to other values of MMD). If all frames of the output are correct then all of the loss functions output zero.

(a) Reduced true positive: The first example shows a reduced true positive where the surrounding frames are correct. In this case CE and WMD output the largest values as this peak could fall below the peak-picking threshold.

(b) Flat line false negative: The second example shows a false negative where the output is a flat line. In this case high relative error values are given by all of the loss functions, however larger error values are given by MMD and especially the FN suppression half of WMD. This example would generally not be selected during peak-picking.

(c) Large false positive: The third example shows a false positive where the surrounding frames are correct. In this case high values are given by CE, MMD and the false positive suppression part of WMD, as this would be an incorrectly selected peak.

(d) Small raised false positive: The fourth example again shows a false positive, similar to the previous example, but the surrounding frames are raised resulting in a less salient false positive. In this case lower values are given by MI and WMD-FP, than CE, as this peak is not as salient as the one in example three (i.e., large false positive).

(e) Raised flat line: The final example presents a raised flat line. In this case the MMD and WMD loss functions penalize less than CE and MI. While the absolute values are slightly wrong, the difference between the sequential frames is correct, resulting in no peaks being correctly chosen.

3. EVALUATION

To identify whether the new loss functions improve performance, we compare the newly proposed loss functions against standard cross entropy (CE) in two common MIR tasks: onset detection (OD) and automatic drum transcription (ADT). To ensure performance trends are consistent with different systems, we implement four neural network

¹ <https://www.tensorflow.org>

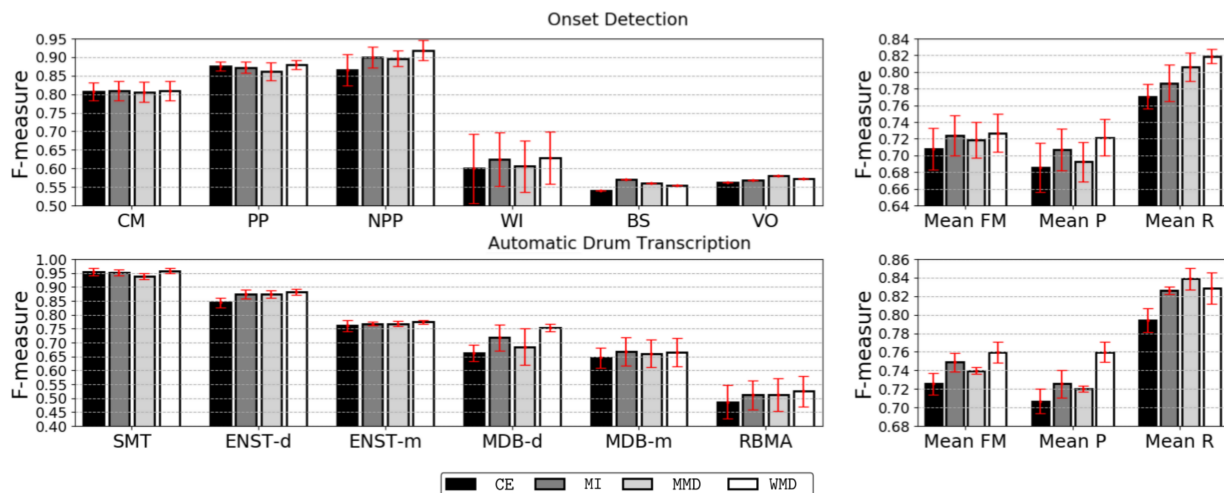


Figure 3. Subset mean system F-measure results for the four implemented cost functions for onset detection and automatic drum transcription. The individual subset F-measure results are on the left and the mean subset F-measure, precision and recall results are on the right. The red error plots display the standard deviations across the folds.

based models which have achieved state-of-the-art results for both of the tasks in recent years. Standard F-measure, derived from precision and recall, is used as the evaluation metric with onset candidates being accepted if they fall within 30ms of the ground truth annotations (i.e., window of acceptance). If onset candidates fall within 30ms of each other, they are combined into a single onset at the middle location (i.e., window of combination).

3.1 Onset Detection (OD)

For the OD evaluation, we utilize the same datasets and subset splits as used in [3], consisting of: complex mixtures (CM), pitched percussion (PP), non-pitched percussion (NPP), wind instruments (WI), bowed strings (BS) and vocals (VO). As OD is a binary classification task, all systems are implemented with a two neuron softmax output layer, one neuron corresponds to an onset and the other neuron corresponds to the absence of an onset.

3.2 Automatic Drum Transcription (ADT)

For the ADT evaluation, we utilize four ADT datasets: IDMT-SMT-Drums [6], ENST-Drums *minus one* subset [8], MDB-Drums [18] and RBMA-2013 [21]. To observe trends between contexts, the datasets are divided into the three groups proposed in [23]: 1) drum only (DTD) consisting of IDMT-SMT-Drums, 2) drums in the presence of extra percussion (DTP) consisting of the drum-only versions ENST-d and MDB-d and 3) drums in the presence of extra percussion and melodic instruments (DTM), which consist of the polyphonic versions ENST-m, MDB-m and RBMA-2013. ENST-m is created by combining the ENST drum tracks and the accompaniment files using ratios of $\frac{2}{3}$ and $\frac{1}{3}$ respectively, as done in [6, 9, 15, 20, 24]. A three-neuron sigmoid output layer is used for all implemented ADT systems, with the neurons corresponding to the three observed drum instruments (i.e., KD, SD and HH).

3.3 Systems

Four different neural network based systems are implemented. All systems consist of the same overlying structure: First, input features are fed into a pre-trained neural network model. Peak-picking is then performed to determine the locations of the onset candidates using the algorithm from [16] (eq.1:3).

3.3.1 Input Features

For both tasks we use the same framewise logarithmic spectral input features x generated using the madmom Python library [2]. The input audio (16-bit .wav file sampled at 44.1 kHz) is segmented into T frames using a Hanning window of m samples ($m = 2048$) with a $\frac{m}{2}$ hop-size. A logarithmic frequency representation of each of the frames is created using a similar process to [22]. The magnitudes of a discrete Fourier are transformed into a logarithmic scale (20Hz–20kHz) using twelve triangular filters per octave. This results in a $84 \times T$ logarithmic spectrogram.

3.3.2 *lstmpB*

The *lstmpB* system is based on the system presented in [23] and the baseline system used in [16]. It consists of two 50-neuron hidden layers containing long short-term memory cells with peephole connections. The input features are processed in a framewise manner.

3.3.3 *lstmpSA3B*

The *lstmpSA3B* system is based on the SA3 system proposed in [16]. It is the same as the *lstmpB* system other than it contains a soft attention mechanism in the output layer. As in the original implementation the attention number a controls the number of attention connections, and is set to three.

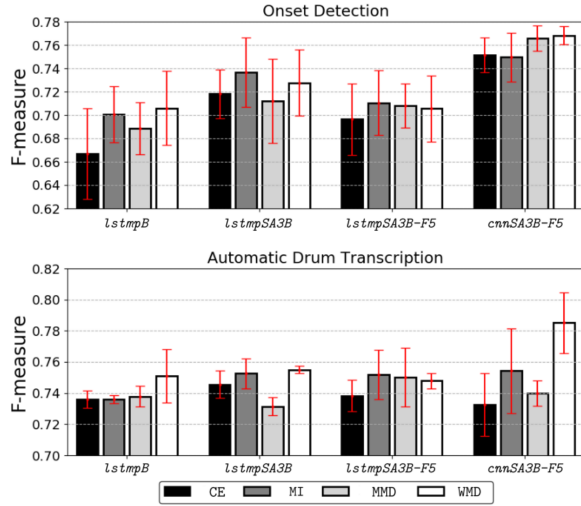


Figure 4. Individual system mean subset F-measure results for the proposed cost functions in OD and ADT tasks. Red bars denote standard deviations across folds.

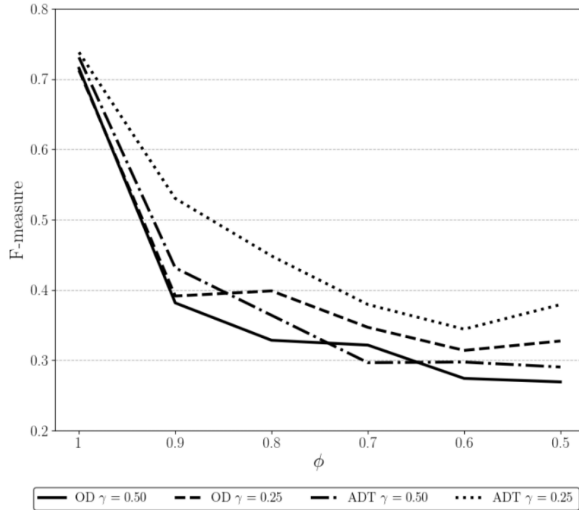


Figure 5. Mean system and mean subset results for different values of WMD parameters (γ and ϕ) in onset detection (OD) and automatic drum transcription (ADT) evaluations.

3.3.4 lstmSA3B-F5

The *lstmSA3B-F5* system is identical to the *lstmSA3B* system with a larger number of input features used. A total of 11 frames (5 either side of the current frame ($x^{t-5} : x^{t+5}$)) are used for each time-step.

3.3.5 cnnSA3B-F5

The *cnnSA3B-F5* [17] combines the convolutional recurrent neural network proposed in [22] and the soft attention mechanism proposed in [16]. It contains two convolutional layers consisting of 3x3 filters, 3x3 max pooling, dropouts [19] and batch normalization [11], with the first layer consisting of 32 channels and the second 64 channels. It contains the same soft attention mechanism output

layer and the same input feature size as *lstmSA3B-F5*.

3.3.6 Training

All systems are trained using mini-batch gradient descent with the Adam optimizer [13]. An initial learning rate of 0.003 is used and three-fold cross validation is performed. Each mini-batch consists of 10 randomly chosen, 100 time-step segments and the data is divided by track into 70% training, 15% validation and 15% testing sets. The training data is used to optimize the systems and the validation data is used to prevent overfitting and to optimize the peak-picking parameters. For datasets containing subsets (i.e., IDMT-SMT Drums and ENST Drums) the splits are performed evenly across the subsets.

4. RESULTS AND DISCUSSION

4.1 Subset Performance

Figure 3 presents the subset results for all cost functions in both evaluations. The red error bars represent standard deviation across folds. The OD results are derived from the mean of the systems and the ADT results are derived from the mean of the systems and the mean of the observed drum instruments (i.e., KD, SD and HH). The left part of the figure presents the individual subset F-measures and the right part of the figure presents the mean subset F-measure, precision and recall. In both MIR tasks, all three of the newly proposed cost functions achieved a higher mean subset F-measure than standard CE, with WMD performing the best in both. Within the ADT evaluation a higher performance is achieved for all three observed drum instruments. A slightly larger increase in performance was witnessed in the ADT task and both versions of the MD cost function achieve higher mean subset F-measures than MI. This highlights that measuring the absolute differences between sequential frames does improve performance. The mean subset precision and recall results highlight that in all cases the newly proposed cost functions achieve higher precision and recall scores than standard CE. In the OD evaluation the highest increase in performance between WMD and CE is in the NPP subset. In the ADT evaluation the largest increase is seen within the DTP subsets (ENST-d and MDB-d). For all subsets in both evaluations the highest F-measure is achieved by one of the three newly proposed cost functions and the error bars show that this improvement occurs in all of the folds. Results from t-tests highlight that the WMD systems improvement over CE within the BS and mean recall OD categories and MDB-d, mean F-measure and precision ADT categories are significant ($p < 0.05$).

4.2 Individual System Performance

Figure 4 presents the individual system, mean subset F-measure results for both MIR evaluations. In all cases the highest F-measure is achieved by one of the newly proposed cost functions, with the WMD cost function achieving the highest F-measure in five of the eight cases. This reinforces that using multiple framed cost functions does

improve performance and that this increase is not just associated with one system. The highest F-measure and the largest increase relative to CE is achieved by the *cnnSA3B-F5* system using the WMD cost function.

4.3 WMD Parameters

Figure 5 presents the mean system, mean subset F-measure results for different parameter settings of the WMD cost function. Plots of six ϕ values for $\gamma = 0.25$ and $\gamma = 0.5$ are presented for both evaluations. For any ϕ values less than one, there is a dramatic decrease in performance which suggests that the false negative suppression half of the WCE function has a negative effect on performance. This is possibly due to the extremely high value given to flat parts of the activation function (see Figure 2), causing these parts of the activation function to become noisy. This suggests that the improvement is due to the false positive suppression half of the WMD system. As this alone achieves higher F-measures than the other proposed cost functions, then it also suggests that their improvement is also due to the suppression of false positives. However, the false negative suppression in those cost functions do not cause repercussions. As γ (i.e., weighting of peak salience measure) increases ($0.5 =$ even weighting with standard CE) then the performance decreases. This trend continues with all values below 0.5 and with the other two proposed loss functions. The highest F-measures were achieved with $\gamma=0.25$ (Standard CE is weighted twice as much as the peak salience measure) for all three proposed loss functions. To categorically identify ideal parameter settings for a particular scenario, a grid search would be required. However, the results suggest that $\gamma = 0.25$ and $\phi = 1$ would always be optimal.

4.4 Understanding the Improvement

After visual comparison of the output activation functions a common situation in which the newly proposed loss functions achieve higher performance was observed. Figure 6 presents an example of this situation, with the top diagram showing the output activation function using CE and the bottom diagram showing the activation function when using the highest performing version of WMD. In the CE diagram, there are two spikes to the right that are wrongly detected as peaks but in the WMD version these peaks are less salient, resulting in no false positives. The consequence of this is that the actual true positive within in the WMD version has a lower amplitude than the one in the CE version. However, this has no effect on performance as the true positive is still a clear peak and correctly chosen within both CE and WMD versions. We believe this situation occurs because within CE a higher error is given to the true positive than the combination of the two smaller false positive errors. This causes the true positive to be closer to the target y but consequentially causes the false positive spikes. Within the WMD version, the false positive suppression assigns a greater loss value to the two false positive spikes than the reduced true positive, ensuring that the spikes are

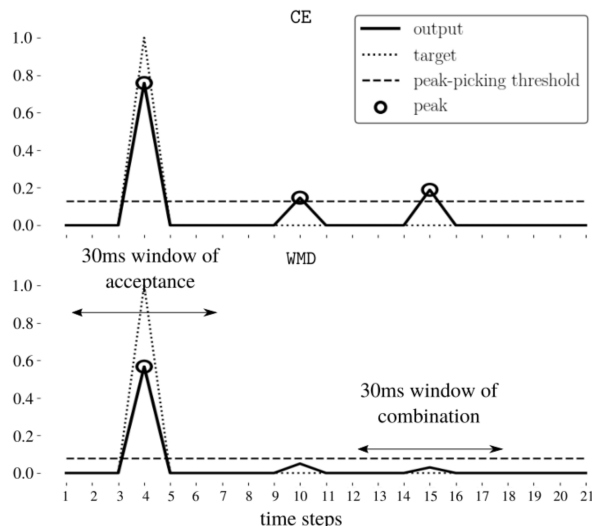


Figure 6. Example of WMD loss function reducing the number of false positives by suppressing false spikes. CE output activation function (top) and WMD output activation function (bottom) with output \hat{y} (solid lines), target y (dotted lines) and the peak-picking threshold (dashed lines). Circles denote selected peaks and arrowed lines show windows of acceptance and combination.

not selected by the peak-picking algorithm. This reduction of noise in the activation function results in less false positives but also enables the peak-picking threshold to be lower, enabling more true positives to be selected. This effect could likely explain both the increase in recall and precision.

5. CONCLUSIONS AND FUTURE WORK

We have developed three new loss functions in an attempt to generate activation functions more suited to peak-picking. The new loss functions utilise information from multiple time-steps which allow them to measure both the absolute values and to maintain peak salience by comparing sequential time-steps. We evaluated the newly proposed loss functions against standard CE using four neural network-based systems in the MIR tasks of onset detection and ADT. The results highlight that all three of the newly proposed cost functions do improve performance, with the WMD loss function achieving the highest accuracy. This work focuses on the inclusion of a single frame on either side of the current time-step. Future work could explore the potential benefit of using a greater number of frames and a version of the WMD equation in which the false negative suppression component does not negatively influence the outcome. Additionally, to make the system end-to-end, the evaluation methodology (i.e., F-measure and tolerance windows) could also be incorporated within the loss functions. Open source implementations of the new loss functions are available online.²

² https://github.com/CarlSouthall/PP_loss_functions

6. REFERENCES

- [1] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 13(5):1–13, 2005.
- [2] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: A new Python audio and music signal processing library. In *Proceedings of the ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 2016.
- [3] Sebastian Böck, Florian Krebs, and Markus Schedl. Evaluating the online capabilities of onset detection methods. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, 2012.
- [4] Sebastian Böck, Jan Schlüter, and Gerhard Widmer. Enhanced peak picking for onset detection with recurrent neural networks. In *Proceedings of the 6th International Workshop on Machine Learning and Music (MML)*, pages 15–18, Prague, Czech Republic, 2013.
- [5] Pieter-Tjerk de Boer, Dirk Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 1 2005.
- [6] Christian Dittmar and Daniel Gärtner. Real-time transcription and separation of drum recordings based on NMF decomposition. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 187–194, Erlangen, Germany, 2014.
- [7] Florian Eyben, Sebastian Böck, Björn Schuller, and Alex Graves. Universal onset detection with bidirectional long-short term memory neural networks. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 589–594, Utrecht, The Netherlands, 2010.
- [8] Olivier Gillet and Gaël Richard. Enst-drums: an extensive audio-visual database for drum signals processing. In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR)*, pages 156–159, Victoria, Canada, 2006.
- [9] Olivier Gillet and Gaël Richard. Transcription and separation of drum signals from polyphonic music. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):529–540, 2008.
- [10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.
- [12] Ismo Kauppinen. Methods for detecting impulsive noise in speech and audio signals,. In *Proceedings of the 14th International Conference on Digital Signal Processing (DSP2002)*, pages 967–970, Santorini, Greece, 2002.
- [13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [14] Jan Schluter and Sebastian Bock. Improved musical onset detection with convolutional neural networks. In *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6979–6983, 2014.
- [15] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription using bi-directional recurrent neural networks. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 591–597, New York City, United States, 2016.
- [16] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 606–612, Suzhou, China, 2017.
- [17] Carl Southall, Ryan Stables, and Jason Hockman. Player vs transcriber: A game approach to data manipulation for automatic drum transcription. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.
- [18] Carl Southall, Chih-Wei Wu, Alexander Lerch, and Jason Hockman. MDB drums an annotated subset of medleyDB for automatic drum transcription. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.
- [19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [20] Richard Vogl, Matthias Dorfer, and Peter Knees. Recurrent neural networks for drum transcription. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 730–736, New York City, United States, 2016.
- [21] Richard Vogl, Matthias Dorfer, and Peter Knees. Drum transcription from polyphonic music with recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 201–205, New Orleans, United States, 2017.

- [22] Richard Vogl, Matthias Dorfer, Gerhard Widmer, and Peter Knees. Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 150–157, Suzhou, China, 2017.
- [23] Chih-Wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, Meinard Müller, and Alexander Lerch. A Review of Automatic Drum Transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(9):1457–1483, 2018.
- [24] Chih-Wei Wu and Alexander Lerch. Drum transcription using partially fixed non-negative matrix factorization with template adaptation. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 257–263, Malaga, Spain, 2015.

ZERO-MEAN CONVOLUTIONS FOR LEVEL-INVARIANT SINGING VOICE DETECTION

Jan Schlüter

Austrian Research Institute for
Artificial Intelligence, Vienna
jan.schluter@ofai.at

Bernhard Lehner

Institute of Computational Perception,
Johannes Kepler University Linz, Austria
bernhard.lehner@jku.at

ABSTRACT

State-of-the-art singing voice detectors are based on classifiers trained on annotated examples. As recently shown, such detectors have an important weakness: Since singing voice is correlated with sound level in training data, classifiers learn to become sensitive to input magnitude, and give different predictions for the same signal at different sound levels. Starting from a Convolutional Neural Network (CNN) trained on logarithmic-magnitude mel spectrogram excerpts, we eliminate this dependency by forcing each first-layer convolutional filter to be *zero-mean* – that is, to have its coefficients sum to zero. In contrast to four other methods – data augmentation, instance normalization, spectral delta features, and per-channel energy normalization (PCEN) – that we evaluated on a large-scale public dataset, zero-mean convolutions achieve perfect sound level invariance without any impact on prediction accuracy or computational requirements. We assume that zero-mean convolutions would be useful for other machine listening tasks requiring robustness to level changes.

1. INTRODUCTION

Automatically annotating the presence of singing voice in a music recording is a challenging task, as singing voice covers a wide range of notes and expressions, is often accompanied by several other instruments, and may be confused with instruments capable of producing similar melody contours. Recent approaches try to capture this variability by training strong classifiers such as deep neural networks on annotated data [9, 12, 14, 20, 22]. While they achieve high accuracies on standard benchmark datasets, classifiers may exploit correlations between inputs and targets that are present in both the training and test data, but are not semantically meaningful (such a classifier is sometimes called a *horse* [24]) or unwanted (leading to *algorithmic bias* [6]). In [13], we demonstrated that three state-of-the-art singing voice detectors – both with hand-designed and learned features – exploit a dependency between singing voice and sound level present in common datasets.

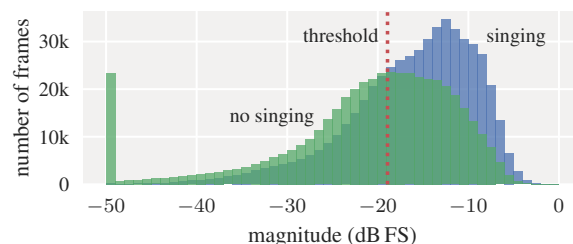


Figure 1: Spectrogram frames of the Jamendo training set containing singing voice tend to have larger magnitudes. A simple threshold allows distinguishing the classes with an accuracy of 61% (8.5 percent points above the baseline).

We can reveal this dependency in a simple experiment: We compute spectrograms for all files of the Jamendo dataset [18] and sum up the linear magnitudes for each frame. The distribution of magnitudes in the training set is clearly skewed towards larger values for frames containing singing voice (Figure 1). Choosing an optimal threshold, we can distinguish vocal from nonvocal frames at an accuracy of 61.1%. With the same threshold, we correctly classify 59.0% of the validation and 68.7% of the test set frames. This is a strong enough improvement over predicting the majority class (52.6%, 51.4% and 53.7%, respectively) that any classifier will pick up this cue. Note that for clarity of presentation, we omitted typical preprocessing steps such as mel scaling, logarithmic magnitude compression or bandwise standardization, but results hardly differ (0.3 percent points improved) with these steps included.

Of course this confound does not stem from inherent characteristics of singing voice, but from production habits in commercial music – if a track contains vocals, those are mixed to stand out. Thus, it affects many other Western-music datasets (we verified this for RWC [8, 16], MSD100 [17], and tracks containing vocals in MedleyDB [3]) that are commonly used for singing voice detection research.

In [13], we argue that to avoid this, datasets should include a sufficient number of instrumental tracks, which cannot feature vocals as the most prominent instrument. And indeed, for the enlarged dataset in [13], there is hardly any linear correlation between input magnitude and class (Figure 2). However, there is still a strong statistical dependency, with vocal frames exhibiting a different magnitude distribution from nonvocal frames, enabling a better-than-chance prediction of the class from the input magnitude.



© Jan Schlüter, Bernhard Lehner. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Jan Schlüter, Bernhard Lehner. “Zero-Mean Convolutions for Level-Invariant Singing Voice Detection”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

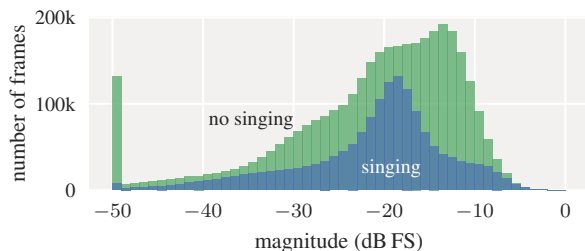


Figure 2: For a dataset including many purely-instrumental tracks, input magnitude and class are not linearly correlated, but still show a clear statistical dependency exploitable by a classifier.

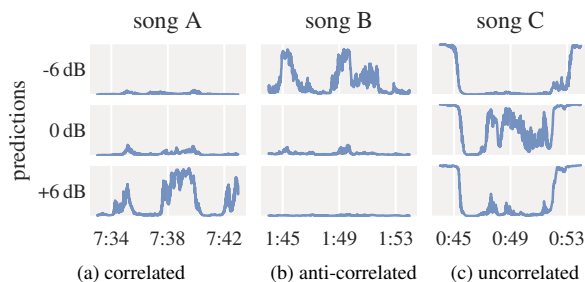


Figure 3: Presenting a state-of-the-art classifier with the same music excerpt at altered sound levels reveals a strong sound level dependency. (a) For some songs, increasing the level by 6 dB increases the classifier’s output. (b) For some, this dependency is inverted. (c) For some, vocals are only detected at the original sound level (second row).

When training a state-of-the-art network on this dataset, it develops a complex sound level dependency: for some test files, predictions are correlated with input magnitude (Fig. 3a), for others, they behave conversely (Fig. 3b) or decrease for any deviation from the original level (Fig. 3c). If and which of these cases applies to a given input seems to depend on the content, not only the original sound level, and sometimes varies from model to model, but the effect appears reliably.

While a closer investigation of the underlying reasons would be highly interesting, for now we content ourselves with stating that this effect is unwanted. As changing the sound level of a music recording does not change the presence of singing voice, we would like a singing voice detector to be invariant to the scale of the input signal. In [13], we show how to achieve this for a system based on hand-designed features. In this work, we propose and evaluate different ways to achieve the same for a Convolutional Neural Network (CNN) trained on mel spectrograms, outperforming the hand-designed system.

The remaining paper is structured as follows: In the next section, we review related work on singing voice detection and level invariance. Section 3 explains the CNN-based baseline system as well as five methods to improve its robustness to level changes, and Section 4 evaluates these methods experimentally. Finally, Section 5 summarizes our findings and their implications.

2. RELATED WORK

From early approaches [2] to recent ones [9, 12, 14, 20, 22], singing voice detection has mostly been addressed with classifiers trained on audio features. Berenzweig et al. [2] based their system on an existing speech recognizer, combined with cepstral coefficients and classified with a simple Gaussian model. Leglaive et al. [12] trained a bidirectional Recurrent Neural Network (RNN) on preprocessed mel spectra, Lehner et al. [14] trained a unidirectional RNN on a set of hand-designed features. Schlüter et al. [22] define the current state of the art using a CNN on logarithmic-magnitude mel spectrograms trained with data augmentation; we will use their public implementation as a starting point. More recent work uses CNNs in attempts to lower annotation effort by learning from song-wise labels [20], or by deriving labels from pairing songs with instrumental versions [9]. The related tasks of auto-tagging (i.e., determining song-wise labels) and singing voice separation are also tackled with CNNs, but will not be considered here.

Apart from our work [13], to the best of our knowledge, invariance to the sound level has not been addressed in the context of singing voice detection, but at least Mauch et al. [15] and Sturm [24, Sec. III.B] recognized it as a possible confounding factor for music information retrieval systems. In speech recognition, early approaches based on Mel-Frequency Cepstral Coefficients (MFCCs) discard the 0th coefficient [4, Eq. 1], effectively becoming invariant to the scale of the input signal. Modern CNN-based systems processing spectrograms or raw signals achieve robustness by using large networks and datasets (e.g., 38 million parameters and 7000 hours in [1]). For smaller CNNs, Wang et al. [26] recently proposed to process spectrograms with an automatic gain control of learnable parameters, termed per-channel energy normalization (PCEN). We will include this method in our experiments.

3. METHOD

In the following, we will describe the state-of-the-art method we used as a starting point, and five modifications aiming to reduce its sound level dependency (which was demonstrated in Figure 3).

3.1 Baseline

We base our work on the system of Schlüter et al. [22], in the variant they made available online¹ and described in [21, Sec. 9.8]. From monophonic input signals sampled at 22 kHz, it computes magnitude spectrograms (frame length 1024, hop size 315 samples), applies a mel filterbank (80 bands from 27.5 Hz to 8 kHz) and scales magnitudes as $\log(\max(10^{-7}, x))$. A CNN classifies 115-frame excerpts of these spectrograms into vocal/nonvocal. It starts with batch normalization [10] across the batch and time axis without learned scale and bias – this effectively standardizes each mel band over the training set as in [22], but can adapt to changes to the frontend during training,

¹ https://github.com/f0k/ismir2015/tree/phd_extra, accessed 2018-03-30

which we need for PCEN. This is followed by two convolutional layers of 64 and 32 3×3 filters, respectively, 3×3 max-pooling, 128 and 64 3×3 convolutions, 128 3×18 convolutions, 4×1 pooling, and three dense layers of 256, 64 and 1 units, respectively. Each convolutional and dense layer is followed by batch normalization and leaky rectification $\max(x/100, x)$ except for the final layer, which uses a sigmoid unit for binary classification.

During training, 50% dropout is applied before each fully-connected layer, and inputs are augmented with pitch shifting and time stretching up to $\pm 30\%$, and random frequency band filters of up to ± 10 dB, before mel scaling.

At test time, we turn the CNN into a fully-convolutional net, replacing dense layers by convolutions and adding dilations as described in [23]. This allows computing predictions over a full spectrogram without redundant computations that would occur when feeding overlapping 115-frame excerpts. All batch normalizations use statistics collected during training, not statistics from test examples.

3.2 Data Augmentation

A sure way to prevent classifiers from exploiting particular correlations in the training data is to remove these correlations from the data. Data augmentation attempts to remove or reduce correlations by varying the training examples along the confounding dimension. In our case, to reduce the dependency between input magnitude and target shown in Figures 1, 2, we scale input signals randomly by up to ± 10 dB in addition to the existing augmentations.

3.3 Instance Normalization

As a more drastic measure, we replace the initial batch normalization with instance normalization [25], i.e., we separately standardize each 115-frame excerpt to zero mean and unit variance per mel band, both at training and at test time. This is in contrast to batch normalization, which uses batch-wise rather than excerpt-wise statistics during training, and fixed dataset-wise statistics² for testing.

Instance normalization trivially results in a representation that is fully invariant to scaling the input signal. However, it prevents using the CNN as a fully-convolutional net at test time, since every excerpt needs to be processed separately. In Section 4.4, we will see how this affects computation time.

3.4 Spectral Delta Features

Scaling the input signal results in a shift of the logarithmic-magnitude mel spectrogram. Delta features, i.e., the elementwise difference between a frame and its predecessor, are invariant to such an offset. They are commonly used as supporting features to include temporal information in frame-wise classification, but have also been used successfully as the only input for RNN-based musical onset detection (albeit in a rectified form, [5]) and might be sufficient for singing voice detection.

² For simplicity, an exponential moving average of batch-wise statistics collected during training, as suggested for validation in [10, Sec. 3.1]. Importantly, the normalization is independent of the input at test time.

3.5 PCEN

Proposed by Wang et al. [26], per-channel energy normalization (PCEN) processes a mel spectrogram of linear magnitudes (i.e., replacing the logarithmic scaling) as

$$Y_{t,f} = \left(\frac{X_{t,f}}{(\epsilon + M_{t,f})^{\alpha_f}} + \delta_f \right)^{r_f} - \delta_f^{r_f}, \quad (1)$$

where M is an estimate of the local magnitude per time step and frequency band computed using a simple infinite impulse response (IIR) filter:

$$M_{t,f} = (1 - s_f)M_{t-1,f} + s_f X_{t,f} \quad (2)$$

The division by M implements an automatic gain control, which is followed by root compression (for $0 < r_f < 1$). Wang et al. parameterize $\alpha_f := \exp(\hat{\alpha}_f)$, $\delta_f := \exp(\hat{\delta}_f)$, $r_f := \exp(\hat{r}_f)$ and learn $\hat{\alpha}$, $\hat{\delta}$, \hat{r} as part of a neural network. Learning the logarithms ensures that α , δ , r remain positive. Instead of learning s , Wang et al. replace M with a convex combination of precomputed IIR filters of different smoothing factors s and learn the combination weights.

We deviate from their approach in two respects:

1. We fix $\alpha_f := 1$, as any other choice will make Y dependent on the scale of X .
2. We parameterize $s_f := \exp(\hat{s}_f)$ and learn \hat{s} directly as part of the neural network.³ Wang et al. noted that option in [26, Sec. 3], but did not explore it.

The IIR filter must process the input sequentially, and thus is not a good fit for massively parallel computation devices such as Graphical Processing Units (GPUs). We will see how this affects computation time in Section 4.4.

3.6 Zero-Mean Convolution

Spectral delta features are just one of many ways to compute differences in the spectrogram that are invariant to adding a constant to the input. For example, we could just as well compute differences between neighbouring frequencies. More generally, any cross-correlation with a zero-mean filter W will remove a global offset c from X :

$$\begin{aligned} ((X + c) * W)_{t,f} &= \sum_{i,j} (X_{t+i,f+j} + c) W_{i,j} \\ &= \sum_{i,j} X_{t+i,f+j} W_{i,j} + c \sum_{i,j} W_{i,j} = (X * W)_{t,f} \end{aligned}$$

The last step uses our assumption of a zero-mean filter, $\sum_{i,j} W_{i,j} = 0$. The first convolutional layer of our CNN already computes 64 separate cross-correlations of the input with learnable filters $W^{(k)}$, where k indexes the 64 filters. We enforce these to be zero-mean by parameterizing $W_{i,j}^{(k)} := \hat{W}_{i,j}^{(k)} - \frac{1}{MN} \sum_{i,j} \hat{W}_{i,j}^{(k)}$ and learning $\hat{W}^{(k)}$, where $M = N = 3$ is the filter size.

³ We could also use a sigmoid function to ensure $0 < s_f < 1$, but in practice, the bound $s_f < 1$ was not at a risk to be broken during learning.

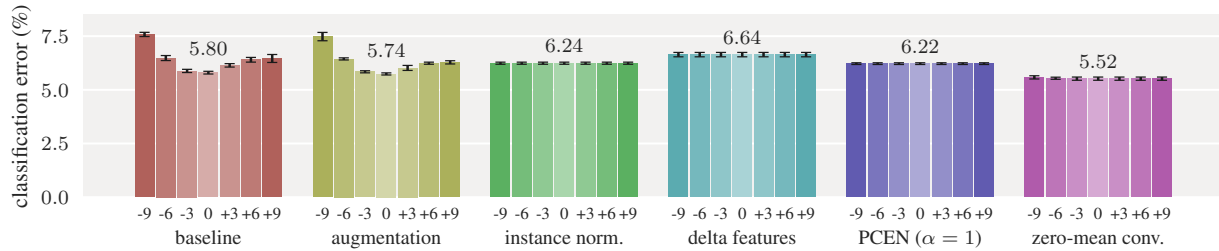


Figure 4: Classification error on our test set for each method with modified input gain between -9 dB to +9 dB. Error bars indicate the standard deviation over five networks. To facilitate comparison, the result at 0 dB is printed at the top.

4. EXPERIMENTS

To compare the five methods and the baseline, we trained and tested each of them on a large public singing voice detection dataset, comparing the quality of their predictions, robustness to level changes, and computational demands.

4.1 Dataset

For our previous work [13], we curated a dataset combining data from Jamendo [18], RWC [8, 16], MSD100 [17], a music video game, YouTube and several instrumental albums. Compared to existing corpora, it is larger and more diverse, both in terms of music genres and by including purely instrumental music pieces – it can be insightful to test a singing voice detection system on music that does not feature vocals as the predominant instrument (for example, Figures 3a,b show excerpts of two instrumental pieces).

In total, the dataset contains almost 80 h of music, split up (without artist overlaps) into 20 h for training, 17.5 h for validation, and 42 h for testing. For a more detailed listing, we refer the reader to [13, Table I].

4.2 Training

Networks are trained to minimize cross-entropy loss on mini-batches of 32 excerpts with ADAM [11]. Weights are initialized following Saxe et al. [19], PCEN parameters $\hat{\delta}_f$ and \hat{r}_f to zeros, \hat{s}_f to $\log(0.025)$, when used. Compared to the public implementation of the baseline system, we use an adaptive learning rate schedule to cope with the larger dataset. We start at a learning rate of 0.001 and drop it to a tenth whenever the training loss⁴ did not reach a new minimum for 10 consecutive mini-epochs of 1000 updates each. At each drop, we reset the network weights to the previous minimum. On the third drop, we stop training.

4.3 Evaluation

After training, we compute framewise predictions (network outputs between 0.0 and 1.0) for all validation and test recordings at their original sound level as well as all test recordings at gains of -9 dB, -6 dB, -3 dB, +3 dB, +6 dB, +9 dB.⁵ Each sequence of predictions is smoothed

⁴ We did not run into any overfitting, possibly because the network was originally designed for a much smaller dataset, and found it beneficial to base the schedule on the training loss rather than the validation loss.

⁵ Gains are applied to the input signal expressed as floating-point samples, so positive gains cannot result in clipping.

	Nvidia Titan Xp	Nvidia GTX 970	Intel i7-4770S
baseline	1.7 s	3.0 s	15.2 s
augmentation	1.7 s	3.0 s	15.2 s
instance norm.	42.5 s	103.1 s	643.1 s
delta features	1.7 s	3.0 s	15.2 s
PCEN	6.9 s	9.0 s	15.5 s
zero-mean conv.	1.7 s	3.0 s	15.2 s

Table 1: Computation time required for predicting singing voice in one hour of audio with each method, for two GPUs and a CPU (using a single core).

in time with a sliding median filter of 800 ms. We determine the optimal classification threshold for the smoothed predictions of the validation set at its original sound level, and apply this threshold to all other predictions. Finally, we compute the classification error for the test recordings, separately for each applied gain.

4.4 Results

Figure 4 depicts our results. The leftmost group of bars shows the classification error of the baseline system: It reaches 5.8% error for the original recordings, but performs worse when scaling the input signals, up to an error of 7.6% for -9 dB (a scale factor of $10^{-9/10} \approx 0.126$).

Training with examples of modified gain apparently does not help: Results at original sound level are comparable to the baseline, and the sound level dependency is as strong as before. Apparently, the augmentation does not sufficiently weaken the dependency between input magnitude and target label. Furthermore, it may not add anything over the existing frequency filtering augmentation, which applies a random gain to a random frequency range.

All remaining methods are invariant to an input gain by construction, so they achieve the same classification error regardless of the gain.⁶ In terms of accuracy, spectral delta features perform worst, at an error of 6.6%. Instance Normalization and PCEN (with fixed α_f parameters as explained in Section 3.5) are noticeably better, but still fall significantly behind the baseline system at 6.2% error.

⁶ Note that the converse is not true: a system achieving the same classification error for altered inputs may still be level-dependent, by improving for some examples and failing on others. In [13], we propose an evaluation scheme to rule out this case, but it is not needed here.

When not fixing α , PCEN reaches an error of 5.9% at 0dB, but is as level-dependent as the baseline, with learned α_f between 0.5 and 0.8 (results not included in Figure 4). Finally, zero-mean convolutions slightly exceed the classification accuracy of the baseline system while still being robust to level changes.

As an additional criterion, Table 1 compares the test-time computational demands of the different variants. Using the baseline system, computing framewise singing voice predictions for one hour of audio (with spectrograms already computed) takes 1.7 seconds with a high-end GPU, 3 seconds with a consumer GPU, and 15 seconds on a single CPU core. Since data augmentation and zero-mean convolutions only affect training, and since spectral delta features are cheap to compute, all three are just as fast as the baseline. The IIR filter of PCEN is inherently serial, hindering parallelization. This is not a problem in single-threaded CPU computation, but up to $4\times$ slower than the baseline on GPU. Finally, Instance Normalization requires processing each 115-frame network input separately, preventing reuse of computation in overlapping excerpts. While still fast enough for real-time processing, this poses a huge disadvantage, and is up to $42\times$ slower than the baseline.

5. CONCLUSION

After demonstrating that singing voice detectors are susceptible to partly base their prediction on the absolute magnitude of the input signal, we explore five different ways to reduce or eliminate this dependency in a CNN-based state-of-the-art system. They have different strengths and weaknesses, but one method turned out to be optimal in terms of classification error, robustness to level changes and computational overhead: parameterizing the filters of the first convolutional layer to be zero-mean. When processing logarithmic-magnitude spectrograms, this removes any constant offset resulting from changing the input gain.

Introducing level invariance with zero-mean convolutions is easy and does not measurably affect training time. This might be useful in other machine listening tasks that should not take the sound level into account – either to stabilize predictions against changes in the input gain, as in our case, or even to improve learning from data of varying loudness. To facilitate reuse, our implementation of all five methods is available online.⁷

A dissatisfying aspect of our solution is that it required understanding the problem and introducing a constraint in the parameter space of the neural network. While this is a reasonable way to make progress, it would be helpful to find a method that forces the network to learn this constraint from data. A possible candidate would be Unsupervised Domain Adaptation [7], although initial experiments did not turn out successful. Level-invariant singing voice detection might be a useful test bed, since we already know what a level-invariant CNN can look like.

⁷ <https://github.com/f0k/ismir2018> or http://jan-schlueter.de/pubs/2018_ismir.zip

In the broader context of the discussion on *horses* [24] (systems that rely on confounding factors for their predictions), our work identified a system to be a horse, and found a way to fix the aspect it identified. Most probably, the system is still partly using the wrong cues, and future work could iteratively find and fix this. However, this may not be the best road to follow: both finding and avoiding confounds is difficult. We discovered the loudness confound after noticing that including the 0th MFCC in the feature set of a classifier unexpectedly improved results, following this trail by testing classifiers with altered examples. Avoiding it required very different approaches for a hand-designed feature set [13] and the CNN addressed here. Another confound, a hypersensitivity of our system to sloped lines in a spectrogram, was discovered by looking at false negatives and false positives, but attempts to avoid it were fruitless [21, p. 190]. A different angle of attack on horses would be to research ways to constrain the learning system to mimic human perception, such that it cannot use cues that humans would not consider in the first place.

6. ACKNOWLEDGEMENTS

This research is supported by the Vienna Science and Technology Fund (WWTF) under grants NXT17-004 and MA14-018. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of two Tesla K40 GPUs and a Titan Xp GPU used for this research. Last, but not least, we would like to thank the anonymous reviewers for their valuable input.

7. REFERENCES

- [1] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. C. Catanzaro, et al. Deep Speech 2: End-to-end speech recognition in english and mandarin. *arXiv e-prints*, abs/1512.02595, 2015.
- [2] A. L. Berenzweig and D. P. W. Ellis. Locating singing voice segments within music signals. In *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 119–122, New Paltz, NY, USA, October 2001.
- [3] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Canam, and J. P. Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, October 2014.
- [4] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, August 1980.
- [5] F. Eyben, S. Böck, B. Schuller, and A. Graves. Universal onset detection with bidirectional long short-term memory neural networks. In *Proceedings of the 11th*

- International Society for Music Information Retrieval Conference (ISMIR)*, pages 589–594, Utrecht, Netherlands, August 2010.
- [6] B. Friedman and H. Nissenbaum. Bias in computer systems. *ACM Transactions on Information Systems*, 14(3):330–347, July 1996.
 - [7] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 1180–1189, Lille, France, July 2015. PMLR.
 - [8] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical, and jazz music databases. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*, pages 287–288, Paris, France, October 2002.
 - [9] E. J. Humphrey, N. Montecchio, R. Bittner, A. Jansson, and T. Jehan. Mining labeled data from web-scale collections for vocal activity detection in music. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, October 2017.
 - [10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, July 2015. PMLR.
 - [11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015.
 - [12] S. Leglaive, R. Hennequin, and R. Badeau. Singing voice detection with deep recurrent neural networks. In *Proceedings of the 40th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 121–125, Brisbane, Australia, April 2015.
 - [13] B. Lehner, J. Schlüter, and G. Widmer. Online, loudness-invariant vocal detection in mixed music signals. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(8):1369–1380, August 2018.
 - [14] B. Lehner, G. Widmer, and S. Böck. A low-latency, real-time-capable singing voice detection method with LSTM recurrent neural networks. In *Proceedings of the 23rd European Signal Processing Conference (EU-SIPCO)*, pages 21–25, Nice, France, August 2015.
 - [15] M. Mauch and S. Ewert. The audio degradation toolbox and its application to robustness evaluation. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, pages 83–88, Curitiba, Brazil, November 2013.
 - [16] M. Mauch, H. Fujihara, K. Yoshii, and M. Goto. Timbre and melody features for the recognition of vocal activity and instrumental solos in polyphonic music. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 233–238, Miami, FL, USA, October 2011.
 - [17] N. Ono, Z. Rafii, D. Kitamura, N. Ito, and A. Liutkus. The 2015 signal separation evaluation campaign. In *International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, pages 387–395, Liberec, France, August 2015.
 - [18] M. Ramona, G. Richard, and B. David. Vocal detection in music with support vector machines. In *Proceedings of the 33rd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1885–1888, Las Vegas, NV, USA, March 2008.
 - [19] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, Banff, Canada, April 2014.
 - [20] J. Schlüter. Learning to pinpoint singing voice from weakly labeled examples. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, NY, USA, August 2016.
 - [21] J. Schlüter. *Deep Learning for Event Detection, Sequence Labelling and Similarity Estimation in Music Signals*. PhD thesis, Johannes Kepler University Linz, Austria, July 2017.
 - [22] J. Schlüter and T. Grill. Exploring data augmentation for improved singing voice detection with neural networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, October 2015.
 - [23] T. Sercu and V. Goel. Dense prediction on sequences with time-dilated convolutions for speech recognition. In *NIPS Workshop on End-to-end Learning for Speech and Audio Processing*, Barcelona, Spain, November 2016.
 - [24] B. L. Sturm. A simple method to determine if a music information retrieval system is a “horse”. *IEEE Transactions on Multimedia*, 16(6):1636–1644, October 2014.
 - [25] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv e-prints*, abs/1607.08022, July 2016.
 - [26] Y. Wang, P. Getreuer, T. Hughes, R. F. Lyon, and R. A. Saurous. Trainable frontend for robust and far-field keyword spotting. In *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5670–5674, March 2017. arXiv:1607.05666.

MUSIC GENERATION AND TRANSFORMATION WITH MOMENT MATCHING-SCATTERING INVERSE NETWORKS

Mathieu Andreux and Stéphane Mallat

Département d'informatique de l'ENS, École normale supérieure,
CNRS, PSL Research University, 75005 Paris, France

ABSTRACT

We introduce a Moment Matching-Scattering Inverse Network (MM-SIN) to generate and transform musical sounds. The MM-SIN generator is similar to a variational autoencoder or an adversarial network. However, the encoder or the discriminator are not learned, but computed with a scattering transform defined from prior information on sparse time-frequency audio properties. The generator is trained by jointly minimizing the reconstruction loss of an inverse problem, and a generation loss which computes a distance over scattering moments. It has a similar causal architecture as a WaveNet and provides a simpler mathematical model related to time-frequency decompositions. Numerical experiments demonstrate that this MM-SIN generates new realistic musical signals. It can transform low-level musical attributes such as pitch with a linear transformation in the embedding space of scattering coefficients.

1. INTRODUCTION

This paper investigates musical sound generation and transformation with a simplified algorithmic architecture, which relates generative networks to time-frequency representations. Image generation has led the way through the development of Generative Adversarial Networks (GANs) [7] and Variational Autoencoders [13] where images are generated from a Gaussian white noise vector, which defines a latent space. Arithmetic operations in this latent space lead to controlled transformations over the images such as aging of faces or transforming women in men. The problem is however different for audio signals which must take into account time causality properties. Some authors have applied image generation algorithms over spectrograms [3, 10] but it then requires to invert the spectrograms with a vocoder or a Griffin-Lim algorithm which is long and has a reduced quality.

Deep autoregressive neural networks such as WaveNet [15, 16] or SampleRNN [14] have achieved exceptional synthesis of music and speech signals. They

do not take as input a Gaussian white noise but estimate a probability distribution with a Markov chain factorization, which computes conditional probabilities given from past values. The generated signals have an outstanding quality but these neural architectures are complex and lack interpretability. Several effective techniques have been introduced to modify audio generations [4–6, 9] by modifying the latent code to change the probability distribution or by using reference signals as targets, which is more complex than arithmetic operations used for images.

This paper introduces a simplified neural network architecture which synthesizes and modifies music signals from Gaussian white noise, with two key contributions. As opposed to image GANs or variational autoencoders, we do not learn a discriminator or an encoder: both are provided by prior information on audio signals, which is captured by their time-frequency regularity. This is done by adapting a result obtained in [2] for images, and introducing a moment matching technique. The second contribution is the introduction of a causal computational architecture allowing to progressively synthesize audio signals, and which can be parallelized in GPU's. The resulting Scattering Autoencoder architecture has some similarities with a Parallel WaveNet [16]. Similarly to warping properties over images, we show that arithmetic transformations in the latent space produce time-frequency deformations, which can modify the pitch of musical notes or interpolate music. Despite a lower synthesis quality than state-of-the-art generating methods, these preliminary results pave the way for a new approach to synthesize audio signals, without learning encoders or discriminators.

2. SCATTERING AUTOENCODER

This section introduces the principles of a scattering autoencoder and its computational architecture. The encoder is not learned but computed based on prior information on audio time-frequency properties. Audio and musical signals have sparse representations over time-frequency dictionaries such as audio wavelets [20]. Their perceptual properties are not much affected by small time-frequency warpings. We define a signal embedding which takes advantage of these characteristics.

The architecture of a scattering autoencoder is illustrated in Figure 1. The random input audio signal $X[t]$ is first transformed into a nearly Gaussian random vec-



© Mathieu Andreux and Stéphane Mallat. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: Mathieu Andreux and Stéphane Mallat. "Music Generation and Transformation with Moment Matching-Scattering Inverse Networks", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

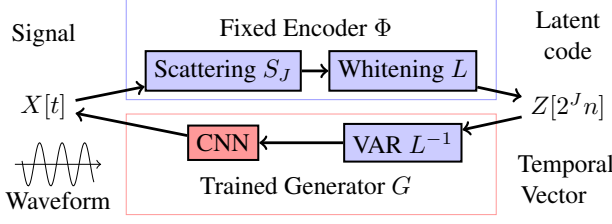


Figure 1. The audio waveforms is encoded with a time-frequency scattering S_J followed by a linear whitening operator L . The Scattering Inverse Network (SIN) G restores a signal from a Gaussian white noise by inverting $L S_J$ on training data.

tor $S_J(X)[2^J n]$ by applying the time-frequency scattering transform S_J , where $J \geq 0$ is a hyperparameter. Gaussianization is achieved through a time averaging over a sufficiently long time interval thanks to the central limit theorem. In order to preserve enough information on the original signal X after averaging, multiple sparse time-frequency channels are built by applying iterative wavelet transforms [1]. The Gaussian scattering $S_J(X)$ is then mapped into a Gaussian white noise with a whitening operator L , which outputs the linear prediction errors of a future scattering vector from its neighboring past. Section 4 details the whitened scattering transformation LS_J .

The generator G inverts the linear whitening operator and the joint scattering transform by synthesizing an approximation of $X[t]$ from $Z[2^J n]$. It first applies a vector autoregressive (VAR) filter L^{-1} , which can be deduced from L . This operation is followed by a causal convolutional neural network (CNN), with the convolutions acting along time. The network has J layers to invert a scattering at scale 2^J . Section 3 describes the architecture which has similarities with a Parallel WaveNet [16]. The optimization of the generator G amounts to inverting the scattering transform S_J in an adapted metric. The statistics of synthesized signals are constrained by ensuring that they have the same moments in the scattering space as the input signal $X[t]$. This network is thus called a Moment Matching-Scattering Inverse Network (MM-SIN).

3. MOMENT MATCHING-SCATTERING INVERSE GENERATOR

A Moment Matching-Scattering Inverse Network G is computed by inverting a scattering embedding computed at a scale 2^J . It is a causal network which takes as input a nearly Gaussian white noise vector Z computed with a scattering transform, to recover an estimation of the signal X . This network is trained with a loss which incorporates the inverse problem loss computed with a scattering metric, regularized by a moment matching term that can be interpreted as a discriminative metric.

The MM-SIN generator is a linear recurrent neural network L^{-1} followed by a causal convolutional network implemented with a cascade of J convolutions and pointwise non-linearities, illustrated in Figure 2. Each intermediate

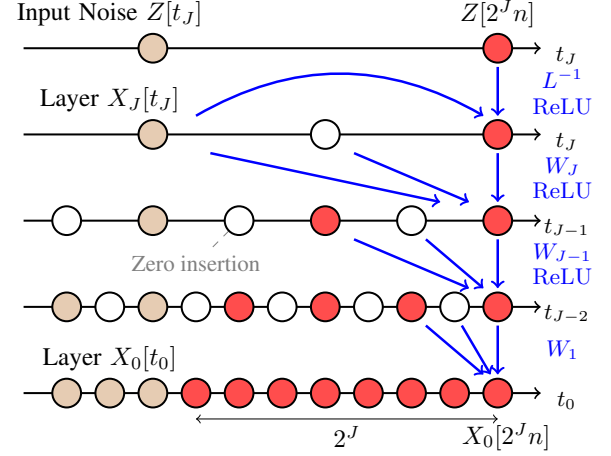


Figure 2. An MM-SIN is a linear recurrent network followed by a causal deep convolutional network with J layers. It takes as input a vector of Gaussian white noise $Z[2^J n]$ (top right, red), and computes the corresponding scattering vector $X_J[2^J n]$ by applying L^{-1} . Intermediate layers $X_j[t_j]$ are then computed with causal convolutions denoted by blue arrows and zero insertions (white points). The single vector $Z[2^J n]$ outputs 2^J values for $X_0[t_0]$, marked with red points.

network layer is composed of vectors $X_j[t_j]$ having k_j channels and sampled at intervals 2^j , for $0 \leq j \leq J$. All layers in Figure 2 appear to be aligned but to understand the causality structure one must realize that each layer is indexed by a time index t_j which is shifted by 2^j relatively to the absolute time variable t of the original input signal $X[t]$: $t_j = t - 2^j$. Using the absolute time t , we thus use the noise vector at a time $t = 2^J(n+1)$ to generate 2^J new output signal values at times $2^J n - 2^J + 1 < t \leq 2^J n + 1$.

The first layer maps $Z[t_J]$ to $X_J[t_J]$ with a vector autoregressive filter L^{-1} which inverts the whitening operator L , followed by a ReLU non-linearity $\rho(u) = \max(u, 0)$

$$X_J = \rho(L^{-1}Z). \quad (1)$$

A new noise vector $Z[2^J n]$ outputs a new vector $X_J[2^J n]$. At depth j , this initial vector gives rise to 2^{J-j} temporal vectors $X_j[t_j]$ for $2^J n - 2^{J-j} < t_j \leq 2^J n$.

At layer $j > 0$, the layer X_j is mapped to X_{j-1} with an *à trous* convolution followed by a ReLU non-linearity plus bias. We first double the size of X_j with a zero insertion:

$$\tilde{X}_j[n2^j] = X_j[n2^j] \text{ and } \tilde{X}_j[n2^j + 2^{j-1}] = 0. \quad (2)$$

Each X_{j-1} is then calculated with a causal convolution along time and a linear operator along channels with a bias, W_j , followed by a ReLU:

$$X_{j-1} = \rho(W_j \tilde{X}_j). \quad (3)$$

Except for the autoregressive layer, the ReLU is preceded by a batch normalization [11]. The last convolution is not followed by a ReLU so that we can output a

signal with negative values. The final output is $X_0[t_0]$ for $2^J n - 2^J < t_0 \leq 2^J n$ which corresponds to $X[t]$ for $2^J n - 2^J + 1 < t \leq 2^J n + 1$. As in standard generative networks [18], the number of the channels k_j decreases with j according to a geometric law fitted such that X_0 has only one channel and X_J has k_J channels, which is the dimensionality of each vector $Z[2^J n]$.

The parameters of the network G are optimized by inverting the scattering transform followed by the causal whitening operator L . The loss \mathcal{L} minimized by G is a sum of two terms, weighted by a hyperparameter $\lambda > 0$. The first term \mathcal{L}_{inv} measures the accuracy of the inversion. The second discriminative term \mathcal{L}_{MM} measures the distance between the scattering moments of the synthesized signals and the scattering moments of the original signals.

$$\min_G \mathcal{L} = \mathcal{L}_{\text{inv}} + \lambda \mathcal{L}_{\text{MM}}. \quad (4)$$

The inverse problem loss \mathcal{L}_{inv} computes the reconstruction error on each training example x_i from its embedding z_i computed with the scattering transform S_J followed by the linear whitening operator L . The reconstruction error is calculated over scattering coefficients computed at a scale $2^K < 2^J$:

$$\mathcal{L}_{\text{inv}} = \frac{1}{N} \sum_{i=1}^N \|S_K(x_i) - S_K G(z_i)\|_1 \quad \text{with } z_i = L S_J(x_i). \quad (5)$$

The l^1 norm promotes the sparsity of the responses, while the scattering S_K allows to generate signals which may be locally deformed, but are perceptually similar to the original ones. In order to avoid useless computations, we do not apply $L^{-1} z_i$ but directly input the vectors $S_J(x_i)$ to the convolutional part of G for the computation of \mathcal{L}_{inv} .

The loss \mathcal{L}_{inv} does not control the quality of the generated samples $G(z)$ when z is sampled as a Gaussian white noise. Similarly to GANs which have a discriminator, the quality is controlled by introducing another loss term \mathcal{L}_{MM} , which controls the distance between the generated distribution and the distribution of the original signals.

The moment matching term \mathcal{L}_{MM} computes the distance between scattering coefficients of generated signals averaged over time t and batch index i , $\overline{S_K G(z_i)[t]}$, and scattering coefficients of the training signals averaged over time t and training examples i , $\overline{S_K x_i[t]}$:

$$\mathcal{L}_{\text{MM}} = \left\| \overline{S_K x_i[t]} - \overline{S_K G(z_i)[t]} \right\|^2 \quad (6)$$

The codes $\{z_i\}$ correspond to a batch of random vectors which is renewed at each iteration of the gradient descent algorithm.

The loss \mathcal{L}_{MM} is similar to the Maximum Mean Discrepancy regularization introduced in [19]. The moment matching term can be interpreted as a distance with a scattering transform kernel [8]. However, in this case it can directly be implemented as a difference of moments.

4. WHITENED TIME-FREQUENCY SCATTERING

This section details the time-frequency scattering transform $S_J(X)$ originally introduced in [1] and its whiten-

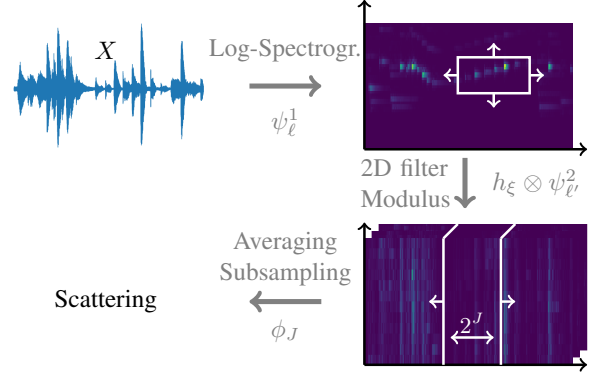


Figure 3. Time-Frequency Scattering transform. The log-spectrogram is obtained with a first wavelet transform ψ_ℓ^1 followed by a modulus. A joint time-frequency filtering of this log-spectrogram with the filters $h_\ell \otimes \psi_\ell^2$ regularizes the time-frequency deformations of the signal. The low-pass convolution with ϕ_J Gaussianizes the resulting tensor.

ing L , which results in the embedding $Z[2^J n]$. Figure 3 sketches the different computational steps. This transform relies on priors on musical signals in order to build a time-dependent vector representation $S_J X[2^J n]$ which is approximately Gaussian and linearizes small time-frequency deformations.

Musical signals admit a sparse decomposition in time-frequency representations with a spectrogram. Here, we first compute a spectrogram with frequencies sampled on a logarithmic scale thanks to a wavelet filterbank $\{\psi_\ell^1\}_{0 \leq \ell < J}$ followed by a modulus non-linearity. The wavelets ψ_ℓ^1 are defined by dilations of a single mother wavelet:

$$\psi_\ell^1[t] = 2^{-\ell/Q} \psi^1[2^{-\ell/Q} t] \quad (7)$$

We use causal analytic Gammatone wavelets [20], which are good perceptual models of auditory filters, with $Q = 12$ wavelets per octave in order to separate high-frequency partials.

On this sparse spectrogram, small time-frequency deformations of the input signal X produce small local translations in the time-frequency plane. These deformations result in smooth perceptual variations. As a consequence, the embedding should be regular with respect to these deformations. This is obtained with a joint 2D filtering of the spectrogram in the time and log-frequency axis. One can prove that the resulting representation is Lipschitz-continuous to these deformations, while preserving invertibility thanks to the use of filters spanning all the energy of the signal [1]. This will imply a form a linearization of these deformations, paving the way for meaningful arithmetic in the latent space.

The time-frequency filters are built as a separable product $h_\ell \otimes \psi_\ell^2$ of frequential filters h_ℓ and temporal filters ψ_ℓ^2 . The frequential filters h_ℓ are localized Fourier atoms with a Hann window whose size P matches one octave, $P = Q = 12$. The convolution is computed in half-overlaps over the frequency axis. The temporal filters ψ_ℓ^2

are also Gammatone wavelets, with only $Q_2 = 1$ wavelet per octave. After performing the convolution, a modulus non-linearity is also applied in order to remove the local phase, thereby regularizing the representation.

The time-frequency filtering of the log-spectrogram results in a large tensor with one temporal axis. Its coefficients are sparse along the channel axis and typically decorrelate as they get farther apart in time. To obtain a variable which is more Gaussian, we use the central limit theorem which says that the averaging of a large number of independant variables converges towards a Gaussian random variable. We perform this averaging with a window size 2^J which should be larger than the typical decorrelation length in order to average over enough independent events. This averaging is carried out with a low-pass filter ϕ_J along the temporal axis. It is followed by a subsampling by 2^J in order to remove redundant information.

The time-frequency scattering transform $S_J X$ is defined as:

$$S_J X[2^J n] = \left[x \star \phi_J[2^J n], |x \star \psi_\ell^1| \star h_\xi \star \phi_J[2^J n], \right. \\ \left. |x \star \psi_{\ell'}| \star (h_\xi \otimes \psi_{\ell'}^2) \star \phi_J[2^J n] \right], \quad (8)$$

for all ℓ, ξ, ℓ' , where convolutions with h_ξ should be interpreted along the frequential ℓ axis and other convolutions along time. $S_J X$ is subsampled in time by a factor 2^J . Each vector $S_J X[2^J n]$ has dimension $k_J = 1 + Q(2J - 3) + Q(J - 2)^2$. For $J = 10$ and $Q = 12$, this amounts to 973 channels. The scale 2^J is chosen as a trade-off between the Gaussianization condition which improves as J increases, and the stability of the scattering invertibility which improves when J decreases.

Thanks to the local averaging, the vectors $S_J X$ tend to a Gaussian distribution. However, this distribution might not be white, *i.e.* it has temporal and channel-wise correlations. The whitening operator L is a causal vector autoregressive linear filter which removes this correlation structure. It is trained by minimizing a prediction error of $S_J X[2^J n]$ given previous vectors $S_J X[2^J(n - m)]$ for $1 \leq m \leq M$. The whitening operator L outputs the innovations of the fitted vector autoregressive process $\{Z[2^J n]\}_n$, which have an approximately Gaussian white distribution.

5. NUMERICAL EXPERIMENTS

We show that the MM-SIN is able to reconstruct waveforms from their embeddings and generate new decent waveforms from noise. Furthermore, we show that it is possible to manipulate low-level attributes of sounds such as pitch with a simple arithmetic in the embedding. In addition, a simple arithmetic in the embedding allows to merge the contents of different inputs, while preserving the musical structure of the resulting signal.

5.1 Methods

We describe the numerical details which lay out experiments. The source code supporting experiments is freely

available at <http://github.com/AndreuxMath/ismir2018>, where the reader may also find the audio recordings corresponding to the figures.

The time-frequency scattering transform S_J is computed with an averaging window of size $2^J = 2^{10} = 1024$. It is implemented on GPU with a code inspired from [17]. In the case of the loss (4), we employ a first-order scattering S_K , which means that it is an averaged scalogram, with an averaging window of size $2^K = 2^5 = 32$. The filterbanks are normalized so as to have responses of average equal magnitude in each band over the training dataset.

The architecture of the SIN G is defined as follows. The whitening operator L has a past size $M = 4$. All subsequent convolutions have a kernel size equal to 7. These values were not tuned: results could likely be improved with a careful hyperparameter search. Each network is trained by Adam [12] with a learning rate of 5×10^{-4} for 1200 epochs and batches of size 128.

We use two different musical datasets: NSynth [5] and Beethoven [14]. NSynth is a dataset consisting of annotated musical notes from multiple instruments, thereby allowing to perform carefully controlled transformation experiments. All recordings begin with the onset of the note and last 4s. We restrict ourselves to two types of acoustic instruments, keyboards and flutes, totalling 40 different instruments with MIDI pitches ranging 20 – 110, leading to a varied and well-balanced dataset. In the original dataset, instruments of the training and testing sets do not overlap. In this paper, we use an alternative split based on the velocity's attributes of the training samples: for each instrument, a random velocity is picked to define the test set. We only use the first 2s of the recordings, as they concentrate most of the energy of the signals.

The Beethoven dataset is closer to an actual musical composition than NSynth, insofar as it consists in 8s extracts of Beethoven's piano sonata. Therefore, it is a good testbed for music generation experiments. We use the train-test split provided by the authors.

For both datasets, the amplitudes of all recordings are normalized in $[-1, 1]$. The sampling rate is reduced from 16000Hz to 4096Hz so as to reduce the computational complexity. It is very likely that the quality of the synthesis could be improved by increasing this sampling rate.

5.2 Waveform generation

We first show that the MM-SIN generator is able to reconstruct and to generate realistic musical samples.

In Figure 4, we display two reconstructions of waveforms from their embeddings, along with the corresponding log-spectrograms, for each of the studied datasets. This shows that the network is able to generalize to a test set, and to adapt to the specifics of a given dataset. The reconstruction is not perfect. The network can introduce small time-frequency deformations because of the scattering encoder and the use of a scattering loss. As witnessed in the log-spectrograms, the time-frequency content of the signals is correctly retrieved, and perceptually the two signals sound similar, up to minor artifacts.

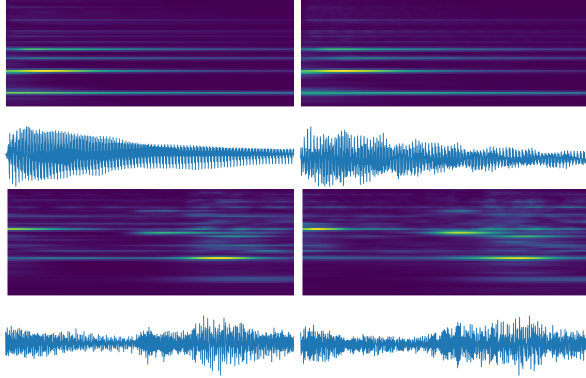


Figure 4. Reconstruction with the MM-SIN G . Left: Original test sample X . Right: Reconstruction $G(Z)$ for $Z = L S_J X$. Top: NSynth dataset. Bottom: Beethoven dataset. A different network was trained for each dataset.

Table 1 provides quantitative reconstruction results on the Beethoven dataset. Training a network with the mean-square error (MSE) metric $\|x_i - G(z_i)\|_2^2$ instead of the proposed perceptual metric within the inverse problem loss (5) negatively impacts results, both on the training and testing sets. Further, the moment-matching term has a positive effect on the reconstruction: even though it degrades reconstruction on the training set, it improves the generalization on the test set both in absolute and relative terms.

We now investigate the ability of the network to generate new waveforms from Gaussian white noise. Figure 5 displays several samples generated from white noise through a network trained on the Beethoven dataset. Despite the input being a pure white noise, the network is able to generate samples which alternate silences and more active phases. Further, the fundamental frequency which is played varies through the samples.

In order to measure the variability of the generated samples, we measure the spread σ of the distribution of the time-averaged scattering coefficients $S_K X$ of the samples. This spread corresponds to the average Euclidean distance between the time-averaged scattering of the waveforms and the average scattering coefficients of this distribution. In the case of the training distribution, we obtained

Loss	\mathcal{L}_{inv}	\mathcal{L}_{inv}	$\mathcal{L}_{\text{inv}} + \lambda \mathcal{L}_{\text{MM}}$
Metric	MSE	S_K	S_K
Train error	0.56	0.16	0.23
Test error	0.77	0.37	0.31
Gap test/train (dB)	1.36	3.53	1.21

Table 1. Reconstruction errors on the Beethoven dataset, expressed in terms of the perceptual loss (5). MSE denotes the mean-square error metric $\|x_i - G(z_i)\|_2^2$. Using the perceptual metric for training instead of the MSE metric reduces the error. Further, adding the moment-matching term during training improves the reconstruction results and the generalization.

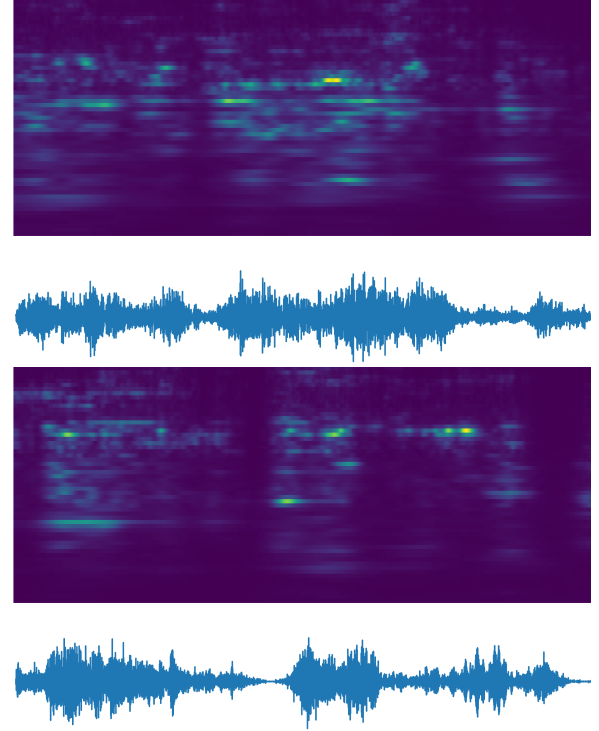


Figure 5. Musical signal $G(Z)$ generated from a white noise Z , where G is learned on the Beethoven dataset. Each line corresponds to an independent sample obtained from a different white noise realization. The resulting signals last about 4s.

$\sigma = 6.69$, whereas $\sigma = 3.51$ for the distribution generated from white noise. This shows that the generated samples exhibit a non-negligible variability, even though it is lower than the one expressed in the training set.

The effect of the moment matching loss (6) on the generated samples is difficult to assess qualitatively, so we resort to quantitative measures. In the case of a network trained without this loss, the moment matching distance between generated samples and the training set was equal to 38.7, whereas the same distance was equal to 0.176 when also optimizing this loss. As a comparison, the testing set has a distance of 0.334 with respect to the training set. Thus, using this loss term brings generated samples much closer to the natural signals' statistics.

5.3 Pitch modification

We now study the ability of the algorithm to transform the pitch of musical signals with an arithmetic operations in the latent space. We use the NSynth dataset, whose careful construction allows to perform modifications with fixed factors of variability. In the test set, we pick two samples belonging to the same instrument, but with a pitch separated by 5 MIDI scales. We compute their embeddings Z_1 and Z_2 , their mean embedding $(Z_1 + Z_2)/2$, and reconstruct the corresponding signals with the generator: $G(Z_1)$, $G(Z_2)$ and $G((Z_1 + Z_2)/2)$.

The results are displayed in Figure 6. The interpolation

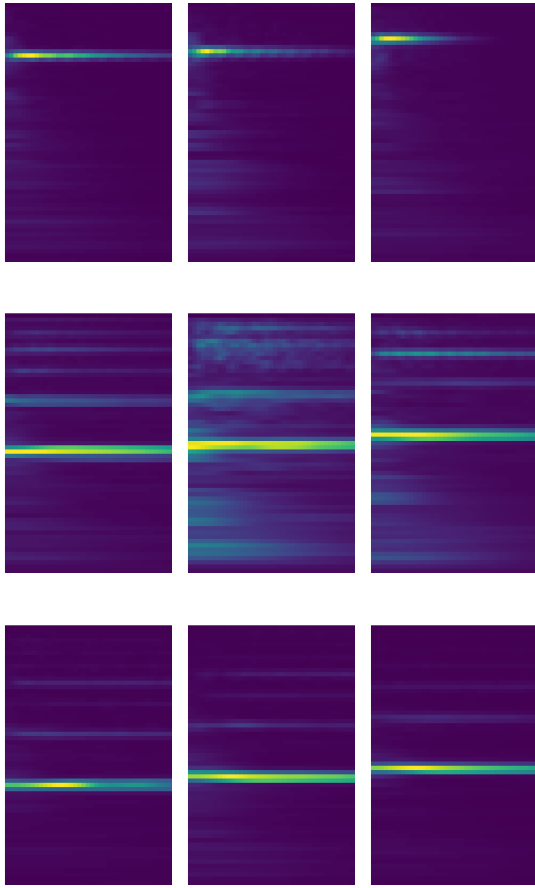


Figure 6. Pitch interpolation. Left column: $G(Z_1)$. Middle column: $G((Z_1 + Z_2)/2)$. Right column: $G(Z_2)$. Z_1 and Z_2 are the embeddings of samples from the test set. The generator interpolates the fundamental frequency with a simple arithmetic. The frequential displacement from left to right corresponds to 5 MIDI scales.

in the latent space does not result in a linear interpolation which would double the number of harmonics. It yields one fundamental frequency in each case. Furthermore, this fundamental frequency is indeed interpolated by this simple arithmetic. Observe that this is also the case of the partials, as can be seen in particular in the bottom example. However, this interpolation suffers from some artifacts. For instance, in the middle example, the partials at highest frequencies are cluttered and the resulting signal misses harmonicity. Yet, these results showcase the ability to transform signals via linear interpolations in the latent space with a simple unsupervised learning procedure and a predefined embedding.

The algorithm owes the ability to perform such pitch interpolations to the time-frequency scattering transform S_J used as an encoder, which regularizes small time-frequency deformations. As such, the pitch interval on which the interpolations can be performed is bounded by the size P of the Hann window used to filter the scalogram along the frequency axis.

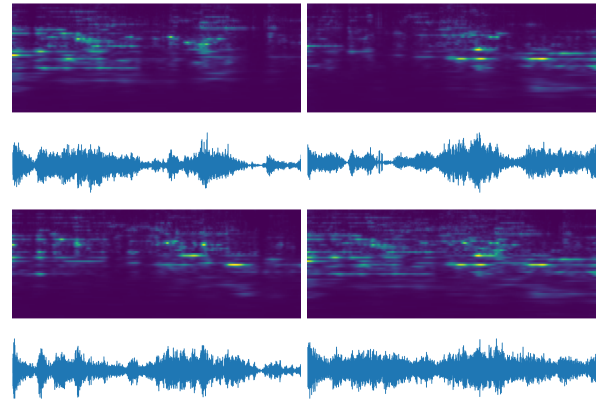


Figure 7. Interpolations in the latent and signal space. Top two signals: $G(Z_1)$ and $G(Z_2)$, where Z_1, Z_2 are Gaussian white noise realizations and G is trained on the Beethoven dataset. Bottom left: Latent interpolation $G((Z_1 + Z_2)/2)$. Bottom right: $(G(Z_1) + G(Z_2))/2$. The latent interpolation is able to merge both signals while preserving the musical structure.

5.4 Waveform interpolation

Let us show results when interpolating waveforms from the Beethoven dataset, which have a high density of musical events. We take two random white noise realizations Z_1 and Z_2 , and compare the effect on the waveforms of an interpolation in the latent space and in the signal space, with a network G trained on the Beethoven dataset.

The results are represented in Figure 7. The top two signals are the original signals $G(Z_1)$ and $G(Z_2)$, while the bottom left is the latent interpolation $G((Z_1 + Z_2)/2)$ and the bottom right the linear interpolation $(G(Z_1) + G(Z_2))/2$. The latent interpolation incorporates patterns from both signals but it respects the musical structure. It has successive musical notes with their harmonics, and it recovers a sound with silences. On the opposite, the linear interpolation merges both signals, which eliminates the silence regions while producing a cluttered log-spectrogram.

6. CONCLUSION

This paper introduces a causal musical synthesis network optimized through an inverse problem and which thus involves no learned encoder or discriminator. The encoder is defined from time-frequency signal priors in order to Gaussianize the input signal. The generator network maps back the resulting codes to raw waveforms. This network inverts the encoder and generates new signals whose scattering moments match those of the original signals. The resulting system synthesizes new realistic musical signals and performs the transformation of low-level attributes, such as pitch, by simple linear combinations in the latent space.

Synthesized signals do not reach the quality of state-of-the-art generating architectures but these first results show that this approach is a new promising avenue to synthesize audio signals directly from Gaussian white noise, without learning encoders or discriminators.

7. ACKNOWLEDGEMENTS

This work is supported by the ERC InvariantClass grant 320959 and an AMX grant from the MNEsR.

8. REFERENCES

- [1] J. Anden, V. Lostanlen, and S. Mallat. Joint time-frequency scattering for audio classification. In *Proc. of IEEE MLSP*, 2015.
- [2] Tomàs Angles and Stéphane Mallat. Generative networks as inverse problems with scattering transforms. In *International Conference on Learning Representations*, 2018.
- [3] M. Blaauw and J. Bonada. Modeling and transforming speech using variational autoencoders. In *Interspeech*, pages 1770–1774, 2016.
- [4] J. Chorowski, R.J. Weiss, R. A. Saurous, and S. Bengio. On using backpropagation for speech texture generation and voice conversion. In *International Conference on Audio and Speech Processing (ICASSP)*, 2018.
- [5] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan. Neural audio synthesis of musical notes with WaveNet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1068–1077, 2017.
- [6] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015.
- [7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
- [8] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample problem. In *Advances in neural information processing systems*, pages 513–520, 2007.
- [9] E. Grinstein, N. Duong, A. Ozerov, and P. Pérez. Audio style transfer. *HAL preprint hal-01626389*, 2017.
- [10] W.-N. Hsu, Y. Zhang, and J. Glass. Learning latent representations for speech generation and transformation. In *Interspeech*, pages 1273–1277, 2017.
- [11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.
- [12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [13] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.
- [14] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio. SampleRNN: An unconditional end-to-end neural audio generation model. In *International Conference on Learning Representations*, 2017.
- [15] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [16] A. Van Den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, et al. Parallel WaveNet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433*, 2017.
- [17] E. Oyallon, E. Belilovsky, and S. Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *Proc. of ICCV*, 2017.
- [18] A. Radford, L. Metz, and R. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.
- [19] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.
- [20] A. Venkitaraman, A. Adiga, and C. S. Seelamantula. Auditory-motivated gammatone wavelet transform. *Signal Processing*, 94:608–619, 2014.

WAVE-U-NET: A MULTI-SCALE NEURAL NETWORK FOR END-TO-END AUDIO SOURCE SEPARATION

Daniel Stoller

Queen Mary University of London

d.stoller@qmul.ac.uk

Sebastian Ewert

Spotify

sewert@spotify.com

Simon Dixon

Queen Mary University of London

s.e.dixon@qmul.ac.uk

ABSTRACT

Models for audio source separation usually operate on the magnitude spectrum, which ignores phase information and makes separation performance dependant on hyperparameters for the spectral front-end. Therefore, we investigate end-to-end source separation in the time-domain, which allows modelling phase information and avoids fixed spectral transformations. Due to high sampling rates for audio, employing a long temporal input context on the sample level is difficult, but required for high quality separation results because of long-range temporal correlations. In this context, we propose the Wave-U-Net, an adaptation of the U-Net to the one-dimensional time domain, which repeatedly resamples feature maps to compute and combine features at different time scales. We introduce further architectural improvements, including an output layer that enforces source additivity, an upsampling technique and a context-aware prediction framework to reduce output artifacts. Experiments for singing voice separation indicate that our architecture yields a performance comparable to a state-of-the-art spectrogram-based U-Net architecture, given the same data. Finally, we reveal a problem with outliers in the currently used SDR evaluation metrics and suggest reporting rank-based statistics to alleviate this problem.

1. INTRODUCTION

Current methods for audio source separation almost exclusively operate on spectrogram representations of the audio signals [6, 7], as they allow for direct access to components in time and frequency. In particular, after applying a short-time Fourier transform (STFT) to the input mixture signal, the complex-valued spectrogram is split into its magnitude and phase components. Then only the magnitudes are input to a parametric model, which returns estimated spectrogram magnitudes for the individual sound sources. To generate corresponding audio signals, these magnitudes are combined with the mixture phase and then converted with an inverse STFT to the time domain. Optionally, the phase can be recovered for each source individually using the Griffin-Lim algorithm [5].

This approach has several limitations. Firstly, the STFT output depends on many parameters, such as the size and overlap of audio frames, which can affect the time and frequency resolution. Ideally, these parameters should be optimised in conjunction with the parameters of the separation model to maximise performance for a particular separation task. In practice, however, the transform parameters are fixed to specific values. Secondly, since the separation model does not estimate the source phase, it is often assumed to be equal to the mixture phase, which is incorrect for overlapping partials. Alternatively, the Griffin-Lim algorithm can be applied to find an approximation to a signal whose magnitudes are equal to the estimated ones, but this is slow and often no such signal exists [8]. Lastly, the mixture phase is ignored in the estimation of sources, which can potentially limit the performance. Thus, it would be desirable for the separation model to learn to estimate the source signals including their phase directly.

As an approach to tackle the above problems, several audio processing models were recently proposed that operate directly on time-domain audio signals, including speech denoising as a task related to general audio source separation [1, 16, 18]. Inspired by these first results, we investigate in this paper the potential of fully end-to-end time-domain separation systems in the face of unresolved challenges. In particular, it is not clear if such a system will be able to deal effectively with the very long-range temporal dependencies present in audio due to its high sampling rate. Further, it is not obvious upfront whether the additional phase information will indeed be beneficial for the task, or whether the noisy phase might be detrimental for the learning dynamics in such a system. Overall, our contributions in this paper can be summarised as follows.

- We propose the Wave-U-Net, a one-dimensional adaptation of the U-Net architecture [7, 19], which separates sources directly in the time domain and can take large temporal contexts into account.
- We show a way to provide the model with additional input context to avoid artifacts at the boundaries of output windows, in contrast to previous work [7, 16].
- We replace strided transposed convolution used in previous work [7, 16] for upsampling feature maps with linear interpolation followed by a normal convolution to avoid artifacts.



© Daniel Stoller, Sebastian Ewert, Simon Dixon. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Daniel Stoller, Sebastian Ewert, Simon Dixon. "Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

This work was partially funded by EPSRC grant EP/L01632X/1. Implementation available at <https://github.com/f90/Wave-U-Net>

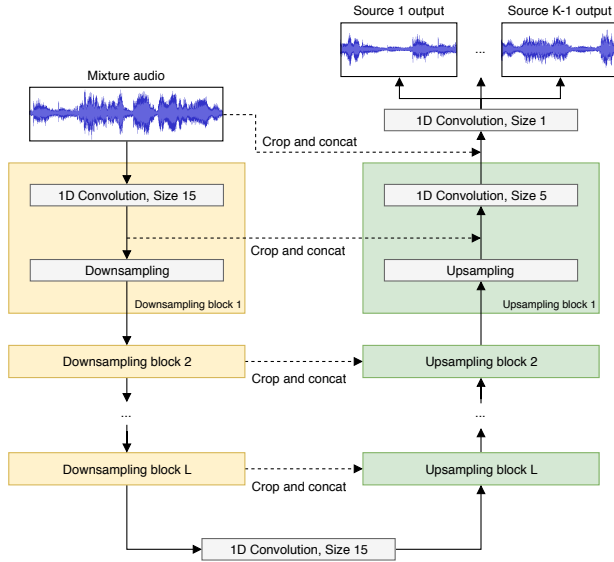


Figure 1. Our proposed Wave-U-Net with K sources and L layers. With our difference output layer, the K -th source prediction is the difference between the mixture and the sum of the other sources.

- The Wave-U-Net achieves good multi-instrument and singing voice separation, the latter of which compares favourably to our re-implementation of the state-of-the-art network architecture [7], which we train under comparable settings.
- Since the Wave-U-Net can process multi-channel audio, we compare stereo with mono source separation performance
- We highlight an issue with the commonly used Signal-to-Distortion ratio evaluation metric, and propose a work-around.

It should be noted that we expect the current state of the art model as presented in [7] to yield higher separation quality than what we report here, as the training dataset used in [7] is well-designed, highly unbiased and considerably larger. However, we believe that our comparison with a re-implementation trained under similar conditions might be indicative of relative performance improvements.

2. RELATED WORK

To alleviate the problem of fixed spectral representations widely used in previous work [6, 11, 13, 14, 20, 23], an adaptive front-end for spectrogram computation was developed [24] that is trained jointly with the separation network, which operates on the resulting magnitude spectrogram. Despite comparatively increased performance, the model does not exploit the mixture phase for better source magnitude predictions and also does not output the source phase, so the mixture phase has to be used for source signal reconstruction, both of which limit performance.

To our knowledge, only the TasNet [12] and MRCAE [4] systems tackle the general problem of audio source separation in the time domain. The TasNet performs a decomposition of the signal into a set of basis signals and weights,

and then creates a mask over the weights which are finally used to reconstruct the source signals. The model is shown to work for a speech separation task. However, the work makes conceptual trade-offs to allow for low-latency applications, while we focus on offline application, allowing us to exploit a large amount of contextual information.

The multi-resolution convolutional auto-encoder (MRCAE) [4] uses two layers of convolution and transposed convolution each. The authors argue the different convolutional filter sizes detect audio frequencies with different resolutions, but they work only on one time resolution (that of the input), since the network does not perform any resampling. Since input and output consist of only 1025 audio samples (equivalent to 23 ms), it can only exploit very little context information. Furthermore, at test time, output segments are overlapped using a regular spacing and then combined, which differs from how the network is trained. This mismatch and the small context could hurt performance and also explain why the provided sound examples exhibit many artifacts.

For the purpose of speech enhancement and denoising, the SEGAN [16] was developed, employing a neural network with an encoder and decoder pathway that successively halves and doubles the resolution of feature maps in each layer, respectively, and features skip connections between encoder and decoder layers. While we use a similar architecture, we rectify the issue of aliasing artifacts in the generated output when using strided transposed convolutions as shown by [15]. Furthermore, the model cannot predict audio samples close to its border output well since it is given no additional input context, which is an issue we address using convolutions with proper padding. It is also not clear if the model’s performance can transfer to other and more challenging audio source separation tasks.

The Wavenet [1] was adapted for speech denoising [18] to have a non-causal conditional input and a parallel output of samples for each prediction and is based on the repeated application of dilated convolutions with exponentially increasing dilation factors to factor in context information. While this architecture is very parameter-efficient, memory consumption is high since each feature map resulting from a dilated convolution still has the original audio’s sampling rate as resolution.

In contrast, our approach calculates the longer-term dependencies based on feature maps with more features and increasingly lower resolution. This saves memory and enables a large number of high-level features, which arguably do not need sample-level resolution to be useful, such as instrument activity, or the position in the current measure.

3. THE WAVE-U-NET MODEL

Our goal is to separate a mixture waveform $\mathbf{M} \in [-1, 1]^{L_m \times C}$ into K source waveforms $\mathbf{S}^1, \dots, \mathbf{S}^K$ with $\mathbf{S}^k \in [-1, 1]^{L_s \times C}$ for all $k \in \{1, \dots, K\}$, C as the number of audio channels and L_m and L_s as the respective numbers of audio samples. For model variants with extra input context, we have $L_m > L_s$ and make predictions for the centre part of the input.

Block	Operation	Shape
	Input	(16384, 1)
DS, repeated for $i = 1, \dots, L$	$\text{Conv1D}(F_c \cdot i, f_d)$	
	Decimate	(4, 288)
	$\text{Conv1D}(F_c \cdot (L + 1), f_d)$	(4, 312)
US, repeated for $i = L, \dots, 1$	Upsample	
	Concat(DS block i)	
	$\text{Conv1D}(F_c \cdot i, f_u)$	(16834, 24)
	Concat(Input)	(16834, 25)
	$\text{Conv1D}(K, 1)$	(16834, 2)

Table 1. Block diagram of the base architecture. Shapes describe the final output after potential repeated application of blocks, for the example of model M1, and denote the number of time steps and feature channels, in that order. DS block i refers to the output before decimation. Note that the US blocks are applied in reverse order, from level L to 1.

3.1 The base architecture

A diagram of the Wave-U-Net architecture is shown in Figure 1. It computes an increasing number of higher-level features on coarser time scales using downsampling (DS) blocks. These features are combined with the earlier computed local, high-resolution features using upsampling (US) blocks, yielding multi-scale features which are used for making predictions. The network has L levels in total, with each successive level operating at half the time resolution as the previous one. For K sources to be estimated, the model returns predictions in the interval $(-1, 1)$, one for each source audio sample.

The detailed architecture is shown in Table 1. $\text{Conv1D}(x, y)$ denotes a 1D convolution with x filters of size y . It includes zero-padding for the base architecture, and is followed by a LeakyReLU activation (except for the final one, which uses tanh). *Decimate* discards features for every other time step to halve the time resolution. *Upsample* performs upsampling in the time direction by a factor of two, for which we use linear interpolation (see Section 3.1.1 for details). *Concat*(x) concatenates the current, high-level features with more local features x . In extensions of the base architecture (see below), where Conv1D does not involve zero-padding, x is centre-cropped first so it has the same number of time steps as the current layer.

3.1.1 Avoiding aliasing artifacts due to upsampling

Many related approaches use transposed convolutions with strides to upsample feature maps [7, 16]. This can introduce aliasing effects in the output, as shown for the case of image generation networks [15]. In initial tests, we also found artifacts when using such convolutions as upsampling blocks in our Wave-U-Net model in the form of high-frequency buzzing noise.

Transposed convolutions with a filter size of k and a stride of $x > 1$ can be viewed as convolutions applied to feature maps padded with $x - 1$ zeros between each original value [2]. We suspect that the interleaving with zeros without subsequent low-pass filtering introduces high-frequency patterns into the feature maps, shown symbolically in Figure 2, which leads to high-frequency noise in the final output as well. Instead of transposed strided convolutions, we thus perform linear interpolation for upsampling, which ensures temporal continuity in the feature space, followed by a normal convolution. In initial tests, we did not observe

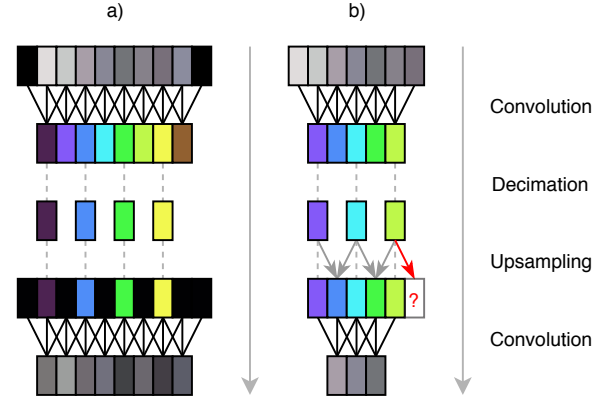


Figure 2. a) Common model (e.g. [7]) with an even number of inputs (grey) which are zero-padded (black) before convolving, creating artifacts at the borders (dark colours). After decimation, a transposed convolution with stride 2 is shown here as upsampling by zero-padding intermediate and border values followed by normal convolution, which likely creates high-frequency artifacts in the output. b) Our model with proper input context and linear interpolation for upsampling from Section 3.2.2 does not use zero-padding. The number of features is kept uneven, so that upsampling does not require extrapolating values (red arrow). Although the output is smaller, artifacts are avoided.

any high-frequency sound artifacts in the output with this technique and achieved very similar performance.

3.2 Architectural improvements

The previous Section described the baseline variant of the Wave-U-Net. In the following, we will describe a set of architectural improvements for the Wave-U-Net designed to increase model performance.

3.2.1 Difference output layer

Our baseline model outputs one source estimate for each of K sources by independently applying K convolutional filters followed by a tanh non-linearity to the last feature map. In the separation tasks we consider, the mixture signal is the sum of its source signal components: $\mathbf{M} \approx \sum_{j=1}^K \mathbf{S}^j$. Since our baseline model is not constrained in this fashion, it has to learn this rule approximately to avoid highly improbable outputs, which could slow down learning and reduce performance. Therefore, we use a difference output layer to constrain the outputs $\hat{\mathbf{S}}^j$, enforcing $\sum_{j=1}^K \hat{\mathbf{S}}^j = \mathbf{M}$: only $K - 1$ convolutional filters with a size of 1 are applied to the last feature map of the network, followed by a tanh non-linearity, to estimate the first $K - 1$ source signals. The last source is then simply computed as $\hat{\mathbf{S}}^K = \mathbf{M} - \sum_{j=1}^{K-1} \hat{\mathbf{S}}^j$.

This type of output was also used for speech denoising in [18] as part of an “energy-conserving” loss, and a similar idea can be found very commonly in spectrogram-based source separation in the form of masks that distribute the energy of the input mixture magnitudes to the output sources. We investigate the impact of introducing this layer and its additivity assumption, since it depends on the extent to which this additivity property is satisfied by the data.

3.2.2 Prediction with proper input context and resampling

In previous work [4, 7, 16], the input and the feature maps are padded with zeros before convolving, so that the resulting feature map does not change in its dimension, as shown in Figure 2a. This simplifies the network’s implementation, since the input and output dimensions are the same. Zero-padding audio or spectrogram input this way effectively extends the input using silence at the beginning and end. However, taken from a random position in a full audio signal, the information at the boundary becomes artificial, i.e. the temporal context for this excerpt is given in the full audio signal but is ignored and assumed to be silent. Without proper context information, the network thus has difficulty predicting output values near the beginning and end of the sequence. As a result, simply concatenating the outputs as non-overlapping segments at test time to obtain the prediction for a full audio signal can create audible artifacts at the segment borders, as neighbouring outputs can be inconsistent when they are generated without correct context information. In Section 5.2, we investigate this behaviour in practice.

As a solution, we employ convolutions without implicit padding and instead provide a mixture input larger than the size of the output prediction, so that the convolutions are computed on the correct audio context (see Figure 2b). Since this reduces the feature map sizes, we constrain the possible output sizes of the network so that feature maps are always large enough for the following convolution.

Further, when resampling feature maps, feature dimensions are often exactly halved or doubled [7, 16], as shown in Figure 2a for transposed strided convolution. However, this necessarily involves extrapolating at least one value at a border, which can again introduce artifacts. Instead, we interpolate only between known neighbouring values and keep the very first and last entries, producing $2n - 1$ entries from n or vice versa, as shown in Figure 2b. To recover the intermediate values after decimation, while keeping border values the same, we ensure that feature maps have odd dimensionality.

3.2.3 Stereo channels

To accommodate for multi-channel input with C channels, we simply change the input \mathbf{M} from an $L_m \times 1$ to an $L_m \times C$ matrix. Since the second dimension is treated as a feature channel, the first convolution of the network takes into account all input channels. For multi-channel output with C channels, we modify the output component to have K independent convolutional layers with filter size 1 and C filters each. With a difference output layer, we only use $K - 1$ such convolutional layers. We use this simple approach with $C = 2$ to perform experiments with stereo recordings and investigate the degree of improvement in source separation metrics when using stereo instead of mono estimation.

3.2.4 Learned upsampling for Wave-U-Net

Linear interpolation for upsampling is simple, parameterless and encourages feature continuity. However, it may be restricting the network capacity too much. Perhaps, the feature spaces used in these feature maps are not structured

so that a linear interpolation between two points in feature space is a useful point on its own, so that a learned upsampling could further enhance performance. To this end, we propose the learned upsampling layer. For a given $F \times n$ feature map with n time steps, we compute an interpolated feature $f_{t+0.5} \in \mathbb{R}^F$ for pairs of neighbouring features $f_t, f_{t+1} \in \mathbb{R}^F$ using parameters $w \in \mathbb{R}^F$ and the sigmoid function σ to constrain each $w_i \in w$ to the $[0, 1]$ interval:

$$f_{t+0.5} = \sigma(w) \odot f_t + (1 - \sigma(w)) \odot f_{t+1} \quad (1)$$

This can be implemented as a 1D convolution across time with F filters of size two and no padding with a properly constrained matrix. The learned interpolation layer can be viewed as a generalisation of simple linear interpolation, since it allows convex combinations of features with weights other than 0.5.

4. EXPERIMENTS

We evaluate the performance of our models on two tasks: Singing voice separation and music separation with bass, drums, guitar, vocals and “other” instruments as categories, as defined by the SiSec separation campaign [10].

4.1 Datasets

75 tracks from the training partition of the MUSDB [17] multi-track database are randomly assigned to our training set, and the remaining 25 tracks form the validation set, which is used for early stopping. Final performance is evaluated on the MUSDB test partition comprised of 50 songs. For singing voice separation, we also add the whole CCMixer database [9] to the training set.

As data augmentation for both tasks, we multiply source signals with a factor chosen uniformly from the interval $[0.7, 1.0]$ and set the input mixture as the sum of source signals. No further data preprocessing is performed, only a conversion to mono (except for stereo models) and down-sampling to 22050 Hz.

4.2 Training procedure

During training, audio excerpts are sampled randomly and inputs padded accordingly for models with input context. As loss, we use the mean squared error (MSE) over all source output samples in a batch. We use the ADAM optimizer with learning rate 0.0001, decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a batch size of 16. We define 2000 iterations as one epoch, and perform early stopping after 20 epochs of no improvement on the validation set, measured by the MSE loss. Afterwards, the last model is fine-tuned further, with the batch size doubled and the learning rate lowered to 0.00001, again until 20 epochs without improvement in validation loss. Finally, the model with the best validation loss is selected.

4.3 Model settings and variants

For our baseline model, we use $L_m = L_s = 16384$ input and output samples, $L = 12$ layers, $F_c = 24$ extra filters per layer and filter sizes $f_d = 15$ and $f_u = 5$.

To determine the impact of the model improvements described in Section 3.2, we train a baseline model M1 as described in Section 3.1 and models M2 to M5 which add the difference output layer from Section 3.2.1 (M2), the input context and resampling from Section 3.2.2 (M3), stereo channels from Section 3.2.3 (M4), and learned upsampling from Section 3.2.4 (M5), and also contain all features of the respectively previous model. We apply the best model of the above (M4) to multi-instrument separation (M6). Models with input context (M3 to M6) have $L_m = 147443$ input and $L_s = 16389$ output samples.

For comparison with previous work, we also train the spectrogram-based U-Net architecture [7] (U7) that achieved state-of-the-art vocal separation performance, and a Wave-U-Net comparison model (M7) under the same conditions, both using the audio-based MSE loss and mono signals downsampled to 8192 Hz. M7 is based on the best model M4, but is set to $L_m = 233459$ and $L_s = 102405$ to have very similar output size compared to U7 ($L_s = 98650$ samples), $F_c = 34$ to bring our network to the same size as U7 (20M param.), and the initial batch size is set to four due to the high amount of memory needed per sample. To train U7, we backpropagate the error through the inverse STFT operation that is used to construct the source audio signal from the estimated spectrogram magnitudes and the mixture phase. We also train the same model with an L1 loss on the spectral magnitudes (U7a), following [7]. Since the training procedure and loss are exactly the same for networks U7 and M7, we can fairly compare both architectures by ensuring that performance differences do not arise simply because of the amount of training data or the type of loss function used, and also compare with a spectrogram-based loss (U7a). Despite our effort to enable an overall model comparison, note that some training settings such as learning rates used in [7] might differ from ours (and are partly unknown) and could provide better performance with U7 and U7a than shown here, even with the same dataset.

5. RESULTS

5.1 Quantitative results

5.1.1 Evaluation metrics

The signal-to-distortion (SDR) metric is commonly used to evaluate source separation performance [25]. An audio track is usually partitioned into non-overlapping audio segments multiple seconds in length, and segment-wise metrics are then averaged over each audio track or the whole dataset to evaluate model performance. Following the procedure used for the SiSec separation campaign 2018 [17], these segments are one second long.

5.1.2 Issues with current evaluation metrics

The SDR computation is problematic when the true source is silent or near-silent. In case of silence, the SDR is undefined ($\log(0)$), which happens often for vocal tracks. Such segments are excluded from the results, so performance on these segments is ignored. For near-silent parts, the SDR is typically very low when the separator output is quiet, but not silent, although such an output is arguably not a

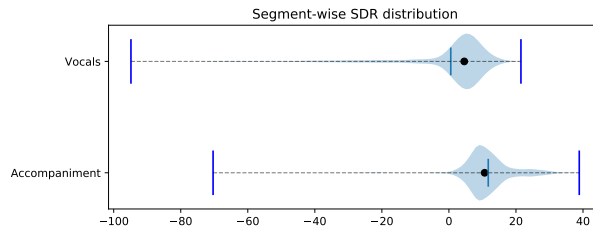


Figure 3. Violin plot of the segment-wise SDR values in the MUSDB test set for model M5. Black points show medians, dark blue lines the means.

grave error perceptually. These outliers are visualised using model M5 in Figure 3. Since the mean over segments is usually used to obtain overall performance measures, these outliers greatly affect evaluation results.

Since the collection of segment-wise vocal SDR values across the dataset is not normally distributed (compare Figure 3 for vocals), the mean and standard deviation are not sufficient to adequately summarise it. As a workaround, we take the median over segments, as it is robust against outliers and intuitively describes the minimum performance that is achieved 50% of the time. To describe the spread of the distribution, we use the median absolute deviation (MAD) as a rank-based equivalent to the standard deviation (SD). It is defined as the median of the absolute deviations from the overall median and is easily interpretable, since a value of x means that 50% of values have an absolute difference from the median that is lower than x .

We also note that increasing the duration of segments beyond one second alleviates this issue by removing many, but not all outliers. This is more memory-intensive and presumably still punishes errors during silent sections most.

5.1.3 Model comparison

Table 2 shows the evaluation results for singing voice separation. The low vocal SDR means and high medians for all models again demonstrate the outlier problem discussed in Section 5.1.2. The difference output layer does not noticeably change performance, as model M2 appears to be only very slightly better than model M1. Initial experiments without fine-tuning showed a larger difference, which may indicate that a finer adjustment of weights makes constrained outputs less important, but they could still enable the usage of faster learning rates. Introducing context noticeably improves performance, as model M3 shows, likely due to better predictions at output borders. The stereo modeling in model M4 yields improvements especially for accompaniment, which may be because its sounds are panned more to the left or right channels than vocals. The learned upsampling (M5) slightly improves the median, but slightly decreases the mean vocal SDR. The small differences could be explained by the low number of weights in learned upsampling layers, considering that we also experimented with unconstrained convolutions, which brought more improvements but also high-frequency sound artifacts. We therefore consider M4 as our best model. For multi-instrument separation, we achieve slightly lower but moderate performance (M6), as shown in Table 3, in part due to less training data.

		M1	M2	M3	M4	M5	M7	U7	U7a
Voc.	Med.	3.90	3.92	3.96	4.46	4.58	3.49	2.76	2.74
	MAD	3.04	3.01	3.00	3.21	3.28	2.71	2.46	2.54
	Mean	-0.12	0.05	0.31	0.65	0.55	-0.23	-0.66	0.51
	SD	14.00	13.63	13.25	13.67	13.84	13.00	12.38	10.82
Acc.	Med.	7.45	7.46	7.53	10.69	10.66	7.12	6.76	6.68
	MAD	2.08	2.10	2.11	3.15	3.10	2.04	2.00	2.04
	Mean	7.62	7.68	7.66	11.85	11.74	7.15	6.90	6.85
	SD	3.93	3.84	3.90	7.03	7.05	4.10	3.67	3.60

Table 2. Test set performance metrics (SDR statistics, in dB) for each singing voice separation model. Best performances overall and among comparison models are shown in bold.

	Vocals				Other			
	Med.	MAD	Mean	SD	Med.	MAD	Mean	SD
M6	3.0	2.76	-2.10	15.41	2.03	1.64	1.68	6.14

	Bass				Drums			
	Med.	MAD	Mean	SD	Med.	MAD	Mean	SD
M6	2.91	2.47	-0.30	13.50	4.15	1.99	2.88	7.68

Table 3. Test performance metrics (SDR statistics, in dB) for our multi-instrument model

U7 performs worse than our comparison model M7, suggesting that our network architecture compares favourably to the state-of-the-art architecture since all else is kept constant during the experiments. However, U7 stopped improving on the training set unexpectedly early, perhaps because it was not designed for minimising an audio-based MSE loss or because of effects related to backpropagating gradients through the inverse STFT. In contrast, U7a showed expected training behaviour using the magnitude-based loss. Our model also outperforms U7a, yielding considerably higher mean and median SDR scores. The mean vocal SDR is the only exception, arising since our model has more outlier segments, but better output the majority of the time.

Models M4 and M6 were submitted as STL1 and STL2 to the SiSec campaign [22]. For vocals, M4 performs better or as well as almost all other systems. Although it is significantly outperformed by submissions UHL3, TAK1-3 and TAU1, all of these except TAK1 used an additional 800 songs for training and thus have a large advantage. M4 also separates accompaniment well, although slightly less so than the vocals. We refer to [22] for more details.

5.2 Qualitative results and observations

As an example of problems occurring when not using a proper temporal context, we generated a vocal source estimate for a song with the baseline model M1, and visualised an excerpt using a spectrogram in Figure 4. Since the model’s input and output are of equal length and the total output is created by concatenating predictions for non-overlapping consecutive audio segments, inconsistencies emerge at the borders shown in red: the loudness abruptly decreases at 1.2 seconds, and a beginning vocal melisma is suddenly cut off at 2.8 seconds, leaving only quiet noise, before the vocals reappear at 4.2 seconds. A vocal melisma with only the vowel “a” can sound similar to a non-vocal instrument and presumably was mistaken for one because no further temporal context was available.

In conclusion, these models suffer not only from inconsistencies at such segment borders, but are also less capable of performing separation there whenever information from a temporal context is required. Larger input and output sizes alleviate the issue somewhat, but the problems at the

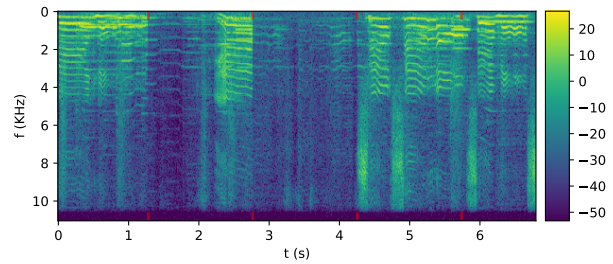


Figure 4. Power spectrogram (dB) of a vocal estimate excerpt generated by a model without additional input context. Red markers show boundaries between independent segment-wise predictions.

borders remain. Blending the predictions for overlapping segments [4] is an ad-hoc solution, since the average of multiple predicted audio signals might not be a realistic prediction itself. For example, two sinusoids with equal amplitude and frequency, but opposite phase would cancel each other out. Blending should thus be avoided in favour of our context-aware prediction framework.

6. DISCUSSION AND CONCLUSION

In this paper, we proposed the Wave-U-Net for end-to-end audio source separation without any pre- or postprocessing, and applied it to singing voice and multi-instrument separation. A long temporal context is processed by repeated downsampling and convolution of feature maps to combine high- and low-level features at different time-scales. As indicated by our experiments, it outperforms the state-of-the-art spectrogram-based U-Net architecture [7] when trained under comparable settings. Since our data is quite limited in size however, it would be interesting to train our model on datasets comparable in size to the one used in [7] to better assess respective advantages and disadvantages.

We highlight the lack of a proper temporal input context in recent separation and enhancement models, which can hurt performance and create artifacts, and propose a simple change to the padding of convolutions as a solution. Similarly, artifacts resulting from upsampling by zero-padding as part of strided transposed convolutions can be addressed with a linear upsampling with a fixed or learned weight to avoid high-frequency artifacts.

Finally, we identify a problem in current SDR-based evaluation frameworks that produces outliers for quiet parts of sources and propose additionally reporting rank-based metrics as a simple workaround. However, the underlying problem of perceptual evaluation of sound separation results using SDR metrics still remains and should be tackled at its root in the future.

For future work, we could investigate to which extent our model performs a spectral analysis, and how to incorporate computations similar to those in a multi-scale filterbank, or to explicitly compute a decomposition of the input signal into a hierarchical set of basis signals and weightings on which to perform the separation, similar to the TasNet [12]. Furthermore, better loss functions for raw audio prediction should be investigated such as the ones provided by generative adversarial networks [3, 21], since the MSE might not reflect the perceived loss of quality well.

7. REFERENCES

- [1] Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [2] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.
- [3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.
- [4] Emad M Grais, Dominic Ward, and Mark D Plumbley. Raw multi-channel audio source separation using multi-resolution convolutional auto-encoders. *arXiv preprint arXiv:1803.00702*, 2018.
- [5] D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.
- [6] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Singing-voice separation from monaural recordings using deep recurrent neural networks. In *International Society for Music Information Retrieval (ISMIR)*, pages 477–482, 2014.
- [7] Andreas Jansson, Eric J. Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep U-Net convolutional networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 323–332, 2017.
- [8] Jonathan Le Roux, Nobutaka Ono, and Shigeki Sagayama. Explicit consistency constraints for STFT spectrograms and their application to phase reconstruction. In *SAPA@ INTERSPEECH*, pages 23–28, 2008.
- [9] Antoine Liutkus, Derry Fitzgerald, and Zafar Rafii. Scalable audio separation with light kernel additive modelling. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 76–80. IEEE, 2015.
- [10] Antoine Liutkus, Fabian-Robert Stöter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave. The 2016 signal separation evaluation campaign. In *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, pages 323–332, 2017.
- [11] Y. Luo, Z. Chen, J. R. Hershey, J. Le Roux, and N. Mesgarani. Deep clustering and conventional networks for music separation: Stronger together. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 61–65, 2017.
- [12] Yi Luo and Nima Mesgarani. Tasnet: time-domain audio separation network for real-time, single-channel speech separation. *CoRR*, abs/1711.00541, 2017.
- [13] Marius Miron, Jordi Janer Mestres, and Emilia Gómez Gutiérrez. Generating data to train convolutional neural networks for classical music source separation. In *Proceedings of the 14th Sound and Music Computing Conference*. Aalto University, 2017.
- [14] Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent. *Multichannel audio source separation with deep neural networks*. PhD thesis, Inria, 2015.
- [15] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.
- [16] Santiago Pascual, Antonio Bonafonte, and Joan Serra. Segan: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*, 2017.
- [17] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation, 2017.
- [18] Dario Rethage, Jordi Pons, and Xavier Serra. A wavenet for speech denoising. *CoRR*, abs/1706.07162, 2017.
- [19] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.
- [20] Andrew JR Simpson, Gerard Roma, and Mark D Plumbley. Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 429–436. Springer, 2015.
- [21] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Adversarial semi-supervised audio source separation applied to singing voice extraction. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2391–2395, Calgary, Canada, 2018. IEEE.
- [22] F.-R. Stöter, A. Liutkus, and N. Ito. The 2018 Signal Separation Evaluation Campaign. *ArXiv e-prints*, 2018.
- [23] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 261–265, March 2017.
- [24] Shrikant Venkataramani and Paris Smaragdis. End-to-end source separation with adaptive front-ends. *CoRR*, abs/1705.02514, 2017.
- [25] E. Vincent, R. Gribonval, and C. Fevotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1462–1469, 2006.

SE AND S_N L DIAGRAMS: FLEXIBLE DATA STRUCTURES FOR MIR

Melissa R. McGuirl¹

Katherine M. Kinnaird¹
Erin H. Bugbee³

Claire Savard²

¹ Division of Applied Mathematics, Brown University, USA

² Department of Mathematics, University of Michigan, USA

³ Department of Biostatistics, Brown University, USA

melissa.mcguirl@brown.edu

ABSTRACT

The matrix-based representations commonly used in MIR tasks are often difficult to interpret. This work introduces start-end (SE) diagrams and start(normalized)-length (S_N L) diagrams, two novel structure-based representations for sequential music data. Inspired by methods from topological data analysis, both SE and S_N L diagrams come equipped with efficiently computable and stable metrics. Utilizing SE or S_N L diagrams as input, we address the cover song task for score-based data with high accuracy. While both representations are concisely defined and flexible, S_N L diagrams in particular address issues introduced by commonly used resampling methods.

1. INTRODUCTION

Since Foote’s introduction of the self-similarity matrix (SSM) in [8], matrix-based representations for music-based data streams have been commonly used in MIR literature. Both SSMs and self-dissimilarity matrices (SDMs) have been used as the starting point for a variety of tasks including the cover song task [2, 10, 13, 21], the chorus detection task [9], and segmentation task [14, 18, 19].

While straightforward to compute, these matrix-based representations are challenging to interpret, requiring extensive post-processing, such as smoothing and resampling techniques used in [10] or path enhancement applied in [15–17]. These post-processing steps can also introduce uncertainty or reduce some of the intuitive explanations for the resulting visualizations. The aligned hierarchies from [13] is an intuitive structure-based representation that is also the result of post-processing SDMs. However, this representation is rigid as it requires two songs to be the exactly the same length for comparisons. The aligned sub-hierarchies attempt to address this rigidity, but many songs do not have enough structure to have this collection of structure-based representations for sections of a song [12].

In this paper, we contribute two new structure-based visualizations for music-based data streams: *Start-end diagrams* (in Section 3) and *Start(normalized)-length diagrams* (in Section 4). With roots in topological data analysis, the presented methods are flexible, computationally efficient, and easily adaptable. Moreover, we present experiments applying these methods to a version of the cover song task. We discuss contributions of our novel methods (in Section 6) and share future directions (in Section 7).

2. MOTIVATION AND BACKGROUND

This work builds upon aligned hierarchies developed in [13]. The aligned hierarchies for a song encodes all possible hierarchical structure decompositions of that song on one common time axis. The aligned hierarchies representation is defined as a collection of three components: a binary onset matrix B_H , a length vector, and an annotation vector that acts as a key for B_H [13]. Each row of B_H corresponds to one kind of repetition, with entries equal to one denoting where instances of a repeat begins.

Aligned hierarchies have been used to compare songs under the fingerprint task by leveraging that this representation can be embedded into a classification space with a natural notion of distance. This distance computes the number of dissimilarities between start-times for repeats of each size and then totals those dissimilarities across all sizes. Using the aligned hierarchies as the basis of comparison yields precise results, yet the metric is both rigid with respect to the length of the songs and computationally expensive as it is based on a binary classification [13].

In this work, we produce novel methods of representing and comparing songs. Inspired by work in topological data analysis, our methods extend the aligned hierarchies while addressing their limitations. Moreover, we offer several variations of our method, which make our representations flexible and easily adaptable to many applications such as cover song and remix detection.

3. START-END DIAGRAMS

Aligned hierarchies represents repeated structures of music data. Similarly, topological data analysis (TDA), an emerging field of mathematics, aims to extract structural, or topological, information from complex data. The start-



© Melissa R. McGuirl, Katherine M. Kinnaird, Claire Savard, Erin H. Bugbee. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Melissa R. McGuirl, Katherine M. Kinnaird, Claire Savard, Erin H. Bugbee. “SE and S_N L diagrams: Flexible data structures for MIR”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

end diagram is a transformation of the aligned hierarchies that is reminiscent of persistence diagrams from TDA.

In TDA, data are thresholded via a sequence of parameter values. Topological summaries, such as the number of loops in the data, are then computed for each parameter value in the sequence [4, 5, 7]. A common way to represent this topological information is with persistence diagrams. Briefly, a *persistence diagram* is a collection of points $\{(b_i, d_i)\}_{i=1}^N \subset \mathbb{R}_+^2$, such that (b_i, d_i) corresponds to a topological structure that appears at some parameter value b_i and disappears at parameter value $d_i \geq b_i$ [4, 5, 7].

Inspired by TDA, we transform aligned hierarchies into a *start-end (SE) diagram*. The *SE diagram* corresponding to aligned hierarchies with N repeated structures is defined as a collection of points $\{(s_i, e_i)\}_{i=1}^N \subset \mathbb{R}_+^2$, where s_i and e_i are the start and end times, respectively, of the i^{th} repeated structure. Under this transformation, we adjust the time scale such that time zero refers to the start of the first block of the aligned hierarchies and truncate the song to end where the last block of the aligned hierarchies ends.

SE diagrams are not inherently topological (in a mathematical sense), rather we are adapting data structures from TDA. While SE diagrams cannot delineate two different types of repeats of the same length, there are several advantages of using SE diagrams over aligned hierarchies. First, they are a more concisely defined structure, as each diagram is simply a finite collection of points. Second, leveraging theoretical results from TDA, there are easily adaptable metrics on the space of SE diagrams (Subsection 3.1). Third, these metrics are more flexible than those for the aligned hierarchies while maintaining accuracy and precision in the cover song task (Subsection 3.2).

3.1 Metrics for SE diagrams

In TDA there are two common metrics for persistence diagrams that can be extended to SE diagrams: *the bottleneck metric* and *the Wasserstein metric*. Both metrics measure the error of an optimal alignment of points in two persistence (or SE) diagrams. The metrics are stable, meaning small differences between aligned hierarchies will yield a small SE diagram distance [6, 7, 11]. Moreover, as shown in [11], these distances can be computed efficiently using k -dimensional trees. Thus, under either the Wasserstein or bottleneck notions, SE diagrams are equipped with stable and computable metrics which facilitate their ability to address the cover song task efficiently and accurately.

3.1.1 Intuitive definitions

Intuitively, the Wasserstein and bottleneck metrics attempt to find the best alignment of points between two SE diagrams and then measure cost of the alignment using an l^p metric. When aligning two diagrams for comparison, each diagram point must have a corresponding aligned point in the other diagram and no points can be aligned with more than one point. The aligned points thus form a pair.

Recall, the l^p norm for any point $\vec{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ is given by $\|\vec{x}\|_p = (\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}$ for $1 \leq p < \infty$, and the l^∞ norm is $\|\vec{x}\|_\infty = \max_i |x_i|$. Norms naturally give

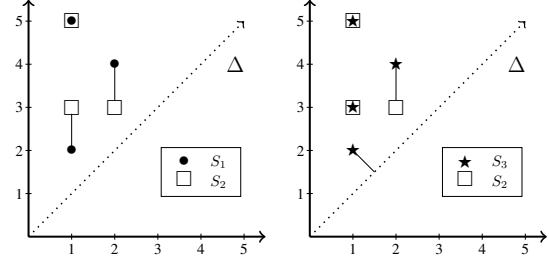


Figure 1. Optimal alignment of S_1 and S_2 without aligning with Δ (left), and optimal alignment of S_2 and S_3 while allowing for alignments with Δ (right). Note that $\Delta \sim (1, 2) \in S_3$ and the l^2 distances between the pairs are $\{\sqrt{\frac{1}{2}}, 0, 1, 0\}$, so $d_W^{2,1}(S_2, S_3) = \frac{1+\sqrt{2}}{\sqrt{2}}$, $d_B^2(S_2, S_3) = 1$.

rise to metrics. Specifically, the l^p metric $d_p : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^+$ between any two points $\vec{x}, \vec{y} \in \mathbb{R}^n$ is defined for $1 \leq p \leq \infty$ as $d_p(\vec{x}, \vec{y}) = \|\vec{x} - \vec{y}\|_p$. Note that in \mathbb{R}^2 , d_2 is the the straight line distance between two points in the plane, while d_∞ is the maximum of the horizontal and vertical distance between two points in the plane.

For example, consider two SE diagrams $S_1 = \{(1, 2), (1, 5), (2, 4)\}$ and $S_2 = \{(1, 3), (1, 5), (2, 3)\}$. To find the optimal alignment we pair points in diagram S_1 with points in diagram S_2 in a way that minimizes the total distance between all pairs. The best alignment of S_1 and S_2 is given by $\{(1, 2) \sim (1, 3), (1, 5) \sim (1, 5), (2, 4) \sim (2, 3)\}$ (see Figure 1). The corresponding l^∞ distances of these pairs are $\{1, 0, 1\}$. The (∞, q) -Wasserstein and ∞ -bottleneck metrics are then defined as the l^q and l^∞ , respectively, norms of the l^∞ distances of the pairs in the optimal alignment.

In this example, the $(\infty, 2)$ -Wasserstein distance between S_1 and S_2 is $d_W^{\infty, 2}(S_1, S_2) = \sqrt{2}$, whereas the ∞ -bottleneck distance between S_1 and S_2 is $d_B^\infty(S_1, S_2) = 1$. Note, the first superscript in $d_W^{\infty, 2}$ corresponds to taking the l^∞ distances between the points in each pair (inner norm), and the second superscript denotes taking l^2 norm of those l^∞ distances (outer norm).

Thus far we have defined distances between two SE diagrams with the same number of points. By definition, computing the Wasserstein or bottleneck distance between two SE diagrams requires both diagrams to have the same number of points. In practice, however, we want to compare SE diagrams with any number of points, as songs have varying amounts of repeated structures.

To compare SE diagrams of differing numbers of points we find the optimal alignment of points in two SE diagrams, while also allowing diagram points to match to repeated structures existing for no time, meaning their start and end times are the same. Formally, we allow points to align with the *diagonal*, defined as $\Delta = \{(s, e) : s = e, s \geq 0\}$ (see Figure 1) [6, 7, 11].

The motivation for allowing points to align with repeated structures that exist for no time is two-fold. First, unlike arbitrary insertions or deletions of points in either SE diagrams, aligning points with Δ will give rise to a

metric that respects the triangle inequality. With the triangle inequality, it is impossible to have the case where songs B and C are both cover songs of song A so that $d(A, B)$ and $d(A, C)$ are small, but $d(B, C)$ is large.

Second, pairing unmatched points with Δ enforces that two songs will be considered dissimilar when their long-lasting repeated structures do not have a corresponding pair under the optimal alignment of points. To see this, observe that when a point $(s_*^1, e_*^1) \in S_1$ does not have a corresponding match in S_2 then $(s_*^1, e_*^1) \sim \Delta$ and this pairing contributes $d_p((s_*^1, e_*^1), \Delta) = 2^{\frac{1}{p}-1}|e_*^1 - s_*^1|$ to the overall cost of the alignment. Thus, the cost for unmatched points aligning with Δ increases as the length $(|e_*^1 - s_*^1|)$ of the unmatched repeated structure increases.

In short, the (p, q) -Wasserstein and p -Bottleneck metrics measure the distances between pairs of points in the optimal alignment of two SE diagrams and Δ . When $q = 2$, the Wasserstein distance is the Euclidean norm of the distances between pairs in the optimal alignment. In contrast, the Bottleneck distance is to the maximum distance between pairs in the optimal alignment. In the following subsection we provide rigorous definitions of these metrics.

3.1.2 Rigorous Definitions

Let $S_1 = \{(s_i^1, e_i^1)\}_{i \in I}$ and $S_2 = \{(s_j^2, e_j^2)\}_{j \in J}$ be SE diagrams, and let ϕ be a bijection between subsets $\tilde{I} \subset I$ and $\phi(\tilde{I}) \subset J$. The p - q penalty of ϕ is defined as:

$$P_q^p(\phi) = \sum_{i \in \tilde{I}} d_p((s_i^1, e_i^1), (s_{\phi(i)}^2, e_{\phi(i)}^2))^q + \sum_{i \in I \setminus \tilde{I}} d_p((s_i^1, e_i^1), \Delta)^q + \sum_{j \in J \setminus \phi(\tilde{I})} d_p((s_j^2, e_j^2), \Delta)^q,$$

for $1 \leq q < \infty$, and the ∞ - q penalty of ϕ is defined as:

$$P_\infty^p(\phi) = \max \left\{ \max_{i \in \tilde{I}} d_p((s_i^1, e_i^1), (s_{\phi(i)}^2, e_{\phi(i)}^2)), \max_{i \in I \setminus \tilde{I}} d_p((s_i^1, e_i^1), \Delta), \max_{j \in J \setminus \phi(\tilde{I})} d_p((s_j^2, e_j^2), \Delta) \right\}.$$

These penalties define a cost function for aligning points in S_1 with points in S_2 (encoded in the first terms), and for aligning all unmatched points with Δ (encoded in the remaining terms). The p -bottleneck distance is then defined as $d_B^p(S_1, S_2) = \min_{\phi} P_\infty^p(\phi)$ and the (p, q) -Wasserstein distance is $d_W^{p,q}(S_1, S_2) = \min_{\phi} P_q^p(\phi)^{\frac{1}{q}}$ [6, 7, 11].

3.2 Applications of SE diagrams

Utilizing the metrics described in the previous section, there are several ways of comparing songs for the cover song task. This work explores the efficacy of using the pairwise p -bottleneck and (p, q) -Wasserstein distances as input for a mutual nearest neighbor search.

Noting that the presented methods take aligned hierarchies as input, we pre-process music-based data in three

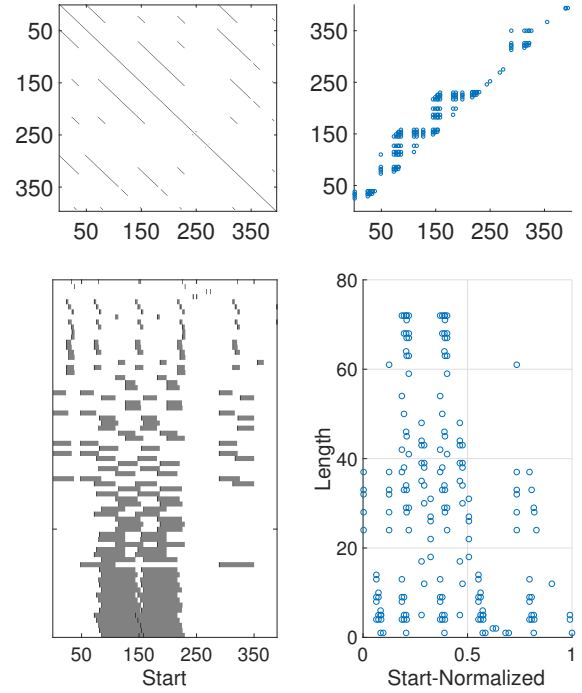


Figure 2. The thresholded SDM (top left), aligned hierarchies (bottom left), SE diagram (top right), and S_NL diagram with $\alpha = 1$ (bottom right) corresponding to Mazurka 52 expanded with threshold=0.02, shingle=12. Each dark block diagonal in the thresholded SDM represents two sections that are repeats of each other. Repetitions of all sizes are encoded in the aligned hierarchies as blocks, separated into rows. Each block in the aligned hierarchies is represented as a point in both the SE and S_NL diagrams. The smallest repeats are close to the diagonal in the SE diagram and are near the horizontal axis in the S_NL diagram. The tops of the peaks in the SE and S_NL diagrams represent the longest repetitions, which are the blocks at the bottom of the aligned hierarchies.

steps: 1) build audio shingles from the concatenated beat-synchronous chroma features, 2) compute the SDM, and finally 3) construct the aligned hierarchies for each song's SDM (see [13] for more details). After the aligned hierarchies are created, the procedure is as follows: ¹:

1. Transform each aligned hierarchies into the corresponding SE diagram as described in Section 3
2. Compute bottleneck or Wasserstein distances between pairs of SE diagrams using Hera [11] ²
3. Mark a pair of songs as cover songs of each other if the songs are mutual nearest neighbors

See Figure 2 for a visual example of our method. To test our method we apply it to 52 Mazurka scores by

¹ Code and processed data are publicly available here: https://github.com/MelissaMcguirl/SE_SNL_analysis.git.

² Hera is publicly available here: https://bitbucket.org/grey_narn/hera.

	Threshold	0.01		0.02		0.03		0.04		0.05	
	Shingle	6	12	6	12	6	12	6	12	6	12
d_B^∞ Metric	Precision	0.871	0.622	0.848	0.718	0.871	0.800	0.818	0.725	0.824	0.757
	Recall	0.519	0.538	0.538	0.538	0.519	0.538	0.519	0.558	0.538	0.538
$d_W^{2,2}$ Metric	Precision	0.966	0.675	0.879	0.903	0.933	0.824	0.909	0.875	0.875	0.848
	Recall	0.538	0.519	0.558	0.538	0.538	0.538	0.576	0.538	0.538	0.538
$d_W^{\infty,2}$ Metric	Precision	1.0	0.683	0.909	0.909	0.933	0.882	0.906	0.906	0.879	0.853
	Recall	0.558	0.538	0.577	0.577	0.538	0.577	0.558	0.558	0.558	0.558

Table 1. Precision and recall values for the mutual nearest neighbor matching of SE diagrams for 104 Mazurka scores.

Chopin downloaded in **kern format from KernScore online database³ (see [20]). Each score produces two data elements, an expanded version which includes all repeated sections as marked in the score, and a non-expanded version which has each repeated section played once. In the cover song task, the goal is to match the expanded and non-expanded versions of each song.

We construct SE diagrams for all 104 songs in our dataset and compute their pairwise distances for three metrics: d_B^∞ , $d_W^{2,2}$, and $d_W^{\infty,2}$. We perform 10 experiment trials per metric, varying the number of chroma vectors per audio shingle and varying the threshold applied to the SDM. The precision and recall values are presented in Table 1.

These results show that SE diagrams can accomplish this challenging version of the cover song task with high precision and moderate recall values regardless of the metric. It is crucial and exciting to note that the SE diagrams achieved these results without any resampling to the diagrams. Moreover, as we will see in the next section, this method is easily adaptable to several useful variations.

4. START(NORMALIZED)-LENGTH DIAGRAMS

Since the length of repeats (e-s) is represented diagonally on SE diagrams, these representations can be difficult to interpret. In this section we describe a transformation of SE diagrams called *start-length (SL) diagrams*, along with normalizations. SL diagrams are more intuitive to read than their predecessor and yield stronger experimental results. While reminiscent of the constellation maps from [22], SL diagrams encode structural repeats instead of audio spectrogram peaks.

4.1 Start-Length Diagrams

Consider a SE diagram $S = \{(s_i, e_i)\}_{i=1}^N$. The associated *SL diagram* is $S' = \{(s_i, e_i - s_i)\}_{i=1}^N$, where the x-coordinate corresponds to the start time of a repeated structure and the y-coordinate denotes the length of that repeat. While SE and SL diagrams encode the same information, SL diagrams emphasize the lengths of repeats. This transformation has also been applied to persistence diagrams for TDA applications [1].

4.2 Start(Normalized)-Length Diagrams

In most cases, normalizing SL diagrams before comparison proves to be more effective. We note that similar normalizations can also be applied to SE diagrams. The start(normalized)-length (S_NL) diagrams are defined as $S'_N = \{(\alpha(s_i/M), e_i - s_i)\}_{i=1}^N$, where α is a positive scaling factor and M is a normalization factor. Throughout this paper we will use $M = \max_i s_i$, but other normalizations may be applied. The vertical coordinate of S_NL diagrams are not normalized in order to maintain the emphasis on the lengths of the repeated structures.

Similar to SL diagrams, S_NL diagrams encode the lengths of repeated structures, except the start times in the normalized diagrams are proportional to the length of the song. This normalization acts as a kind of resampling by condensing start times of each song to be on the same scale, while also preserving the lengths of the found repetition patterns. The α parameter is a hyper-parameter that imbues a maximum tolerance on our comparisons. The impact of this parameter is left to Section 4.4.

4.3 Metrics for S_NL diagrams

As with SE diagrams, the bottleneck and Wasserstein metrics can be used to compare S_NL diagrams with one modification. Define $\tilde{\Delta}$ to be the set $\{(s, l) : s = 0, (s, l) \in \mathbb{R}_+^2\}$. The set $\tilde{\Delta}$ is the y -axis of SL and S_NL diagrams and it encodes repeats that start at zero. The S_NL p-bottleneck and S_NL (p,q)-Wasserstein metrics, \tilde{d}_B^p and $\tilde{d}_W^{p,q}$, are then the same as the p-bottleneck metric and (p,q)-Wasserstein metric defined in Section 3.1, except Δ is replaced by $\tilde{\Delta}$ in the definitions of P_∞^p and P_q^p , respectively. We use a modified version of Hera to implement these metrics [11].

Pairing unmatched points with repeated structures that start at zero rather than repeated structures that exist for no time allows the user to control the penalty for having unmatched points while maintaining the emphasis on the length of the repeated structures. To better understand this, consider a point $(s_*^1, e_*^1 - s_*^1) \in S_1$ that does not have a corresponding pair in S_2 under the optimal alignment of points in S_1 and S_2 . In this case we pair $(s_*^1, e_*^1 - s_*^1)$ with $\tilde{\Delta}$ so that $d_p((s_*^1, e_*^1 - s_*^1), \tilde{\Delta}) = |s_*^1|$. The penalty for unmatched points aligning with $\tilde{\Delta}$ consequently increases as the start time of the corresponding repeated structure increases. It is critical to note, however, that $s \in [0, \alpha]$

³ <http://kern.humdrum.org/search?s=t&keyword=Chopin>

	Threshold	0.01		0.02		0.03		0.04		0.05	
	Shingle	6	12	6	12	6	12	6	12	6	12
\bar{d}_B^∞ Metric	Precision	1.0	0.827	1.0	0.818	1.0	0.978	1.0	0.935	0.975	0.936
	Recall	0.788	0.827	0.769	0.865	0.788	0.865	0.750	0.827	0.750	0.846
$\bar{d}_W^{2,2}, \bar{d}_W^{\infty,2}$ Metric	Precision	1.0	0.833	0.974	0.975	1.0	0.976	0.976	1.0	0.976	1.0
	Recall	0.731	0.769	0.731	0.750	0.731	0.788	0.788	0.769	0.788	0.788

Table 2. Precision and recall values for mutual nearest neighbor matching using the distance between S_NL diagrams with $\alpha = 1$ corresponding to 104 Mazurkas. Note, we observe $\bar{d}_W^{\infty,2}$ and $\bar{d}_W^{2,2}$ to be equivalent when the optimal alignment only requires shifts in the start component so that $(|s_2 - s_1|^p + |e_2 - e_1|^p)^{\frac{1}{p}} = |s_2 - s_1| = \max(|s_2 - s_1|, |e_2 - e_1|)$.

for all start times s under the S_NL normalization. Thus, the penalty for having an unmatched diagram point is bounded above by α for S_NL diagrams with \bar{d}_B^p or $\bar{d}_W^{p,q}$. We further explain the choice of α in the following section.

4.4 Choosing α

The hyper-parameter α is a positive scaling factor in the normalization. For small α , the cost of aligning points with $\hat{\Delta}$ is low. For example, if $\alpha = 0.5$, then the cost of aligning a S_NL diagram point with $\hat{\Delta}$ is at most 0.5. Consequently, when comparing two S_NL diagrams, points within each diagram will be paired only when the difference between their lengths is negligible. Otherwise, it will be more effective to pair both points with $\hat{\Delta}$. Thus, a small value for α induces a strict matching criterion, where repeated structures are mostly shifted in the start coordinate to pair with a repeated structure of similar or equal length, and structures with unmatched lengths get paired to $\hat{\Delta}$.

The penalty for matching points with $\hat{\Delta}$ increases as α increases. For a large value of α , two S_NL diagram points of slightly different lengths are more likely to be matched with each other than with $\hat{\Delta}$. Consequently, a larger α value yields a more flexible length-based matching system.

The choice of α depends on the importance of the length of the repeated structures. One might consider $0 < \alpha \leq 1$ if repeated structures of different lengths are considered significantly dissimilar, or $\alpha \gg 1$ to allow for flexibility in length-based matchings. The inclusion of this parameter further adds to the flexibility of S_NL diagrams.

4.5 Applications of S_NL diagrams

We apply the same algorithm to the same dataset defined in Section 3.2 with SL and S_NL diagrams using the adapted bottleneck and Wasserstein metrics for the cover song task. Again, 10 experiment runs are performed per metric, varying the number of beats per audio shingle and the threshold applied to the SDM. For SL diagrams, the mean precision values across the 10 experiments are 0.791, 0.803, and 0.791 with \bar{d}_B^∞ , $\bar{d}_W^{2,2}$, and $\bar{d}_W^{\infty,2}$, respectively. The corresponding mean recall values are 0.596, 0.581, and 0.577.

Experiments on S_NL diagrams yield a significant increase in accuracy over both SL and SE diagrams on the cover song task, suggesting that a strict matching criterion in the length coordinate and flexibility in the start coordinate is the most accurate way to approach the cover song

task with these diagram representations. Across the 10 experiments, the ∞ -Bottleneck metric yields mean precision and recall values of 0.947 and 0.808, respectively. The (2,2) and $(\infty,2)$ -Wasserstein metrics yield a mean precision value of 0.971 and a mean recall value 0.763.

The experimental results for S_NL diagrams with $\alpha = 1$ are presented in Table 2. Separate analyses show that precision and recall remain constant for $0 < \alpha \leq 1$, and decrease monotonically as α increases above 1 for this data.

S_NL diagrams are not restricted to the standard start-normalization presented here. Applying the same method under the (2,2)-Wasserstein metric with a Chebyshev-normalized start component yields comparable results with slightly lower precision values and higher recall values. This further demonstrates the robustness of S_NL diagrams.

To push the limits of S_NL diagrams, we applied this representation to audio-based data without making any modifications in the preprocessing steps. We extracted beat-synchronous chroma feature vectors using `librosa` toolbox⁴ for a collection of performances of Mazurka⁵ and constructed the corresponding S_NL diagrams. Following the evaluation method in [2], we ranked the songs based on their pairwise-distances for the cover song task. While the mean average precision values were less than 0.1 across a range of metrics and α values, these results demonstrate that audio-specific preprocessing must be done in order to mitigate noise and other artifacts on tracks. Since there exist theoretical guarantees of stability of the S_NL diagrams, we are confident that with the appropriate preprocessing methods S_NL diagrams are suitable for a range of both score-based and audio-based music.

5. COMPARISON TO PREVIOUS WORK

Previous experiments on this Mazurka score dataset were done with the aligned hierarchies [13] and with the aligned sub-hierarchies (AsH) [12]. The metric for aligned hierarchies only allowed for pairwise comparisons between songs that were of the same length, meaning that it could only be used for the fingerprint task [13].

Initial experiments using AsH to address the cover song task were completed in [12]. Following the same exper-

⁴ <https://librosa.github.io/librosa/>

⁵ The CHARM Project Discography website maintains a list of commercially available Mazurka recordings at <http://www.mazurka.org.uk/info/discography/>

		Mean	Median	Min.	Max.
SE	Precision	0.847	0.873	0.622	1.0
	Recall	0.545	0.538	0.519	0.577
S _N L	Precision	0.963	0.976	0.818	1.0
	Recall	0.778	0.779	0.731	0.865
AsH	Precision	0.9511	0.9808	0.840	1.0
	Recall	0.771	0.754	0.692	0.882

Table 3. Summary statistics of the precision and recall values for mutual nearest neighbor matching of the Mazurka score data using SE diagrams, S_NL diagrams, and aligned sub-hierarchies (AsH). The AsH statistics are computed over the 10 combinations of thresholds and shingles, whereas the SE and S_NL statistics are computed over the 30 combinations of thresholds, shingles, and metrics. The S_NL experiments apply to a more complete dataset than the AsH and still attain similar high precision-recall values.

imental design varying the thresholds between 0.01 and 0.05 and testing shingle widths of 6 and 12, these experiments produced high precision rates (between 0.840 and 1.0) and modest recall rates (between 0.692 and 0.882).

Table 3 presents the summary statistics of the precision-recall values for mutual nearest neighbor matching using SE diagrams, S_NL diagrams, and AsH on the Mazurka score data. The AsH results are comparable to the precision-recall values of the S_NL experiments, while the SE experiments yield slightly lower precision-recall rates than both AsH and S_NL diagrams. However, the AsH post-processing technique requires repetitions to have enough repeated structure within them to build a smaller aligned hierarchies for these song sections. Consequently, as many as 68 songs (depending on the shingle size and threshold) do not have an AsH representation.

An advantage of the presented methods is that if a song has an aligned hierarchies representation then it has a SE diagram and S_NL diagram. Thus, the S_NL experiments work with a more complete dataset than the AsH experiments and still attain similar high precision-recall values.

6. DISCUSSION

Both the SE diagrams and the S_NL diagrams offer exciting contributions to the representation of music-based sequential data streams. These diagrams offer a clear representation of the relationships between repeated structural elements and have advantages over previous structure-based methods. By allowing for two recordings of different lengths to be directly compared without altering the beat-synchronized lengths of structural repeats, the S_NL diagram addresses an issue created by current resampling methods for music-based data streams.

The S_NL diagram provides a new method for resampling music-based data streams. The goal of resampling is to ensure that all matrix representations are the same size, which eases comparisons between music-based data streams. Current resampling methods compress all mu-

sical structures to represent a proportion of the length of the song, which results in comparing sections of a song that are proportionally the same length but not actually the same number of beats. In [10], the proportional comparisons of structural elements had issues comparing versions of Mazurka Op. 68, No. 4, which could be mitigated using the kind of resampling offered by S_NL diagrams.

Structure-based comparisons on resampled representations of a piece of music are then between proportions of the song instead of the true lengths of the repeats. This is especially an issue in cases where one artist plays a song once through, while another plays the piece through twice in its entirety. In such an example, a section of 100 beats long in the piece will look twice as long in the first representation when compared to the second representation.

The S_NL diagrams balance resampling all representations to be of the same length with maintaining the lengths of repeated structures. To accomplish this, the S_NL diagrams resample only the starting position of each repeated structure, while leaving the lengths alone. What is exciting about this innovative approach is that it not only allows for uniform comparisons – as desired by traditional resampling – but it also allows for comparisons between sections of the same length of time (or beats) instead of sections of the same proportional length of the song.

7. CONCLUSION

In this paper we present SE and S_NL diagrams, two novel, concisely defined, and flexible representations for music-based data. Leveraging theory from TDA, these diagrams come equipped with stable metrics which allow us to apply a mutual nearest neighbor search for the cover song task.

Experimental results demonstrate that SE and S_NL diagrams address the cover song task with high accuracy for score-based data, and these results are robust with respect to the choice of metric. Moreover, SE diagrams avoid resampling all together, while S_NL diagrams resample only the starting positions of repeated structures.

Overall, S_NL diagrams yield the highest accuracy in addressing the cover song task and they are more flexible. In addition to the choice of normalization, S_NL diagrams include the hyper-parameter α which allows the user to directly control the rigidity of the length-matching criterion.

In future work we plan to apply SE and S_NL diagrams to preprocessed audio data, and to extend these diagram representations so that they are suitable for machine learning tasks. Theoretical guarantees provide strong evidence that SE and S_NL diagrams will be applicable to both score and audio data after appropriate preprocessing. Beyond the method presented here, SE and S_NL diagrams can be mapped into spaces that are more suitable for machine learning tasks, just as, for example, persistence diagrams have been transformed to sequences of piecewise-linear functions and vectors in Euclidean space [1, 3]. Thus, SE and S_NL diagrams open up a range of new opportunities for applying machine learning methods through the lens of TDA to music-based tasks.

8. ACKNOWLEDGEMENTS

The first author is supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 1644760. The third and fourth authors are supported by the National Science Foundation under Grant No. DMS-1439786 and The Karen T. Romer Undergraduate Teaching and Research Awards while the authors were in residence at the Institute for Computational and Experimental Research in Mathematics in Providence, RI, during the Summer@ICERM 2017 program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors would like to thank Arnur Nigmatov for his generous help with adapting the Hera software for the S_NL diagrams.

9. REFERENCES

- [1] H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta, and L. Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017.
- [2] J. Bello. Measuring structural similarity in music. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2013–2025, 2011.
- [3] P. Bubenik. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16:77–102, 2015.
- [4] G. Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.
- [5] G. Carlsson, A. Zomorodian, A. Collins, and L. J. Guibas. Persistence barcodes for shapes. *International Journal of Shape Modeling*, 11(02):149–187, 2005.
- [6] F. Chazal, D. Cohen-Steiner, L.J. Guibas, F. Memoli, and S.Y. Oudot. Gromov-Hausdorff Stable Signatures for Shapes using Persistence. *Computer Graphics Forum*, 2009.
- [7] H. Edelsbrunner and J. L. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.
- [8] J. Foote. Visualizing music and audio using self-similarity. *Proc. ACM Multimedia 99*, pages 77–80, 1999.
- [9] M. Goto. A chorus-section detection method for musical audio signals and its application to a music listening station. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1783–1794, 2006.
- [10] P. Grosche, J. Serrà, M. Müller, and J.Ll. Arcos. Structure-based audio fingerprinting for music retrieval. *Proc. of 13th ISMIR Conference*, pages 55–60, 2012.
- [11] M. Kerber, D. Morozov, and A. Nigmatov. *Geometry Helps to Compare Persistence Diagrams*. 2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX).
- [12] K. M. Kinnaird. *Aligned Hierarchies for Sequential Data*. PhD thesis, Dartmouth College, 2014.
- [13] K. M. Kinnaird. Aligned hierarchies: A multi-scale structure-based representation for music-based data streams. *Proc. of 17th ISMIR Conference*, 2016.
- [14] B. McFee and D. P. W. Ellis. Learning to segment songs with ordinal linear discriminant analysis. In *International conference on acoustics, speech and signal processing*, ICASSP, 2014.
- [15] M. Müller. *Information Retrieval for Music and Motion*. Springer Verlag, 2007.
- [16] M. Müller, P. Grosche, and N. Jiang. A segment-based fitness measure for capturing repetitive structures of music recordings. *Proc. of 12th ISMIR Conference*, pages 615–620, 2011.
- [17] M. Müller and F. Kurth. Enhancing similarity matrices for music audio analysis. *Proc. of ICASSP*, 2006.
- [18] J. Paulus, M. Müller, and A. Klapuri. Audio-based music structure analysis. *Proc. of 11th ISMIR Conference*, pages 625–636, 2010.
- [19] C. Rhodes and M. Casey. Algorithms for determining and labelling approximate hierarchical self-similarity. *Proc. of 8th ISMIR Conference*, 2007.
- [20] C.S. Sapp. Online database of scores in the humdrum file format. *Proc. of 6th ISMIR Conference*, pages 664–665, 2005.
- [21] C. J. Tralie. Early MFCC and HPCP fusion for robust cover song identification. *Proc. of 18th ISMIR Conference*, 2017.
- [22] A. L. Wang. An industrial-strength audio search algorithm. In *Proc. of 4th ISMIR Conference*, 2003.

JSYMBOLIC 2.2: EXTRACTING FEATURES FROM SYMBOLIC MUSIC FOR USE IN MUSICOLOGICAL AND MIR RESEARCH

Cory McKay

Marianopolis College
cory.mckay@mail.mcgill.ca

Julie E. Cumming

McGill University
julie.cumming@mcgill.ca

Ichiro Fujinaga

McGill University
ichiro.fujinaga@mcgill.ca

ABSTRACT

jSymbolic is an open-source platform for extracting features from symbolic music. These features can serve as inputs to machine learning algorithms, or they can be analyzed statistically to derive musicological insights.

jSymbolic implements 246 unique features, comprising 1497 different values, making it by far the most extensive symbolic feature extractor to date. These features are designed to be applicable to a diverse range of musics, and may be extracted from both symbolic music files as a whole and from windowed subsets of them. Researchers can also use jSymbolic as a platform for developing and distributing their own bespoke features, as it has an easily extensible plug-in architecture.

In addition to implementing 135 new unique features, version 2.2 of jSymbolic places a special focus on functionality for avoiding biases associated with how symbolic music is encoded. In addition, new interface elements and documentation improve convenience, ease-of-use and accessibility to researchers with diverse ranges of technical expertise. jSymbolic now includes a GUI, command-line interface, API, flexible configuration file format, extensive manual and detailed tutorial.

The enhanced effectiveness of jSymbolic 2.2's features is demonstrated in two sets of experiments: 1) genre classification and 2) Renaissance composer attribution.

1. INTRODUCTION

The majority of research performed by musicologists, music theorists, music librarians and others focuses on symbolic music representations. Unfortunately, relatively few MIR-oriented software tools are available to assist such research, particularly with respect to research involving the increasingly large corpora being studied.

jSymbolic is an open-source Java framework designed to at least partially address this shortcoming. Its primary function is to extract a large number of features (statistical descriptors) from potentially huge collections of digi-

tally-represented symbolic music. These features can then be used to directly assist music researchers in analysis and search-based tasks, as well as in research incorporating machine learning.

Possible research applications include: empirical testing of existing musicological theories [11]; exploratory research that can reveal unexpected insights [11]; reconciling historical evidence with content-based data [12]; annotation of large corpora to allow content-based searches [10]; performing multimodal research by combining symbolic features with audio, textual and other features [9]; and generating novel music in specific styles by using feature values as stylistic guideposts [23].

jSymbolic 2.2 has been dramatically improved and expanded since its last properly published version (1.2) was released in 2010 [9]. It is also a component of the larger jMIR research software framework [9]. jSymbolic and the other jMIR components (including source code) can all be downloaded from [13].

2. RELATED RESEARCH

Surprisingly few frameworks designed specifically for extracting features from symbolic music have been published, although there are several MIR toolkits for analyzing symbolic music more generally. The MIDI Toolbox [6] is one particularly well-known system implemented in Matlab. The powerful music21 analysis toolkit [4] includes ports of 57 of the original jSymbolic 1.2 features, and also offers substantial additional useful functionality.

The Humdrum toolkit [8] is a well-known tool for analyzing music, although it does not extract features as such. The Melisma Music Analyzer [22] is another excellent analysis-oriented system, and `pretty_midi` [17] provides helpful creation, manipulation and extraction tools.

Additional work has been published where symbolic feature extraction is performed as a part of larger research projects, but where the feature extraction code has not itself been published. Standouts include [1] and [15]. Corrêa and Rodrigues have written a nice survey of related symbolic genre classification research [2].

To the best of our knowledge, there is no existing software that extracts anywhere near the number or diversity of features as jSymbolic, nor is there any with the same focus on broad accessibility and extensibility.



© Cory McKay, Julie E. Cumming, Ichiro Fujinaga.
Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Cory McKay, Julie E. Cumming, Ichiro Fujinaga. "jSymbolic 2.2: Extracting Features from Symbolic Music for use in Musicological and MIR Research", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

3. CHARACTERISTICS OF JSYMBOLIC

3.1 Features Extracted

jSymbolic 2.2 extracts 246 unique features, some of which are multidimensional, for a total of 1497 values (version 1.2 extracted 111 features and 1022 values). Details on the original musicological and music theoretical sources and motivations for the features are available in [9]. The features can be divided into eight general groups:

- **Pitch Statistics:** How common are various pitches and pitch classes relative to one another? How are they distributed and how much do they vary?
- **Melodic Intervals:** What melodic intervals are present? How much melodic variation is there? What can be observed from melodic contour measurements?
- **Chords and Vertical Intervals:** What vertical intervals are present? What types of chords do they represent? What kinds of harmonic movement are present?
- **Rhythm:** Information associated with note attacks, durations and rests, measured in ways that are both dependent and independent of tempo. Information on rhythmic variability, including rubato, and meter.
- **Instrumentation:** Which instruments are present, and which are emphasized relative to others? Both pitched and non-pitched instruments are considered.
- **Texture:** How many independent voices are there and how do they interact (e.g. parallel vs. contrary motion)? What is the relative importance of voices?
- **Dynamics:** How loud are notes and what kinds of variations in dynamics occur?
- **MEI-Specific:** Information that cannot be represented explicitly in MIDI (e.g. slurs) but can be in the Music Encoding Initiative (MEI) file format [16].

See Figure 1 for a complete list of the jSymbolic 2.2 features, including indications of which ones are new, as well as which ones are multidimensional.

These features are designed to be wide-ranging, in order to be applicable to a diverse range of musics from a variety of cultures, styles and time periods. A few features are intentionally partially redundant; for example, the Vertical Interval Histogram indicates the number of minor thirds and major thirds (among other things) separately, but the Vertical Thirds feature combines them. Such partial redundancies help highlight patterns in alternative ways to musicologists examining features. Also, some features are based on information explicitly (but not necessarily correctly) specified as metadata in the input files, such as meter or key, and others attempt to infer such information directly from the music itself.

3.2 Designing New Features

Extensibility and modularity are key priorities, as jSymbolic is intended to be a platform for developing and testing new features just as much as it is an out-of-the-box tool. New features can be added as plug-ins simply by extending an existing Java class, and it is easy to incorpo-

rate the values of existing features into new features in order to iteratively build new features of increasing sophistication. jSymbolic also automatically handles all infrastructure relating to feature dependencies and extraction scheduling. The overall design of the software is extensible, as is its configuration file format.

A tool has been added for exploring MIDI messages directly at a low-level, in order to help debug new features. jSymbolic also now automatically validates and error-checks new features as they are added, and there is substantial new general unit testing infrastructure.

3.3 Configuration Files

jSymbolic now includes a flexible configuration file format that can be used for batch processing, as a way of applying consistent settings across sessions and for keeping a record of settings used in individual experiments. These configuration files can be saved with the GUI, or they can be edited directly.

3.4 Avoiding Systematic Encoding Bias

One must always be careful that extracted features are not correlated with the source of data rather than its underlying musical content. This could happen, for example, in a corpus constructed by joining data from different sources, where each source uses different encoding conventions (e.g. different instrumentation designations for voices, or different interpretations of tempo markings). Such issues have been discussed regarding audio [21], but less so for symbolic data. Ideally, all data in a corpus would be systematically encoded in the same way, but this is rarely the case in practice.

jSymbolic therefore now includes functionality for generating “consistency reports.” These automatically check sets of symbolic music files for such biases.

An optional “safe” configuration file is also provided, which disables features associated with instrumentation, dynamics, microtonal pitches and tempo, as these tend to be particularly vulnerable to encoding bias. This is especially useful for musics where these qualities are typically unspecified, such as Renaissance music.

Many of jSymbolic 2.2’s new features are also specifically designed to avoid such biases. For example, many of the new rhythmic features are tempo-independent, so that they can be used even if tempo is source-correlated, while the old tempo-linked features can still be used if tempos are meaningfully and consistently encoded.

[3] presents a more detailed analysis of related issues, including empirical results produced with jSymbolic 2.2.

3.5 Windowed Extraction

Users can now perform windowed feature extraction with jSymbolic, with overlapping or non-overlapping windows, as well as extraction over entire pieces. Although common with audio, this ability to extract features separately from subsets of a piece is rare in the symbolic domain, and enables powerful new kinds of analysis.

Overall Pitch Statistics

(128) Basic Pitch Histogram
 (12) Pitch Class Histogram
 (12) Folded Fifths Pitch Class Histogram
Number of Pitches
Number of Pitch Classes
 Number of Common Pitches
Number of Common Pitch Classes
 Range
 Importance of Bass Register
 Importance of Middle Register
 Importance of High Register
 Dominant Spread
 Strong Tonal Centres
 Mean Pitch
Mean Pitch Class
 Most Common Pitch
 Most Common Pitch Class
 Prevalence of Most Common Pitch
 Prevalence of Most Common Pitch Class
 Relative Prevalence of Top Pitches
 Relative Prevalence of Top Pitch Classes
 Interval Between Most Prevalent Pitches
 Interval Between Most Prevalent Pitch Classes
 Pitch Variability
 Pitch Class Variability
Pitch Class Variability After Folding
Pitch Skewness
Pitch Class Skewness
Pitch Class Skewness After Folding
Pitch Kurtosis
Pitch Class Kurtosis
Pitch Class Kurtosis After Folding
 Major or Minor
 First Pitch
 First Pitch Class
 Last Pitch
 Last Pitch Class
 Glissando Prevalence
 Average Range of Glissandos
 Vibrato Prevalence
Microtone Prevalence

Melodic Intervals

(128) Melodic Interval Histogram
 Most Common Melodic Interval
 Mean Melodic Interval
 Number of Common Melodic Intervals
 Distance Between Most Prevalent Melodic Intervals
 Prevalence of Most Common Melodic Interval
Relative Prevalence of Most Common Melodic Intervals
 Amount of Arpeggiation
 Repeated Notes
 Chromatic Motion
 Stepwise Motion
 Melodic Thirds
Melodic Perfect Fourths
 Melodic Tritones
 Melodic Perfect Fifths
Melodic Sixths
Melodic Sevenths
 Melodic Octaves
Melodic Large Intervals
Minor Major Melodic Third Ratio
Melodic Embellishments
 Direction of Melodic Motion
 Average Length of Melodic Arcs
 Average Interval Spanned by Melodic Arcs
Melodic Pitch Variety

Chords and Vertical Intervals

(128) Vertical Interval Histogram
 (12) Wrapped Vertical Interval Histogram
 (11) Chord Type Histogram
Average Number of Simultaneous Pitch Classes
Variability of Number of Simultaneous Pitch Classes
Average Number of Simultaneous Pitches
Variability of Number of Simultaneous Pitches
 Most Common Vertical Interval
 Second Most Common Vertical Interval
 Distance Between Two Most Common Vertical Intervals
 Prevalence of Most Common Vertical Interval
 Prevalence of Second Most Common Vertical Interval
 Prevalence Ratio of Two Most Common Vertical Intervals
 Vertical Unisons
 Vertical Minor Seconds
 Vertical Thirds

Vertical Tritones
Vertical Perfect Fourths
Vertical Perfect Fifths
Vertical Sixths
Vertical Sevenths
Vertical Octaves
Perfect Vertical Intervals
Vertical Dissonance Ratio
Vertical Minor Third Prevalence
Vertical Major Third Prevalence
Chord Duration
Partial Chords
Standard Triads
Diminished and Augmented Triads
Dominant Seventh Chords
Seventh Chords
Non-Standard Chords
Complex Chords
Minor Major Triad Ratio

Rhythm

(2) Initial Time Signature
 Simple Initial Meter
 Compound Initial Meter
 Complex Initial Meter
 Duple Initial Meter
 Triple Initial Meter
 Quadruple Initial Meter
 Metrical Diversity
 Total Number of Notes
 Note Density per Quarter Note
 Note Density per Quarter Note per Voice
 Note Density per Quarter Note Variability
 (12) Rhythmic Value Histogram
 Range of Rhythmic Values
 Number of Different Rhythmic Values Present
 Number of Common Rhythmic Values Present
 Prevalence of Very Short Rhythmic Values
 Prevalence of Short Rhythmic Values
 Prevalence of Medium Rhythmic Values
 Prevalence of Long Rhythmic Values
 Prevalence of Very Long Rhythmic Values
 Prevalence of Dotted Notes
 Shortest Rhythmic Value
 Longest Rhythmic Value
 Mean Rhythmic Value
 Most Common Rhythmic Value
 Prevalence of Most Common Rhythmic Value
 Relative Prevalence of Most Common Rhythmic Values
 Difference Between Most Common Rhythmic Values
 Rhythmic Value Variability
 Rhythmic Value Skewness
 Rhythmic Value Kurtosis
 (12) Rhythmic Value Median Run Lengths Histogram
 Mean Rhythmic Value Run Length
 Median Rhythmic Value Run Length
 Variability in Rhythmic Value Run Lengths
 (12) Rhythmic Value Variability in Run Lengths Histogram
 Mean Rhythmic Value Offset
 Median Rhythmic Value Offset
 Variability of Rhythmic Value Offsets
 Complete Rests Fraction
 Partial Rests Fraction
 Average Rest Fraction Across Voices
 Longest Complete Rest
 Longest Partial Rest
 Mean Complete Rest Duration
 Mean Partial Rest Duration
 Median Complete Rest Duration
 Median Partial Rest Duration
 Variability of Complete Rest Durations
 Variability of Partial Rest Durations
 Variability Across Voices of Combined Rests
 (161) Beat Histogram Tempo Standardized
 Number of Strong Rhythmic Pulses - Tempo Standardized
 Number of Moderate Rhythmic Pulses - Tempo Standardized
 Num. Relatively Strong Rhythmic Pulses - Tempo Standardized
 Strongest Rhythmic Pulse - Tempo Standardized
 Second Strongest Rhythmic Pulse - Tempo Standardized
 Harmonicity of Two Strongest Rhythmic Pulses - Tempo Stand.
 Strength of Strongest Rhythmic Pulse - Tempo Standardized
 Strength of Second Strongest Rhythmic Pulse - Tempo Stand.
 Strength Ratio of Two Strongest Rhythmic Pulses - Tempo Stand.
 Combined Strength of 2 Strongest Rhyth. Pulses - Tempo Stand.
 Rhythmic Variability - Tempo Standardized
 Rhythmic Looseness - Tempo Standardized
 Polyrythms - Tempo Standardized

Initial Tempo
 Mean Tempo
 Tempo Variability
 Duration in Seconds
 Note Density
 Note Density Variability
 Average Time Between Attacks
 Average Time Between Attacks for Each Voice
 Variability of Time Between Attacks
 Average Variability of Time Between Attacks for Each Voice
 Minimum Note Duration
 Maximum Note Duration
 Average Note Duration
 Variability of Note Durations
 Amount of Staccato
 (161) Beat Histogram
 Number of Strong Rhythmic Pulses
 Number of Moderate Rhythmic Pulses
 Number of Relatively Strong Rhythmic Pulses
 Strongest Rhythmic Pulse
 Second Strongest Rhythmic Pulse
 Harmonicity of Two Strongest Rhythmic Pulses
 Strength of Strongest Rhythmic Pulse
 Strength of Second Strongest Rhythmic Pulse
 Strength Ratio of Two Strongest Rhythmic Pulses
 Combined Strength of Two Strongest Rhythmic Pulses
 Rhythmic Variability
 Rhythmic Looseness
 Polyrythms

Instrumentation

(128) Pitched Instruments Present
 (47) Unpitched Instruments Present
 (128) Note Prevalence of Pitched Instruments
 (47) Note Prevalence of Unpitched Instruments
 (128) Time Prevalence of Pitched Instruments
 Variability of Note Prevalence of Pitched Instruments
 Variability of Note Prevalence of Unpitched Instruments
 Number of Pitched Instruments
 Number of Unpitched Instruments
 Unpitched Percussion Instrument Prevalence
 String Keyboard Prevalence
 Acoustic Guitar Prevalence
 Electric Guitar Prevalence
 Violin Prevalence
 Saxophone Prevalence
 Brass Prevalence
 Woodwinds Prevalence
 Orchestral Strings Prevalence
 String Ensemble Prevalence
 Electric Instrument Prevalence

Texture

Maximum Number of Independent Voices
 Average Number of Independent Voices
 Variability of Number of Independent Voices
 Voice Equality - Number of Notes
 Voice Equality - Note Duration
 Voice Equality - Dynamics
 Voice Equality - Melodic Leaps
 Voice Equality - Range
 Importance of Loudest Voice
 Relative Range of Loudest Voice
 Relative Range Isolation of Loudest Voice
 Relative Range of Highest Line
 Relative Note Density of Highest Line
 Relative Note Durations of Lowest Line
 Relative Size of Melodic Intervals in Lowest Line
 Voice Overlap
 Voice Separation
 Variability of Voice Separation
 Parallel Motion
 Similar Motion
 Contrary Motion
 Oblique Motion
 Parallel Fifths
 Parallel Octaves

Dynamics

Dynamic Range
 Variation of Dynamics
 Variation of Dynamics in Each Voice
 Average Note to Note Change in Dynamics

MEI-Specific

Number of Grace Notes
 Number of Slurs

Figure 1. All features implemented by jSymbolic 2.2. Headings in bold refer to feature groups, not features. Features in italics are new (added since jSymbolic 1.2). Numbers in parentheses indicate the size of multi-dimensional features. Detailed descriptions of individual features are available in jSymbolic’s manual [14] and in its GUI.

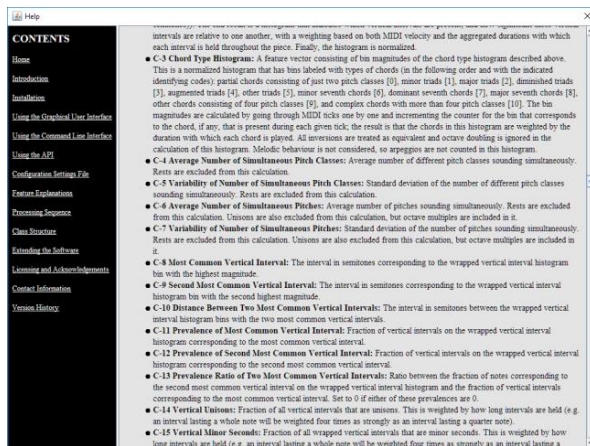


Figure 2. The new jSymbolic 2.2 manual.

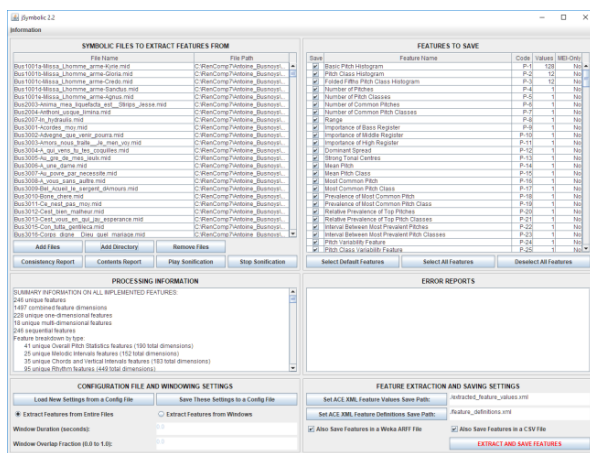


Figure 3. The redesigned jSymbolic 2.2 GUI.

3.6 I/O Formats and jMei2Midi

jSymbolic can extract features from music stored in either MIDI or MEI [16]. MIDI, despite its well-documented limitations, has the essential advantage that it can be read or generated by almost any symbolic music software, and also permits live performance encoding. This latter benefit makes MIDI compatible with non-Western (and Western) musics that do not conform to the quantized tunings and rhythms typical of common practice music notation and the symbolic music formats based on it. MIDI is also more suitable for transcribing audio, which also rarely conforms to strict quantization.

MEI, in turn, is a rich and extensible format that allows many kinds of important information to be represented that cannot be encapsulated with MIDI. jSymbolic’s new support for MEI is achieved via our custom-built Java MEI parser and MEI-to-MIDI converter called jMei2Midi, which can also be used as a standalone software library. jMei2Midi performs a more extensive level of MEI conversion than any other converter, and also maintains a channel for preserving and transmitting information that cannot be represented in MIDI.

Although jSymbolic cannot yet directly parse formats such as Music XML, OSC, Humdrum **kern or

LilyPond, there are fortunately many converters that can translate such formats to MIDI or MEI for jSymbolic feature extraction. jSymbolic’s Rodan [7] wrapper can already do this with Music XML.

Extracted features can now be saved as both Weka ARFF [24] files (a machine learning format) and as general-purpose CSV files. Previously, ACE XML [9] was the only output file format option.

3.7 Usability and Interfaces

It is crucial that jSymbolic be easy to learn and use for users with diverse technical backgrounds, and that it be easily adaptable to a broad range of use cases. This has been a primary focus of the upgrades since version 1.2.

jSymbolic now includes a detailed HTML manual (Figure 2) [14] and an extensive step-by-step tutorial that includes worked exercises with both jSymbolic and the Weka data mining framework [24]. jSymbolic’s Java implementation and lack of external dependencies make the software platform-independent and easy-to-install.

The original jSymbolic was only usable via a GUI, which has been substantially improved in version 2.2 (Figure 3). jSymbolic also now also includes command-line interface for batch processing, a well-documented Java API for programmatic access and a Rodan [7] workflow for those wishing to take advantage of distributed processing. New feedback on progress is provided as processing proceeds, and cleaner error handling and more detailed reporting in general have been instituted.

4. GENRE CLASSIFICATION EXPERIMENTS

4.1 Experimental Goals and Methodology

Our first set of experiments involved using the jSymbolic features to classify music by genre. This was done using the MIDI portion of our (balanced) “SAC” dataset [9], which consists of 250 pieces of music. SAC is divided into ten genres: Hardcore Rap, Pop Rap, Bop, Swing, Baroque, Romantic, Alternative Rock, Metal, Modern Blues and Traditional Blues. These genres can be combined pairwise into five parent genres: Rap, Jazz, Classical, Rock and Blues. This ontological structure permits one to evaluate how well a given approach can distinguish between both dissimilar genres (the five parent genres) and similar genres (the two classes comprising each pair).

Features were extracted from SAC using both the old jSymbolic 1.2 [9] and the new jSymbolic 2.2, in order to explore the effects of the new features. All implemented features were used, as no systematic encoding biases were found in the data (see Section 3.4).

The Weka machine learning framework [24] was used to perform 10-fold cross-validation experiments using its SMO support vector machine implementation (with default hyper-parameter settings). No dimensionality reduction pre-processing was applied, beyond what SMO does itself. This simple and generic classification methodology was chosen intentionally, as an important goal of this pa-

per is to emphasize the accessibility of jSymbolic’s features to music researchers who may have little or no background in machine learning. A more sophisticated approach would have been used if this were a paper specifically on classification.

4.2 Classification Results and Discussion

Corpus	jSymbolic 1.2		jSymbolic 2.2	
	Accuracy	F-score	Accuracy	F-score
SAC 5	90.4%	0.809	93.2%	0.872
SAC 10	75.6%	0.703	77.6%	0.631

Table 1. SAC (5-class and 10-class) genre classification accuracies and F-scores (averaged across 10 folds).

jSymbolic’s performance (Table 1) is quite impressive overall, especially since such basic machine learning techniques were used. Although there has not been a symbolic genre classification MIREX event in over a decade, the 2017 audio genre classification results [5] provide a rough general context: the highest classification accuracies were 75.9% in the 10-class Latin genre task, 76.8% in the 10-class popular genre task and 67.9% in the 7-class K-Pop genre task.

The new 2.2 features provided better classification accuracies than the old 1.2 features on both versions of SAC, by 2.8% and 2.0%. The F-score, however, declined for SAC 10, but improved for SAC 5.

The value of jSymbolic 2.2’s greatly expanded feature catalogue has a scope well beyond its classification performance gains. Many music researchers are interested in specifically what it is that differentiates various kinds of music, and a greater number of features make it possible to explore and understand music more thoroughly and precisely. This is revisited below.

5. COMPOSER ATTRIBUTION EXPERIMENTS

5.1 Experimental Goals and Methodology

The second set of experiments involved the same Weka-based classification methodologies described in Section 4.1. This time, however, the experiments involved Renaissance composer attribution; this is much more than a toy problem in early music studies, as there are many pieces whose composer is unknown or disputed, and feature-based machine learning holds significant potential for resolving such debates.

We constructed our (unbalanced) “RenComp7” dataset by combining the Josquin (top two Rodin security levels [19] only, based on historical sources), La Rue, Ockeghem, Busnoys and Martini data from [19] with John Miller’s Palestrina data and the Victoria data used in [20]. All files not already encoded as MIDI were converted. The resultant RenComp7 corpus consists of 1584 pieces.

An analysis of the data found that certain features were influenced by systematic encoding bias (see Section 3.4), namely those based on instrumentation, dynamics and tempo. Since including these features would have arti-

cially inflated performance, it was necessary to exclude certain features from consideration. As a result, only 335 of 1022 jSymbolic 1.2 feature values and 801 of 1497 jSymbolic 2.2 feature values were used.

We conducted one experiment where classification was performed among all seven RenComp7 composers. This was followed by two pairwise classifications that are of particular musicological interest: Josquin vs. La Rue (exact contemporaries who are musically similar) and Josquin vs. Ockeghem (from different generations).

5.2 Classification Results and Discussion

Corpus	jSymbolic 1.2		jSymbolic 2.2	
	Accuracy	F-score	Accuracy	F-score
All 7 Composers	87.9%	0.634	92.4%	0.715
Josq / Ockeghem	84.7%	0.818	92.6%	0.911
Josquin / La Rue	82.0%	0.771	86.3%	0.824

Table 2. RenComp7 composer attribution classification accuracies and F-scores (averaged across 10 folds).

The overall ability of the jSymbolic 2.2 features to distinguish between the composers (Table 2) is unprecedented in the automatic classification literature, and is all the more impressive given the simple machine learning methodology used. The excellent work of Brinkman et al. [1] provides the best basis for comparison: the authors used 53 features to classify 6 composers (J. S. Bach and five Renaissance composers), and obtained success rates of roughly 63% on average. Their approach did very well at discriminating Bach from the Renaissance composers (97%). This highlights both the quality of their approach and the particular difficulty of identifying Renaissance composers, and makes the success of the jSymbolic features on exclusively Renaissance music all the more encouraging. The new 2.2 features outperformed the old 1.2 features in all cases.

5.3 Diving into Features

As noted above, the relative performance of individual features can be at least as important in revealing musicological insights as overall classification performance. As an example of research along these lines, we asked two experts on Renaissance music, Julie E. Cumming and Peter Schubert, to predict what characteristics they thought would best differentiate the music of Josquin and Ockeghem, based on their extensive general experience studying the music of the two composers, and without any *a priori* exposure to the feature data. The jSymbolic feature data was then used to test these expectations. The results, as outlined in Figure 4, demonstrate how some of their predictions were indeed confirmed, but others were shown to be incorrect. This emphasizes the general need in musicology and music theory for empirical validation of a wide range of widespread beliefs and assumptions that have never been confirmed via systematic studies of large datasets. It is hoped that jSymbolic and similar software can help address this issue.

Empirical Testing of Expert Predictions of Characteristics More Evident in Ockeghem than Josquin

CONFIRMED: Less music for more than 4 voices

CONFIRMED: More 3-voice music

CONFIRMED: More triple meter

SAME: Less stepwise motion

SAME: More notes at the bottom of the range

SAME: More chords (or simultaneities) without a third

SAME: More varied rhythmic note values

OPPOSITE: More large leaps (larger than a 5th)

OPPOSITE: More dissonance

Figure 4. Results of empirical testing of expert predictions. “CONFIRMED” means the expectations were empirically correct, “SAME” indicates no statistically significant difference between the two composers and “OPPOSITE” means the expected characteristic was in fact more associated with Josquin than Ockeghem.

Next, in order to demonstrate the types of novel musical insights jSymbolic’s features can reveal, Weka was used to apply seven statistical feature analysis techniques (based on feature-class correlations, information gain, etc.) to highlight the features that most effectively distinguish the composers in each of the two composer pairs. The results were compiled into ranked feature lists.

It turns out that a combination of rhythmic characteristics are particularly important in distinguishing Josquin from Ockeghem and, furthermore, Ockeghem tends to have more vertical sixths and diminished triads, as well as longer melodic arcs. With respect to Josquin and La Rue, Josquin tends to have: more vertical unisons and thirds; fewer vertical fourths and octaves; and more melodic octaves.

6. CONCLUSIONS

jSymbolic is a powerful and accessible tool that music researchers can apply to diverse research areas and types of music. It can also serve as a platform that researchers can use to develop their own bespoke features. It is hoped that this will help address the paucity of symbolic music software produced by the MIR community to date, relative to the extensive range of software it has produced associated with audio and other data, and will encourage greater MIR engagement with musicologists and music theorists. jSymbolic’s easy-to-use interfaces and extensive documentation are intended to facilitate this.

Although jSymbolic features can certainly be used in classification tasks, as in the experiments described above, the direct study of feature values also has important potential. Such work can combine expert manual study with the use of statistical analysis techniques. Research can consist of empirical validation of existing hypotheses or of purely exploratory research, all involving the study of potentially huge quantities of music. Both approaches can help scholars arrive at initially unintuitive but potentially crucial musicological insights.

In terms of experimental conclusions, the results from Sections 4 and 5 permit the following observations:

- The new jSymbolic 2.2 features were quite effective in both genre and composer classification, even using generic machine learning approaches. They were able to distinguish between seven Renaissance composers 92.4% of the time, and achieved 93.2% genre classification accuracy when applied to a 5-genre ontology, and 77.6% when classifying amongst 10 genres.
- The new jSymbolic 2.2 features produced better classification accuracies than the old jSymbolic 1.2 features in all tests, and better F-scores in all but one.
- The new jSymbolic 2.2 features were effective in testing expert expectations about differences in the musical styles of pairs of Renaissance composers, and in revealing additional unanticipated differences.

7. FUTURE WORK

We will continue to work with musicologists and music theorists by helping them carry out research on large musical datasets with jSymbolic. We will also assist them in implementing specialized features of their own. A particular focus of this collaborative work will be placed on the determination of which features are most effective in distinguishing different musical classes (composers, genres, regions, etc.), and on investigating why. We will also expand our work on using machine learning to help resolve controversial composer attribution. We also intend to work on expanding the extent to which jSymbolic can extract features from non-Western musics by adding still more relevant features.

We are currently working on integrating jSymbolic2 into the SIMSSA/MIRAI architecture [10], so that researchers can search the project’s rich music databases using content-based queries formulated using feature values and ranges. A researcher could thus filter results based on the amount of chromaticism in a piece, for example, or the amount of parallel motion between voices. Of even greater interest, queries could potentially be formulated based on hard-to-quantify high-level characteristics, such as degree of tonality, made possible by machine learning models trained on jSymbolic features.

Related to this project, we also plan to apply jSymbolic to symbolic files that have been generated using optical music recognition or automatic audio transcription software, and to investigate the robustness of the features to such error-prone data. In addition to making an enormous amount of new music available for symbolic feature extraction, doing this successfully would also greatly facilitate multimodal research. The huge Lakh dataset [18] can also be studied with similar goals.

An additional priority will be to add new parsers so that features can be extracted from additional file formats. We are especially looking at adding features specially designed for music encoded using mensural or other early music notations. Finally, we intend to work towards porting jSymbolic2’s new features to other platforms, especially music21 [4].

8. ACKNOWLEDGEMENTS

We would like to thank the Social Sciences and Humanities Research Council of Canada (SSHRC) and the Fonds de recherche du Québec - Société et culture (FRQSC) for their generous funding. We would also like to acknowledge the formidable contributions of our many collaborators on the MIRAI and SIMSSA projects, especially Tristano Tenaglia.

9. REFERENCES

- [1] A. Brinkman, D. Shanahan and C. Sapp, "Musical stylometry, machine learning and attribution studies: A semi-supervised approach to the works of Josquin," *Proc. of the Biennial Int. Conf. on Music Perception and Cognition*, pp. 91–97, 2016.
- [2] D. C. Corrêa and F. A. Rodrigues, "A survey on symbolic data-based music genre classification," *Expert Systems with Applications*, Vol. 60, pp. 190–210, 2016.
- [3] J. E. Cumming et al., "Methodologies for creating symbolic corpora of Western music before 1600," *Proc. of the Int. Soc. for Music Information Retrieval Conf.*, accepted for publication, 2018.
- [4] M. S. Cuthbert, C. Ariza and L. Friedland, "Feature extraction and machine learning on symbolic music using the music21 toolkit," *Proc. of the Int. Soc. for Music Information Retrieval Conf.*, pp. 387–92, 2011.
- [5] J. S. Downie et al., "MIREX2017 Results - MIREX Wiki," Music-ir.org, 2016. [Online]. Available: http://www.music-ir.org/mirex/wiki/2017:MIREX2017_Results. [Accessed: 28-March-2018].
- [6] T. Eerola and P. Toiviainen, "MIR in Matlab: The MIDI Toolbox," *Proc. of the Int. Conf. on Music Information Retrieval*, pp. 22–7, 2004.
- [7] A. Hankinson, "Optical music recognition infrastructure for large-scale music document analysis," Ph.D. diss., Schulich School of Music, McGill Univ., Montreal, Canada, 2015.
- [8] D. Huron, "Music information processing using the Humdrum toolkit: Concepts, examples, and lessons," *Computer Music J.*, Vol. 26, No. 2, pp. 11–26, 2002.
- [9] C. McKay, "Automatic music classification with jMIR," Ph.D. diss., Schulich School of Music, McGill Univ., Montreal, Canada, 2010.
- [10] C. McKay and I. Fujinaga, "Building an infrastructure for a 21st-century global music library," *Int. Soc. for Music Information Retrieval Conf. Late Breaking and Demo Papers*, 2015.
- [11] C. McKay et al., "Using statistical feature extraction to distinguish the styles of different composers," *45th Medieval and Renaissance Music Conf.*, 2017.
- [12] C. McKay et al., "Characterizing composers using jSymbolic2 features", *Extended Abstracts for the Late-Breaking Demo Session of the Int. Soc. for Music Information Retrieval Conf.*, 2017.
- [13] C. McKay, "jMIR," SourceForge.net, SourceForge.net, 2018. [Online]. Available: <http://jmir.sourceforge.net>. [Accessed: 6-June-2018].
- [14] C. McKay, "jSymbolic Manual," SourceForge.net, 2018. [Online]. Available: http://jmir.sourceforge.net/manuals/jSymbolic_manual/home.html. [Accessed: 6-June-2018].
- [15] P. J. Ponce de León and J. M. Iñesta, "Statistical description models for melody analysis and characterization," *Proc. of the Int. Computer Music Conf.*, pp. 149–56, 2004.
- [16] P. Roland, "The music encoding initiative (MEI)," *Proc. of the First Int. Conf. on Musical Applications Using XML*, pp. 55–9, 2002.
- [17] C. Raffel, and D. P. W. Ellis, "Intuitive analysis, creation and manipulation of MIDI data with pretty_midi," *Int. Soc. for Music Information Retrieval Conf. Late Breaking and Demo Papers*, 2014.
- [18] C. Raffel. "Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching," Ph.D. diss., Columbia Univ., New York, USA, 2016.
- [19] J. Rodin, C. S. Sapp and C. Bokulich, "The Josquin Research Project," Stanford Univ., 2017. [Online]. Available: <http://josquin.stanford.edu>. [Accessed: 28-March-2018].
- [20] A. Sigler, J. Wild and E. Handelman, "Schematizing the treatment of dissonance in 16th-century counterpoint," *Proc. of the Int. Soc. for Music Information Retrieval Conference*, pp. 645–51, 2015.
- [21] B. L. Sturm, "A simple method to determine if a music information retrieval system is a 'horse'," *IEEE Trans. on Multimedia*, Vol. 16, No. 6, pp. 1636–44, 2014.
- [22] D. Temperley, *The cognition of basic musical structures*. Cambridge, MA: MIT Press, 2001.
- [23] E. Verdaguer Morales, "Anàlisi i generació algorísmica de línies de baix en estil Funk," Thesis, Pompeu Fabra Univ., Barcelona, Spain, 2014.
- [24] I. H. Witten, E. Frank and M. A. Hall, *Data mining: Practical machine learning tools and techniques*. New York: Morgan Kaufman, 2011.

RELEVANCE OF MUSICAL FEATURES FOR CADENCE DETECTION

Louis Bigo¹ Laurent Feisthauer¹ Mathieu Giraud¹ Florence Levé^{2,1}

¹ CRISTAL, UMR 9189, CNRS, Université de Lille, France

² MIS, Université de Picardie Jules Verne, Amiens, France

{louis, laurent, mathieu, florence}@algonus.fr

ABSTRACT

Cadences, as breaths in music, are felt by the listener or studied by the theorist by combining harmony, melody, texture and possibly other musical aspects. We formalize and discuss the significance of 44 *cadential features*, correlated with the occurrence of cadences in scores. These features describe properties at the arrival beat of a cadence and its surroundings, but also at other onsets heuristically identified to pinpoint chords preparing the cadence. The representation of each beat of the score as a vector of cadential features makes it possible to reformulate cadence detection as a classification task. An SVM classifier was run on two corpora from Bach and Haydn totaling 162 perfect authentic cadences and 70 half cadences. In these corpora, the classifier correctly identified more than 75% of perfect authentic cadences and 50% of half cadences, with low false positive rates. The experiment results are consistent with common knowledge that classification is more complex for half cadences than for authentic cadences.

1. INTRODUCTION

1.1 Cadences

Music, like all languages, is organized into structural units. In Western tonal music, these units often end with strong harmonic formulas called *cadences*, from the Latin *cadere*, “to fall.” Despite their structural function, cadences are hard to define. Based on a review of dozens of music theory papers, Blombach defined the cadence as “any musical element or combination of musical elements, including silence, that indicates relative relaxation or relative conclusion in music” [3]. This definition highlights the way a listener (whether musically trained or not) can *hear* the presence of a cadence by feeling that the music “breaths”. A cadence is generally characterized by *local* musical elements, such as a specific harmonic progression and a falling melody. However, these elements do not necessarily imply a cadence. A global or high-level structure such as the sonata form may also induce the impression of a cadence [10].

Cadences are usually classified by harmonic progression. The *authentic* cadence is characterized by a *dominant* harmony (notated V) followed by a *tonic* harmony (notated I). In the American terminology, when both chords are in root position and the melody ends on the tonic, the authentic cadence is said to be *perfect* (PAC), otherwise it is *imperfect* (IAC). If the IAC is in root position (but the melody does not end on the tonic), it is said to be a *rooted* IAC (rIAC). The *half* cadence (HC) ends with a dominant harmony, generally in root position. The *deceptive* cadence (DC) is an authentic cadence where the expected final tonic is replaced by another harmony (often VI). Some authors theorize the *evaded* cadence as a particular IAC, while others see it as a DC-like progression but including a melodic break, for instance while repeating a phrase [19]. Some scholars do not consider the *plagal* progression IV/I as a cadence but rather as a post-cadential prolongation [4].

Each cadence type provide a different feeling of closure. The strongest cadence is the PAC, followed in turn by the rIAC, IAC, HC, and the DC and related cadences [20]. Some traditions consider the rIAC to be very conclusive. For instance, French music teachers refer to both PAC and rIAC as *cadence parfaite*. Using a preparation chord before the dominant chord, generally a *subdominant harmony* (SD, that is II, IV, or V/V), strengthens the salience of a PAC/rIAC. In contrast, DC and related cadences renew tension, extending the musical phrase and delaying closure until a more conclusive cadence is used.

1.2 Cadences, Musicology and MIR

Modeling cadences is a current challenge in musicology [15]. Although cadence definitions found in music education textbooks are often quite short, music theorists agree on the difficulty to define cadences because of the variety of their realizations observed in the repertoire [4].

Cadences are therefore usually studied within the frame of one specific corpus – see for example Martin and Pedneault-Deslauriers’s study of HC in Mozart’s piano sonatas [14]. However, more systematic analyses of large corpora would help to understand the evolution of compositional choices over time. Rohrmeier and Neuwirth suggested a first characterization using grammars, based on the degrees and the bass line [18].

Detecting cadences throughout the score requires a specific training to find clues pointing out to the breaths in music. Can we algorithmically detect cadences from a score encoded in symbolic notation? Some works in MIR have



© Louis Bigo, Laurent Feisthauer, Mathieu Giraud, Florence Levé. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Louis Bigo, Laurent Feisthauer, Mathieu Giraud, Florence Levé. “Relevance of musical features for cadence detection”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

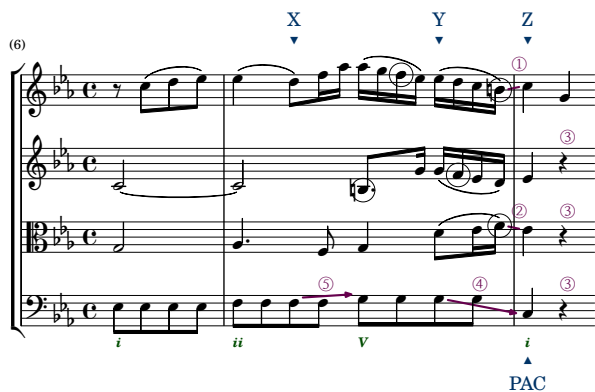


Figure 1. Haydn, op. 17/4, iv, PAC at measure 8 (offset Z). Compare to Figure 1 of [21]. Features describe here the constitution of the Z chord (*Z-in-perfect-triad*, *Z-in-perfect-triad-or-sus4*, *Z-highest-is-1*), voice leading to Z (*Z-1-comes-from-7* ①, *Z-3-comes-from-4* ②), rests after Z (*R-after-Z-rest-lowest*, *-middle* ③) and the metric structure (*R-Z-strong-beat*). Features also describe relations with chord Y (*Y-Z-bass-moves-compatible-V-I* ④, *Y-Z-bass-same-voice*), and cadence preparation (*X-Y-bass-moves-2nd-Maj* ⑤).

Note that the heuristic choice of a single offset Y implies here that the features *Y-in-V7-3* and *Y-has-7* are not true, even if the dominant chord actually contain several pitches 3 and 7 (circled notes). Nevertheless, these pitches are caught by the tonality features (*Z-bass-compatible-with-I*, *Z-bass-compatible-with-I-scale*) and some of them are considered by the voice leading features (①, ②).



Figure 2. Haydn, op. 17/5, i, HC at measure 8. Features notably describe here a voice leading (*Z-6-comes-from-7*, *Z-4-comes-from-5*, *Z-1-comes-from-2* ①) continued by a 6/4 suspension (*Z-6-moves-to-5*, *Z-4-moves-to-3* ②). Other features also describe the tonality compatibility of an HC (*bass-Z-compatible-with-V*, circled notes, but both *Z-bass-compatible-with-I* and *Z-bass-compatible-with-I-scale* are also true due to the squared *c#*) as well as bass movements (*Y-Y-bass-moves-chromatic*, *Y-Z-bass-moves-2nd-Maj* ③) and the metric structure (*R-Z-strong-beat*).

focused on melodic cadences [25] and a few studies have tackled the problem of the identification of harmonic progressions [16] or their representation as musical trajectories [1]. The authors of [7] took Rohrmeier's works further, extending it into a system deriving harmonic relations between chords, where grammars rules were inferred for jazz harmony. Currently, only a few algorithms recognize simple cadences [12]. We previously suggested a rule-based detection of PAC/rIAC in fugues [9] and used it in a study on the sonata form [2]. Recently, Sears and colleagues [22] used the software IDyOM [17] on a corpus of Haydn string quartets to show that music predictability increases at cadential points and decreases on the following note.

1.3 Contents

Our goal is to identify *binary*, *musical*, and *local* features that coincide with cadences and that can be used to train a model that detects new cadences, either PAC/rIAC or HC. Rather than agnostically discovering cadential features on the musical surface, we intend here to confirm and study traditional music theory knowledge regarding cadences. The proposed strategy avoids chord segmentation, which is itself a difficult MIR problem. Section 2 details the selected features and Section 3 describes the learning process. Finally, Sections 4 and 5 discuss the application of the method on Bach and Haydn corpora.

2. MUSICAL FEATURES AT THREE ONSETS

Each beat *Z* of the score is considered as the potential arrival point of a cadence. A set of 44 binary features is computed at each beat. These features are then used to train a classifier whose aims to predict whether a beat corresponds to the arrival point of a cadence or not. The features aim at detecting cadences at a *local* level, i.e. the surroundings of the cadential beat including its immediate past, presumably corresponding to the *preparation* of the cadence. The idea is to try and detect SD-V-I progressions for a PAC/rIAC, and progressions ending with V for an HC.

What we propose here is a simple heuristic focusing on three specific onsets: *Z*, *Y(Z)* and *X(Z)*, or for short *Z*, *Y*, and *X*. Most of the features describe sets of notes sounding at these onsets (even when they begin before), namely *chord(Z)*, *chord(Y)*, and *chord(X)*. We therefore do not start from a complete harmony analysis nor a chord segmentation, that can be error-prone. Even when the methods finding *Y(Z)* and *X(Z)* return approximate onsets, the computed features may be relevant.

2.1 Features on the Arrival Point Z or around it

The arrival chord of a cadence is usually a perfect triad, possibly with some suspensions. A first set of features describes this chord and its immediate neighborhood:

- *Z-in-perfect-major-triad* (respectively *Z-in-perfect-triad*): *chord(Z)* is included in $\{1, \underline{3M}, \underline{5}\}^1$ (resp. $\{1, \underline{3m}, \underline{3M}, \underline{5}\}$)

¹ Pitches in underlined figures (i.e. $\underline{1}$, $\underline{3}$, etc.) are here computed by the interval modulo octave relative to the bass. As some chords are not in root

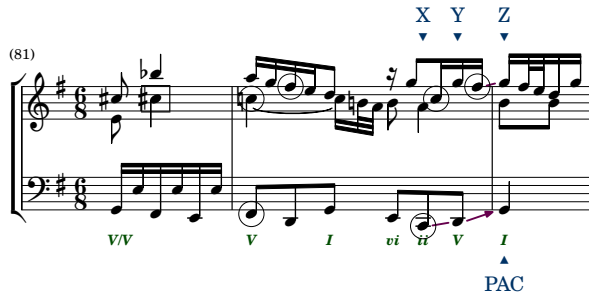


Figure 3. Bach, fugue #15 in G major BWV860, PAC at measure 83. Voice leading and highlighted notes as in the Figures 1 and 2. To cope with the faster harmonic rhythm, every eighth before Z is considered as a potential Y .

- *Z-in-perfect-triad-or-sus4*: $chord(Z)$ is included in $\{1, 3, 4, 5\}$
- *Z-is-sus4*: $chord(Z)$ is exactly $\{1, 4, 5\}$
- *Z-highest-is-1* (resp. *Z-highest-is-3*): The highest note of $chord(Z)$ is the tonic 1 (resp. the major or minor third 3), as expected for a PAC (rIAC)

Another set of features describes *voice leadings* from preceding notes (see list on Table 2). *Z- β -comes-from- α* means that the note β in $chord(Z)$ is an “immediate resolution” of a note α (the interval being still relative to the bass of Z) that is *exactly before* β (even if this note is not at the onset Y that will be defined later). For example, *Z-3-comes-from-4* means that there is a 3 in $chord(Z)$ that is an immediate resolution of a 4 (dominant seventh in the case of a PAC, see ② on Figure 1).

There can also be a suspension at the arrival point, as on the HC on Figure 2. We thus add symmetrical features *Z- α -moves-to- β* (see list on Table 2). For example, *Z-4-moves-to-3* means that there is a suspended fourth 4 in $chord(Z)$ that is immediately resolved to the third 3 .

Finally, features try to grasp the tonality in the neighborhood of Z . We do not perform tonality estimation [13, 23] because of the usual difficulty of algorithms to disambiguate adjacent tonalities in the circle of fifths.

- *Z-bass-compatible-with-I* (resp. *Z-bass-compatible-with-V*): Both notes 4 and 7 of the tonality that would be implied if the bass of Z is I (resp. V) are present in the four beats before Z
- *Z-bass-compatible-with-I-scale*: The 8 previous beats exhibits the whole scale of the same implied tonality – Temperley suggesting that such PACs with SD before $V-I$ feel more conclusive [24].

For example, on the PAC of Figure 1, *Z-bass-compatible-with-I* and *-with-I-scale* are true (and not *-with-V*), and on the HC of Figure 2, *Z-bass-compatible-with-V* is true. However, these features may be triggered by

position, these pitches may differ from the actual function. For example, the top voice d on offset X on Figure 1 is the sixth 6 of the chord $II6$ but is actually the tonic of the II chord.

other events close on the circle of fifths: Both *Z-bass-compatible-with-I* and *-with-I-scale* may be triggered by a previous V/V (as on Figure 2) or, in minor, when Z is actually a III in root position.

2.2 Rhythmic Features around the Arrival Point Z

These textural features intend to detect either breaks or continuation in music.

- *R-Z-strong-beat*: Z is a strong beat (beat 1 and 3 for 4/4, and beat 1 for other time signatures)
- *R-Z-same-rhythm-1* (resp. *R-Z-same-rhythm-2*): There is exactly the same sequence of durations in the one (resp. the two) beat(s) preceding Z than on the one (resp. the two) beat(s) at onset Z
- *R-Z-sustained-note*: At least one note sounding at Z started before Z
- *R-after-Z-rest-highest*, *R-after-Z-rest-lowest*, *R-after-Z-rest-middle*: There is a rest in some voice right after the note at onset Z (see Figure 1)
- *R-after-Z-one-voice-ends*: Z is the last onset in at least one voice (end of the piece)

2.3 Features on the Point Y or around it

For each arrival beat Z , we identify a point Y prior to Z supposed to pinpoint the chord “preceding” Z . For example, identifying $chord(Y)$ as a dominant chord is a sign indicating a potential PAC at Z . Although V chords generally span over more than one beat, associating Y with a single beat eases the computation of features.

We thus propose to identify the point Y as the latest beat preceding Z for which the bass voice includes a sounding note, limited to one measure in the past. If the bass includes a rest just before Z , we look just before. The usual time span corresponding to the preparation of a cadence depends on the *harmonic rhythm* and varies among musical styles. The beat resolution to search the Y point should therefore depend on the corpus.

- *Y-Z-offsets-at-most-1*: Y is at most one quarter note before Z

Some features are concerned with $chord(Y)$:

- *Y-has-7* (resp. *Y-has-9*): $chord(Y)$ contains 7 (resp. 9), that is the leading tone (resp. the dominant seventh or the dominant ninth) in the case of a candidate PAC
- *Y-in-V7* : $chord(Y)$ is included within a dominant seventh chord
- *Y-in-V7-3* : $chord(Y)$ is included within a dominant seventh chord and contains a third

Other features focus on bass moves:

- *Y-Y-bass-moves-8ve*: The bass note preceding Y is at the same pitch but with an octave jump (expected on some V or $V64$ chords)

		pieces	voices	beats	PAC (final)	rIAC	HC
haydn-quartets	Haydn string quartets [22]	42 expositions	4	7173	99 (21)	(8)	70
bach-wtc-i	Bach fugues [9]	24 fugues	2 to 5	4739	63 (23)	24	(5)

Table 1. Corpora with manual annotations of cadences. Cadences are labeled at about 2% of the beats. We narrow to sets with significant number of cadences (PAC and HC for the Haydn corpus, PAC and PAC+rIAC for the Bach corpus).

- *Y'-Y-bass-moves-chromatic*: The bass note preceding Y is at a distance of one semitone (HC)
- *Y-Z-bass-moves-2nd-min* (resp. *Y-Z-bass-moves-2nd-Maj*)
- *Y-Z-bass-same-voice*: Bass notes of both chords are on the same voice
- *Y-Z-bass-moves-compatible-V-I* (resp. *Y-Z-bass-moves-compatible-I-V*): The bass moves by an ascending fourth or descending fifth (PAC) (resp. ascending fifth or descending fourth, HC I-V)

2.4 Features on the Cadence Preparation (Point X)

We identify the onset X as the latest beat before Y whose lowest sounding note has a different pitch (modulo octave) than the lowest note of Y. Features focus on this bass move:

- *X-Y-bass-moves-2nd-min* (V/V-V-I)
- *X-Y-bass-moves-2nd-Maj* (IV-V-I or II6-V-I)
- *X-Y-bass-moves-4th* (expected in II-V-I)

3. CLASSIFICATION PROCESS

A model is built in order to reflect the correlation between the features listed in Section 2 and the occurrences of cadences in corpora. These corpora bear manual annotations indicating the position of PAC, rIAC and HC. Assuming that the arrival points of cadences do not fall between beats, each beat (quarter note, or three eighths depending on the time signature) of each piece is described by:

- a *vector of boolean values* corresponding to the set of features and computed from the musical score,
- a boolean *class* specifying whether the beat is annotated in the reference as a PAC/rIAC/HC or not.

This way of representing data enables us to reformulate cadence detection as a *classification task*. To avoid overfitting, each dataset is randomly divided into two subsets: a *training set* used to train a classifier and a *test set* left to evaluate the classifier performance at the end. The classifier and the value of its hyper-parameters have been selected by performing *Leave-One-Piece-Out cross-validation* over the training set. This is done by evaluating the classification on each piece of the training set by a classifier trained on the remaining pieces of the training set. Indeed, the traditional *Leave-One-Out (LOO)* cross-validation approach that would consist in leaving only *one beat* of one piece out of the training set would result here in overfitting due to intra-piece musical repetitions.

4. EXPERIMENTS AND DISCUSSION

4.1 Corpora and Implementations

Table 1 shows the corpora which was used in this study. The corpus *bach-wtc-i* includes the 24 fugues of the first book of the Well-Tempered-Clavier by J.-S. Bach. Cadence annotations were taken from our previous work [9]. The corpus *haydn-quartets* includes 42 expositions from movements of Haydn string quartets in sonata form, annotated with cadences by Sears and colleagues [22]. Even if these annotated corpora model cadences in the light of a global analysis of the form, we have used them as a benchmark on our local feature-based detection. Only a minority of annotated PAC are *final* in the sense that they are included in the last four measures of the piece (or of the exposition).

Pieces were downloaded as voice-separated *.krn* files from *kern.ccarh.org* [11]. Note that the features proposed here could also apply to non-separated files, except for *after-Z-rest-** and *Y-Z-bass-same-voice*. In this case, features on voice leading would only check that the coming note or the suspended note is found at the right place in the polyphonic texture.

For each beat *Z* (and their related onsets *X*, *Y*), the features described in Section 2 are extracted using code based on the Python framework *music21* [6]. Points *Y* and *X* are searched at a beat resolution of a quarter note (Haydn) or eighth note (Bach, see Figure 3). Classifiers were computed thanks to the *scikit-learn* framework [8].

4.2 Discussion on Feature Statistics

Table 2 shows tallies of features, their correlation with cadences as well as an estimation of their significance. Many features are significant in both corpora, despite differences in musical style. Unsurprisingly, features *R-Z-strong-beat*, *Y-Z-bass-moves-compatible-V-I*, *Z-perfect-triad-or-sus4* and *Z-highest-note-is-1* are activated nearly for every PAC. Note that PAC lacking the fifth leap are the ones where the bass passes by another note before tonic resolution. Rhythmic and break features are also quite significant. Some features differ between corpora. For example, *R-Z-sustained-note* is absent in nearly all PACs in the Haydn corpus, whereas it can be found in some PACs in Bach fugues due to the contrapuntal writing.

We were expecting to find more *suspensions* for both PAC and HC as a way to retain tension before the ultimate resolution but they do not appear significantly in these corpora. We also notably lack strong significant features for HC. Indeed, the *Y-Z bass move* in a HC is variable (it is typically similar to *X-Y moves* in PAC).

	Features	bach-wtc-i			haydn-quartets		
		beats	PAC	rIAC	beats	PAC	HC
Rhythmic features R	<i>R-Z-strong-beat</i>	1920	60* / 25	24* / 9	3126	98* / 43	70* / 30
	<i>R-Z-same-rhythm-1</i>	394	1 / 5	· / 1	1254	2* / 17	2* / 12
	<i>R-Z-same-rhythm-2</i>	176	· / 2	· / 0	448	0 / 6	1 / 4
	<i>R-Z-sustained-note</i>	2341	14* / 31	5 / 11	2521	1* / 34	8* / 24
	<i>R-after-Z-rest-highest</i>	166	14* / 2	1 / 0	501	56* / 6	10 / 4
	<i>R-after-Z-rest-middle</i>	477	22* / 6	9* / 2	1227	72* / 16	35* / 11
	<i>R-after-Z-rest-lowest</i>	194	15* / 2	13* / 0	1130	59* / 15	34* / 11
	<i>R-after-Z-one-voice-ends</i>	180	19* / 2	2 / 0	·	· / 0	· / 0
Arrival point Z	<i>Z-in-perfect-major-triad</i>	1167	43* / 15	12 / 5	2760	94* / 38	53* / 26
	<i>Z-in-perfect-triad</i>	1819	56* / 24	19* / 9	3256	97* / 44	53* / 31
	<i>Z-in-perfect-triad-or-sus4</i>	2078	62* / 27	20* / 10	3434	97* / 47	55* / 33
	<i>Z-is-sus4</i>	680	20* / 9	1 / 3	1308	14 / 18	4 / 12
	<i>Z-highest-is-1</i>	592	55* / 7	1 / 2	1765	96* / 24	19 / 17
	<i>Z-highest-is-3</i>	1488	1* / 19	21* / 7	1596	1* / 22	28* / 15
	<i>Z-bass-compatible-with-I</i>	1724	63* / 22	23* / 8	2279	98* / 31	56* / 22
	<i>Z-bass-compatible-with-V</i>	1265	8 / 16	4 / 6	1616	3* / 22	44* / 15
	<i>Z-bass-compatible-with-I-scale</i>	1902	63* / 25	22* / 9	2104	91* / 29	46* / 20
	<i>Z-1-comes-from-7</i>	663	52* / 8	15* / 3	1016	89* / 14	30* / 9
	<i>Z-1-comes-from-1</i>	180	13* / 2	1 / 0	828	9 / 11	0* / 8
	<i>Z-1-comes-from-2</i>	523	23* / 6	7 / 2	893	65* / 12	27* / 8
	<i>Z-3-comes-from-4</i>	1078	25 / 14	16* / 5	1488	72* / 20	45* / 14
	<i>Z-4-comes-from-5</i>	197	4 / 2	· / 0	291	· / 4	9 / 2
	<i>Z-5-comes-from-5</i>	153	9* / 2	· / 0	769	2 / 10	13 / 7
	<i>Z-5-comes-from-6</i>	510	1 / 6	2 / 2	495	0 / 6	9 / 4
	<i>Z-6-comes-from-7</i>	200	· / 2	· / 1	130	· / 1	· / 1
	<i>Z-2-moves-to-1</i>	57	· / 0	· / 0	90	2 / 1	1 / 0
	<i>Z-4-moves-to-3</i>	160	2 / 2	1 / 0	340	2 / 4	11* / 3
	<i>Z-6-moves-to-5</i>	138	1 / 1	· / 0	180	· / 2	8* / 1
	<i>Z-7-moves-to-1</i>	7	· / 0	· / 0	105	2 / 1	1 / 1
Point Y	<i>Y-in-V7</i>	1267	52* / 16	17* / 6	3290	81* / 45	15* / 32
	<i>Y-in-V7-3</i>	721	44* / 9	14* / 3	2413	69* / 33	14 / 23
	<i>Y-has-7</i>	554	22* / 7	7 / 2	767	66* / 10	8 / 7
	<i>Y-has-9</i>	607	1 / 8	4 / 3	486	2 / 6	5 / 4
	<i>Y-Z-offsets-at-most-1</i>	4525	63 / 60	24 / 22	5668	90 / 78	66* / 55
	<i>Y-Z-bass-same-voice</i>	4270	63 / 56	24 / 21	5297	98* / 73	70* / 51
	<i>Y-Z-bass-moves-2nd-min</i>	1313	0* / 17	0* / 6	1328	1* / 18	35* / 12
	<i>Y-Z-bass-moves-2nd-Maj</i>	880	0* / 11	· / 4	559	0* / 7	28* / 5
	<i>Y-Z-bass-moves-compatible-I-V</i>	125	1 / 1	· / 0	448	2 / 6	6 / 4
	<i>Y-Z-bass-moves-compatible-V-I</i>	512	62* / 6	23* / 2	578	95* / 7	6 / 5
	<i>Y'-Y-bass-moves-chromatic</i>	1139	6 / 15	2 / 5	2050	10* / 28	33 / 20
	<i>Y'-Y-bass-moves-8ve</i>	193	29* / 2	7* / 0	522	22* / 7	6 / 5
Point X	<i>X-Y-bass-moves-2nd-min</i>	433	2 / 5	1 / 2	1060	10 / 14	10 / 10
	<i>X-Y-bass-moves-2nd-Maj</i>	568	25* / 7	12* / 2	803	65* / 11	5 / 7
	<i>X-Y-bass-moves-4th</i>	670	11 / 8	4 / 3	1626	4* / 22	9 / 15
Total		4739	63	24	7173	99	70

Table 2. Feature tallies for PAC (both corpora), rIAC (Bach corpus) and HC (Haydn corpus). The table shows, for each feature, the number of beats where this feature occurs (*all* beats, cadential points or not), followed by its number of occurrences on beats labeled as cadences in the reference annotation, as well as, in small, its expected number should the feature be random and uniformly distributed across the beats. (· means 0, and not significant). For example, there are 70 HC out of 7173 beats in the Haydn quartets corpus. There are 35 beats corresponding to a HC with the feature *Y-Z-bass-2nd-min*, out of 1328 beats with this feature, and compared to only 12 beats should this feature be random.

For each feature and each cadence type, p -values are estimated by an exact Fisher test computed by the Python *scipy* package. Fisher tests are computed independently. To account for the large number of tests, only features with p -values under .001 (**bold**, *) can be considered as significant, either by their absence (*italic*) or their presence. For example, the feature *Y-Z-bass-2nd-min* is significantly absent in PACs of both corpora ($p < 10^{-7}$) and significantly present in HCs of the Haydn corpus ($p < 10^{-8}$).

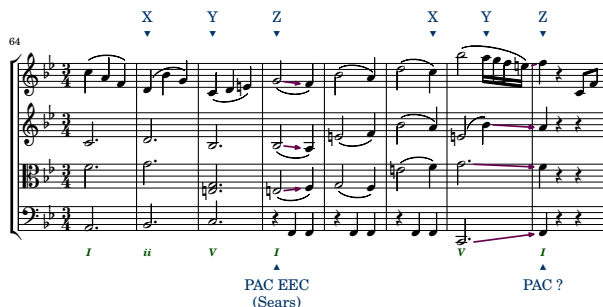


Figure 4. Haydn, op. 55/3, i, potential PACs at m67 and m71. The PAC at m67 is hard to detect with the silence at the bass. In their global analysis of the form, Sears et. al see the end of the secondary theme (called the EEC, for *Essential Expositional Closure* by [10]) at m67 and discard any further PAC in the following concluding section [22]: The PAC candidate at m71, found by the proposed strategy, is thus counted here as a FP. It could be debated whether the EEC is indeed at m67 (first cadential I, but weakened by the bass rest) or rather at m71 (cadential feeling augmented by the following rests and bass note on upbeat, m67 considered as an evaded cadence).

4.3 Learning Process

A linear Support Vector Machine (SVM) classifier was trained on each training set as explained in Section 3, splitting the feature space with a hyperplane [5]. As datasets are *unbalanced* (about 98% of the beats are “non-cadential”), we assigned stronger weights to data belonging to the under-represented class, here the cadential beats. Other classifying algorithms such as *k*-nearest-neighbor or decision trees were tested and turned out to provide comparable or inferior results.

4.4 Discussion on Detection Results

Table 3 shows the comparison between the predictions of each classifier on the test set of each corpus and the reference annotations. The detection of PAC is good, with more than 75% PAC detected and a low false positive rate ($< 1\%$). Note that we previously reported 82% of PAC detection in fugues [9] but with manual hard-coded rules that may have resulted in overfitting.

False positives (FP) beats may still have many cadential features. An inspection of the 28 PAC reported as FP in the Haydn corpus shows that at least 5 FP can be seen as actual cadences, for example measure 71 in Haydn op. 55/3, i, shown on Figure 4. Other notable sources of FP are tonic chords following actual HC cadences activating significant features for PAC. The same Figure 4 shows an example of FN, where a silence in the bass makes the computation of many features fail.

Adding rIAC (Bach corpus) lowers the results, but there may be too few such cadences to efficiently build the model. The detection of HC is difficult (Haydn corpus), as there is not a single feature applicable to every case. Half of them are detected, with about 2% FP.

		beats	ref	TP	FP	FN	F_1
haydn-quartets (21 quatuors)	PAC	3583	51	42	28	9	0.69
	HC	3583	32	18	73	14	0.29
bach-wtc-i (12 fugues)	PAC	2357	36	26	3	10	0.80
	PAC+rIAC	2357	46	30	12	16	0.68

Table 3. Detection of cadences on the test sets of both corpora using all features: Number of beats annotated in the reference annotation (ref), true positives (TP), false positives (FP), false negatives (FN), and F_1 measure (harmonic mean of the recall and the precision).

	haydn-quartets		bach-wtc-i	
	PAC	HC	PAC	PAC+rIAC
All features XYZR	0.69	0.29	0.80	0.68
Features YZR	0.69	0.27	0.71	0.68
Features ZR	0.59	0.24	0.52	0.34
Features XYZ	0.72	0.25	0.74	0.54

Table 4. F_1 measure while detecting cadences on the test sets of both corpora with different sets of features.

Table 4 further studies these results while varying the set of considered features. Some features in *Z* already consider the past. Nevertheless, the features around *Y* are essential to improve the overall detection. Features on *X* bring a small but significant gain for PAC. Rhythmic features (*R*) bring an improvement especially for HC, in particular with *R-Z-strong-beat* that correctly filters out more than half of the beats.

5. CONCLUSION

Different musical clues give the cadential impression of a “breath in music”. We evaluated cadential features on and around three onsets at the arrival and in the preparation of cadences. Without performing any chord segmentation, these features describe the underlying harmony, the voice leading as well as structural aspects of the music.

These features reflect common knowledge of music: we have shown that some of them are specific to cadential points. They make it possible to learn how to predict cadences – PAC/rIAC, and, to a lesser extent, HC – in corpora with reference annotations. Such features may also be used in other systematic musicology approaches.

Perspectives include the extension of our set of features to cadential and non-cadential positions. Some features could be not necessarily theory driven and could possibly have metric values. Coupled with automatic selection, this could lead to the discovery of significant but unexpected features. More generally, the method used to identify points *X* and *Y* could be compared to other heuristics. Cadence preparations could for example be described by features regarding contiguous “spans” of onsets rather than single onsets *X* and *Y*, in order to improve the harmony relevance of the model. Research along these lines could significantly improve HC detection.

Acknowledgements. The authors are grateful to Robin Alais, Mikaela Keller, Marie Perrier, and the Algomus team for fruitful discussions and proofreading of the manuscript. We also thank the anonymous reviewers for their insightful comments. This work is partially funded by French CPER MAuVE (ERDF, Région Hauts-de-France) and by a grant from the French Research Agency (ANR-11-EQPX-0023 IRDIVE).

6. REFERENCES

- [1] Louis Bigo, Jean-Louis Giavitto, Moreno Andreatta, Olivier Michel, and Antoine Spicher. Computation and visualization of musical structures in chord-based simplicial complexes. In *Mathematics and Computation in Music (MCM 2013)*, pages 38–51, 2013.
- [2] Louis Bigo, Mathieu Giraud, Richard Groult, Nicolas Guimard-Kagan, and Florence Levé. Sketching sonata form structure in selected classical string quartets. In *International Society for Music Information Retrieval Conference (ISMIR 2017)*, 2017.
- [3] Ann Blombach. Phrase and cadence: A study of terminology and definition. *Journal of Music Theory Pedagogy*, 1:225–51, 1987.
- [4] William E. Caplin. The classical cadence: Conceptions and misconceptions. *Journal of the American Musicological Society*, 57:51–117, 2004.
- [5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [6] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 637–642, 2010.
- [7] W. Bas De Haas, Martin Rohrmeier, Remco C. Veltkamp, and Frans Wiering. Modeling harmonic similarity using a generative grammar of tonal harmony. In *International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 549–554, 2009.
- [8] Fabian Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [9] Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. Computational fugue analysis. *Computer Music Journal*, 39(2), 2015.
- [10] James Hepokoski and Warren Darcy. *Elements of Sonata Theory: Norms, Types, and Deformations in the Late-Eighteenth-Century Sonata*. Oxford University Press, 2006.
- [11] David Huron. Music information processing using the Humdrum toolkit: Concepts, examples, and lessons. *Computer Music Journal*, 26(2):11–26, 2002.
- [12] Plácido R. Illescas, David Rizo, and José M. Iñesta. Harmonic, melodic, and functional automatic analysis. In *International Computer Music Conference (ICMC 2007)*, pages 165–168, 2007.
- [13] Carol L. Krumhansl and Edward J. Kessler. Tracing the dynamic changes in perceived tonal organisation in a spatial representation of musical keys. *Psychological Review*, 89(2):334–368, 1982.
- [14] Nathan John Martin and Julie Pedneault-Deslauriers. *The Mozartean Half Cadence*, pages 185–213. In Neuwirth and Bergé [15], 2015.
- [15] Markus Neuwirth and Pieter Bergé, editors. *What Is a Cadence? Theoretical and Analytical Perspectives on Cadences in the Classical Repertoire*. Leuven University Press, 2015.
- [16] Jean-François Paiement, Douglas Eck, and Samy Bengio. A probabilistic model for chord progressions. In *International Conference on Music Information Retrieval (ISMIR 2005)*, 2005.
- [17] Marcus Thomas Pearce. *The construction and evaluation of statistical models of melodic structure in music perception and composition*. PhD thesis, City University London, 2005.
- [18] Martin Rohrmeier and Markus Neuwirth. *Towards a Syntax of the Classical Cadence*, pages 287–338. In Neuwirth and Bergé [15], 2015.
- [19] Janet Schmalfeldt. Cadential processes: The evaded cadence and the “one more time” technique. *Journal of Musicological Research*, 12(1-2):1–52, 1992.
- [20] David Sears, William E. Caplin, and Stephen McAdams. Perceiving the classical cadence. *Music Perception*, 31(5):397–417, 2014.
- [21] David R. W. Sears, Andreas Arzt, Harald Frostel, Reinhard Sonnleitner, and Gerhard Widmer. Modeling harmony with skip-grams. In *International Society for Music Information Retrieval Conference (ISMIR 2017)*, pages 332–338, 2017.
- [22] David R. W. Sears, Marcus T. Pearce, William E. Caplin, and Stephen McAdams. Simulating melodic and harmonic expectations for tonal cadences using probabilistic models. *Journal of New Music Research*, 47(1):29–52, 2018.
- [23] David Temperley. What’s key for key ? the Krumhansl-Schmuckler key-finding algorithm reconsidered. *Music Perception*, 17(1):65–100, 1999.
- [24] David Temperley. *Music and probability*. The MIT Press, 2007.
- [25] Peter van Kranenburg and Folger Karsdorp. Cadence detection in western traditional stanzaic songs using melodic and textual features. In *International Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 391–396, 2014.

On the Relationships between Music-induced Emotion and Physiological Signals

Xiao Hu

The University of Hong Kong Shenzhen
Institute of Research and Innovation
xiaoxhu@hku.hk

Fanjie Li

Sichuan University
fjlichn@gmail.
com

Jeremy T. D. Ng

The University of Hong Kong
jeremyng@hku.hk

ABSTRACT

Emotion-aware music information retrieval (MIR) has been difficult due to the subjectivity and temporality of emotion responses to music. Physiological signals are regarded as related to emotion and thus could potentially be exploited in emotion-aware music discovery. This study explored the possibility of using physiological signals to detect users' emotion responses to music, with consideration of individual characteristics (personality, music preferences, etc.). A user experiment was conducted with 23 participants who searched for music in a novel MIR system. Users' listening behaviors and self-reported emotion responses to a total of 628 music pieces were collected. During music listening, a series of peripheral physiological signals (e.g., heart rate, skin conductance) were recorded from participants unobtrusively using a research-grade wearable wristband. A set of features in the time- and frequency- domains were extracted from the physiological signals and analyzed using statistical and machine learning methods. Results reveal 1) significant differences in some physiological features between positive and negative arousal and mood categories, and 2) effective classification of emotion responses based on physiological signals for some individuals. The findings can contribute to further improvement of emotion-aware intelligent MIR systems exploiting physiological signals as an objective and personalized input.

1. INTRODUCTION

Mood-based music discovery is a typical scenario of music information retrieval (MIR). Previous research has adopted content-based [11][27], collaborative filtering [7], or semantic-based [4] approach to recognize the emotion the music expressed and therefore enable emotion-aware MIR. Beyond research, Web-based music services are also available to support searching for music based on emotions, such as MoodFuse, Musicoverly, etc.

However, emotion responses to music are subjective, varying from one user to another. They are also temporal in that the same user may respond to the same music differently at different times [33]. These make it challenging

to optimize emotion-aware music retrieval.

Physiological signals such as heart rate, blood pressure and skin conductance were found to be related to people's emotion status [1][12][20] and they are deemed objective compared to self-reported emotion status which has been criticized as being subjective and sometimes inaccurate [2][3]. Therefore, physiological signals could potentially be exploited in emotion-aware music discovery. In addition, with the rapid development of wearable technology in recent years, peripheral physiological signals can be collected from users through small and less noticeable devices (e.g., wristband) in naturalistic settings in unobtrusive manners [13]. This advantage is not yet comparable by methods of gathering other physiological signals such as eye tracking and electroencephalography, and thus peripheral physiological signals are currently preferable for studying users' emotion responses to music in everyday life.

This study, therefore, aims to explore to what extent physiological signals measured by a wearable device are related to users' emotion responses to music played on an MIR system. Specifically, we are interested in the following research questions:

RQ1: Among features extracted from physiological signals collected during music listening, which ones differ significantly across different emotion responses?

RQ2: To what extent can physiological signals collected during music listening be used to predict users' emotion response to music?

As emotion responses to music may vary across listeners [37], we take into consideration listeners' characteristics by asking the following question:

RQ3: To what extent do prediction performances vary across different users and user characteristics (i.e. personality, music preferences)?

To answer these questions, a user experiment was conducted to collect data of users' interactions with a novel MIR system [17]. During the experiment, participants were asked to explore the music collection in the system while users' music listening behaviors and self-reported emotion responses to the music were recorded by the system. Simultaneously, physiological signals of the users were collected using a research-grade wearable wristband. Statistical tests and machine learning classifiers were applied to analyze the data, with classification performances compared across different classification



© Hu, X. Li, F. Ng, J. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Hu, X. Li, F. Ng, J. "On the Relationships between Music-induced Emotion and physiological signals", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

algorithms and users. Furthermore, collected in the pre-experiment questionnaire, participants' personality and music preferences were analyzed to see if they played a role in the relationships between physiological signals and emotion responses to music. As one of the first studies exploiting peripheral physiological signals in MIR, findings of this study can shed light on the feasibility of predicting users' emotion responses to music based on physiological signals and contribute to future implementation of emotion-aware MIR systems.

2. RELATED WORK

Work related to this study can be broadly categorized into physiological signal analysis in information retrieval and emotion-based music discovery.

2.1 Physiological Signals in Information Retrieval

Although using physiological signals as an implicit measurement of users' affective states during information retrieval process is still an emerging research topic, several recent studies have demonstrated its usefulness in predicting users' relevance judgments [2][3][28] and engagement levels [12]. Moshfeghi and Jose [28] used physiological features derived from heart rate, galvanic skin response, and skin temperature, along with facial expression features and behavioral features (i.e. dwell time), to predict users' relevance judgments in video retrieval tasks. They found that the combination of dwell time and heart rate features performed better for the task with entertainment-based search intention (i.e., when the main purpose of video search was to adjust arousal level or mood). Edwards and Kelly [12] combined skin conductance, heart rate with search behavior measures to evaluate users' levels of engagement, frustration, and stress when conducting searching tasks on a Web search interface. The results suggested that heart rate might be more associated with negative arousal, and skin conductance with positive arousal.

These studies in text and video information retrieval were encouraging and inspiring, yet there is little research on MIR exploiting physiological measures. Like many videos, music is a strong stimulus in eliciting emotion from listeners [23], and many users indeed listen to music for the very purpose of emotion modulation [16][34]. Considering the close relationship between music and emotion, as well as that between emotion and physiological signals, this study aims to help bridge the gap of incorporating physiological signals in MIR.

2.2 Emotion-aware MIR

The majority of previous research on music emotion recognition adopted content-based [11][27], collaborative filtering [7], and/or semantic-based [4] approaches which may suffer various shortcomings such as ignoring individual differences and the "cold start" problem (i.e., the recommendation performance is poor when few user ratings are available) [25]. Physiological signals, on the other

hand, provide a new approach to understand users' emotion response to music. Several prior studies have probed physiology-based approach in MIR and yielded promising initial results. The Affective DJ project [9] used changes in users' skin conductance to detect users' mood based on which it helped users select music and generate playlists. Their evaluation results confirmed that skin conductance has a significant correlation with perceived excitement level of a song. Oliver et. al [30] also proposed a framework of automatic playlist generation by monitoring users' purpose of music listening and physiological responses (i.e. heart rate, galvanic skin response, respiration rate, and movement) to music. As an exemplar case of the framework, the MPTrain system was designed and implemented for selecting songs for runners to accompany their exercises. More recently, an affective music player (AMP) was developed to select music for mood enhancement by modeling the effects of music based on changes in skin conductance level and skin temperature [22]. Validation of the AMP found that lower skin temperatures were related to more positive emotions induced by music listening.

Notwithstanding the impact of these existing studies, the investigation on the relationship between physiological signals and emotion responses to music is still limited. Moreover, to the best of our knowledge, few studies have probed whether and how individual differences (in personality, music preferences, etc.) may play a role in such relationships. This study aims to bridge these gaps.

3. USER EXPERIMENT OF INTERACTIVE MUSIC SEARCH

The purpose of this experiment was to collect physiological signals and self-reported emotion response to music during music searching and listening. To encourage participants to interact with music, during the experiment, participants were asked to create a playlist using a novel Web-based music retrieval system. Participants' physiological signals during music listening were collected, as well as their self-reported emotion responses to each piece of music they listened to.

3.1 The MIR System

The Moody system [18] (now in its 3rd version) is a novel music retrieval system which supports searching for songs using several criteria: Genre (e.g., folk, jazz), Occasion (e.g., party, workout), Artist, Song, Album, and presents basic metadata and album image of each retrieved song (shown in Figure 1). Users can listen to any songs they are interested in using an HTML5 music player embedded in the Web interface of the system. They can also select any songs to add into a playlist at any time. Users' interactions with the systems (e.g., search, play) are recorded in the system logs.

The music collection hosted in the system is a subset of the Jamendo dataset, one of the world's largest digital services for free music. The subset of 10K tracks was ob-

tained through the Grand Challenge in User Experience of the Music Information Retrieval Evaluation Exchange (MIREX)[17]. All the tracks are under the CC-BY license and thus the full tracks (of 60+ Gigabytes in total) can be freely listened to by the public. Metadata of the tracks (e.g., title, album, artist) as well as free tags were also obtained from Jamendo and displayed in the system to facilitate searching and browsing.

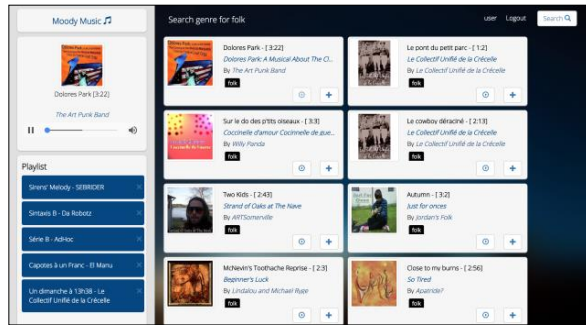


Figure 1. Interface of the Moody System (version 3)

3.2 Experiment Procedure

The experiment consisted of four main phases: 1) pre-experiment questionnaire; 2) instructions of the Moody system and the search task; 3) participants searching and listening to music; and 4) post-experiment questionnaire.

The pre-experiment questionnaire gathered information on demographics, music listening behaviors, music preference, and personality as measured by the Ten Item Personality scale (TIPI) [14] which contains 10 questions in five dimensions: extrovert - introvert; agreeable - disagreeable; open-close; stable - unstable; conscientious - unconscientious.

During phase 3, participants were asked to use the Moody system for no less than 40 minutes, looking for at least 10 songs they liked to make a playlist. They were encouraged to search for different types of music for a more diverse experience. For each music piece listened to for more than 30 seconds, participants would be prompted to answer two questions on their current emotion. The first question asked participants to give a score of arousal [32] on a scale of -10 (low arousal) to 10 (highly aroused), while the second question was to choose a mood category from a set of options adapted from [37] (Figure 2).

Also during phase 3, participants were asked to wear an Empatica E4 wristband [13] which is one of the few wearable devices designed specifically for measuring physiological signals for research purposes. This device supports real-time data acquisition and provides a secure API for raw data downloading. It has been used in emotion-related studies in psychology, health sciences, etc. with high reliability (e.g., [29]). For signal stabilization, the wristband was mounted on a participant's wrist 2 minutes before the search task started.

After conducting the task, the last phase of the experiment was for the participants to fill out a post-experiment questionnaire concerning their emotional states and general experience of the experiment including the search process.

The experiment took place in a computer classroom where participants worked on iMac computers. Earphones were used during the music searching and listening. Ethical consent forms were signed at the beginning and a nominal remuneration was paid at the end to compensate participants' time.



Figure 2. Question on mood popped up in the Moody system. (Translation of the instruction: “Please choose one of the following moods that can best represent your current mood. (Note: This refers to your mood, not the mood expressed by the music piece).”)

3.3 Data Collection

23 participants (15 males, 8 females) were recruited to join this experiment. All participants were undergraduate or graduate students in a comprehensive university in Hong Kong, whose majors ranged from Social Sciences, Science, Engineering, Business, Humanities & Arts to Medicine, with a diverse background in music knowledge and a relatively high frequency of music listening ranging from several times a week to a daily basis.

Physiological signals collected included electrodermal activity (EDA), blood volume pulse (BVP), inter beat interval (IBI), heart rate (HR) and skin temperature (TEMP). The sampling rates of EDA, BVP, HR, and TEMP are 4 Hz, 64Hz, 1Hz, and 4Hz respectively.

Among all participants, we collected arousal and mood ratings for 628 pieces of music. Each piece of music was listened to for approximately 80 seconds on average.

4. DATA ANALYSIS

4.1 Data Preprocessing

Before extracting features, we constructed two datasets: one with raw physiological signals, and the other with normalized physiological signals by z-score normaliza-

tion [31]. As physiological signals vary across individuals [26], the normalization was conducted within each individual participant.

Time-series of physiological data in both datasets were then aligned with the starting and ending time of each music piece played in the experiment as recorded by the Moody system logs. Physiological data were then split into chunks corresponding to each song played by each participant in the experiment.

The emotion status reported by participants after listening to each piece was aligned with the physiological signals and taken as the ground truth labels in the classification analysis. The arousal and mood values rated by participants were then grouped into three main categories (i.e. positive, negative, neutral) for comparison using ANOVA and t-tests as well as classification. This resulted in 436 positive, 175 negative, and 17 neutral (i.e., 0) arousal ratings. For mood ratings, we combined the mood categories into positive (happy, blessed, etc.), negative (sad, fearful, etc.), and neutral moods (i.e. none), resulting in 387, 141, and 100 ratings respectively.

4.2 Feature Extraction

After data preprocessing, features of physiological signals during each music listening period were extracted based on time series and spectrum analysis. Table 1 summarizes the features extracted in this study.

Category	Features
Descriptive statistics of raw signals	Mean, Standard deviation, Median, Range [8], [20]
Time series features	Means of the absolute values of the 1st / 2nd differences of the raw / normalized signals [31]
Frequency domain features	HF, LF, LF/HF [36]
Physiological signal specific features	skin conductance response (SCR)[6], heart rate variability (HRV) [1]

Table 1. Features extracted from physiological signals.

Features considered in this study were those found closely linked to emotions by previous studies [1][6][8][20][31][36], including descriptive statistics such as median, range, standard deviation (stdev), means of the first difference in raw values (MFDR) and in normalized values (MFDN), means of the second difference in raw values (MSDR) and in normalized values (MSDN), low and high frequency (LF, HF) in frequency spectrum which was obtained through a Fast Fourier Transformation (FFT) on the time domain signals, as well as the ratio of the two (LF/HF). In addition, two features specific to physiological signals were considered. The first is skin conductance response (SCR) which depicts the phenomenon of the skin momentarily being a better conductor of electricity. The SCR is characterized by an increase in electrodermal response followed by a decrease in re-

sponse [6]. It is generally related to stimulus arousal [24]. The second is heart rate variability (HRV) which measures the continuous interplay between sympathetic and parasympathetic influences on the heart rate. HRV has been found relevant to arousal as well [1].

4.3 Feature Analysis

Using a one-way ANOVA, we compared physiological features across the three arousal categories as well as the three mood categories (i.e., positive, negative, neutral). We also applied t-tests on the features between positive and negative categories of arousal and mood (i.e., without consideration of the neutral categories). As multiple comparisons were involved, Bonferroni correction [15] and Benjamini–Hochberg procedure [5] were applied to ANOVA and t-tests respectively to control Type I error. Features with significant differences across arousal and mood categories are identified.

4.4 Classification and Evaluation

A machine learning approach was applied to measure the extent to which physiological signals could be used to recognize users’ emotion responses to music listening, in positive and negative categories of arousal and mood. Specifically, we trained and compared the performance of several well-adopted classification models representative of different approaches, namely decision tree, k-Nearest Neighbor (k-NN), naïve Bayes and SVM.

As the sample distribution across the positive and negative categories is unbalanced, for each classifier and each category pair (i.e., arousal, mood), we constructed a balanced dataset by randomly selecting samples from the larger categories and performed a classification experiment on the balanced dataset. This process was repeated 10 times and within each time a 10-fold cross-validation was applied to evaluate the performances.

Besides classification based on the whole feature sets, the forward feature selection method was applied in combination with the classifiers to remove redundant and noisy features and improve classification performances.

In addition, to examine whether prediction performances vary across different user characteristics, we also conducted a classification experiment on data partitioned by participants, their personality, as well as their music preferences.

5. RESULTS AND DISCUSSION

5.1 Features with Significant Differences across Emotion Categories

Features found with significant differences in the ANOVA and t-test results after Bonferroni correction are shown in Table 2.

For arousal, both tests indicated that BVP, HR and EDA features differed significantly across categories. For mood, HR_range showed consistent significance across the two tests. In addition, HR and EDA seem more prom-

ising than other physiological signals in predicting and/or monitoring listeners' emotion responses to music. However, TEMP features, SCR, and time domain measures of HRV were not significant in either test.

Feature	Arousal		Mood	
	ANOVA	t-test	ANOVA	t-test
BVP_median	0.034	0.012	0.004	-
BVP_HF	0.049	0.007	-	-
HR_stdev	0.022	0.002	-	0.005
HR_range	0.010	< 0.001	0.039	0.003
HR_LF	0.022	< 0.001	-	0.004
HR_HF	0.028	0.001	-	0.006
EDA_MFDN	0.020	0.003	-	-
EDA_MSDN	0.017	0.003	-	0.008
EDA_LF/HF	0.036	-	-	-
IBI_median	-	-	0.006	-
IBI_mean	-	-	0.006	-

Table 2. Significant results (p values) of ANOVA and t-tests across extracted features.

5.2 Classification on the Dataset of All Listeners

Table 3 shows the performances of different classifiers with feature selection, on datasets balanced by repeated random sampling. Both accuracy and Cohen's kappa are used as performance measures. In general, the classification performances on the dataset aggregated across all users were low, with the best performances (k-NN) being around 60% in accuracy (baseline 50%) and lower than 0.2 in Cohen's kappa (indicating low agreement [35]).

Classifier	Arousal		Mood	
	Accuracy	Kappa	Accuracy	Kappa
Decision Tree	55.43%	0.109	60.04%	0.201
k-NN	59.97%	0.199	60.78%	0.216
Naïve Bayes	57.74%	0.155	58.83%	0.177
SVM	58.40%	0.168	59.50%	0.190

Table 3. Classification results on balanced datasets consisting of all listeners.

5.3 Classification on Individual Listeners

To examine whether and how prediction results differ across participants, the k-NN classifier was applied to datasets of individual participants. As the sample sizes of some participants were not sufficient for constructing balanced datasets of non-trivial sizes, these sets of experiments on individual participants were performed on unbalanced datasets. Therefore, F1 measure and Cohen's kappa were used and reported in Table 4. From the Cohen's kappa values in the results, we can see that the performances on individual listeners were much better than those on the dataset of all participants (Table 3). This difference implies that individual variability on physiological signal analysis might be too large to build generic classifiers that work for most (if not all) listeners.

The results also indicate that, for some participants, e.g. Users 5, 8, and 18, the prediction worked well for both arousal and mood, whereas for other participants, such as users 11, neither prediction exhibited good results (Cohen's kappa values were lower than 0.2). The variability of prediction performances across participants corroborates with findings in existing research that physiological signals are highly individual dependent [19][26].

User	No. of songs	Arousal		Mood	
		F measure	Kappa	F measure	Kappa
User 1	24	90.00%	0.400	78.57%	0.294
User 2	24	81.48%	0.577	62.50%	0.262
User 3	28	78.57%	0.571	90.00%	0.800
User 4	45	80.00%	0.537	93.33%	0.700
User 5	43	98.70%	0.788	93.55%	0.604
User 6	30	81.25%	0.490	84.85%	0.516
User 7	29	86.36%	0.435	93.03%	0.516
User 8	17	96.54%	0.767	96.55%	0.767
User 9	24	94.44%	0.694	96.30%	0.765
User 10	33	87.80%	0.670	92.31%	0.747
User 11	21	93.75%	-0.063	97.14%	0.000
User 12	18	88.89%	0.722	93.33%	0.843
User 13	25	94.74%	0.614	90.48%	0.405
User 14	29	91.30%	0.580	95.00%	0.750
User 15	31	80.00%	0.427	83.72%	0.440
User 16	30	92.59%	0.259	92.31%	0.423
User 17	32	88.89%	0.595	85.00%	0.475
User 18	33	98.36%	0.784	98.31%	0.784
User 19	33	84.45%	0.507	90.32%	0.750
User 20	35	84.00%	0.380	91.67%	0.586
User 21	16	91.67%	0.673	96.00%	0.818

Table 4. Classification performance on individual participants.

5.4 Classification on Participant Groups

Table 5 shows classification performance of users with different personalities. Personality was determined based on responses to the TIPI questionnaire [14] which consisted of five personality dimensions. For each dimension, each user was categorized to either end based on their answers to the two question items in that dimension. This set of experiments were also conducted on balanced datasets, with accuracy and Cohen's kappa values reported (Table 5). Compared to performances on the dataset of all listeners (Table 3), classification performances on some of the personality dimensions were better. In particular, predictions on users with Agreeable personality reached Cohen's kappa values of 0.581 (for arousal) and 0.682 (for mood) which are deemed as medium and high agreement levels respectively [35].

Another observation is that the personality dimensions with relatively high classification performances had lower numbers of users compared to other personality dimensions. A correlation analysis revealed significant negative correlations between the number of users and classification performances ($r = -0.78$ for both measures of arousal

al, $p = 0.008$; and $r = -0.62$ for both measures of mood prediction, $p = 0.054$). This again implies the significance of individual differences in physiological signal analysis. A suggestion for future research is thus to analyze physiological signals within individual users.

Personality	No. of users	Arousal		Mood	
		Accuracy	Kappa	Accuracy	Kappa
Extrovert	12	62.42%	0.248	65.00%	0.300
Introvert	11	67.56%	0.351	63.65%	0.273
Agreeable	3	79.03%	0.581	84.09%	0.682
Disagreeable	20	62.50%	0.250	61.64%	0.233
Open	7	64.51%	0.290	66.58%	0.332
Close	16	62.96%	0.259	74.69%	0.494
Stable	9	64.07%	0.281	69.38%	0.388
Unstable	14	62.39%	0.248	62.41%	0.248
Conscientious	6	69.75%	0.395	68.91%	0.378
Unconscientious	17	60.65%	0.213	61.05%	0.221

Table 5. Classification performances of each personality dimension.

Besides personality, users' music preference might play a role in their emotion responses to music [21]. Therefore, we grouped the participants based on their self-reported genre preferences using the k-means clustering algorithm, with the optimal k value selected by the Davies Bouldin index [10]. The results yielded three clusters corresponding to preferences as shown in Table 6. We then conducted classification experiments on participants in each of the clusters, using the k-NN algorithm and balanced datasets. Prediction performances (Table 6) were higher than those on the whole dataset (Table 3), and the performances on the mood classification of cluster 2 were comparable to those of some individual users (Table 4). These results indicate that certain music preferences might play a role in predicting emotion responses to music based on physiological signals. Future work is needed to further investigate this phenomenon, preferably with larger samples.

Cluster	Preferences	No. of users	Arousal		Mood	
			Accuracy	κ	Accuracy	κ
0	Pop only	7	64.26%	0.285	71.32%	0.426
1	Classical, Folk, Pop	10	65.17%	0.303	64.06%	0.281
2	Electronica, Rock, Pop	6	69.02%	0.380	76.76%	0.535

Table 6. Classification performances of participant clusters based on music preferences.

6. CONCLUSION AND FUTURE WORK

This paper presented a study towards recognizing users' emotion response to music using physiological data obtained from wearable sensors. ANOVA and t-tests revealed that heart rate (HR) and electrodermal activity

(EDA) features were consistently significant in both arousal and mood dimensions. This finding provides empirical evidence for feature extraction and selection in future studies. In predicting emotion responses to music based on physiological signals during music listening, predictions on individual participants showed promising performances as well as large performance differences across individuals. These results verified that the predictability of emotion responses to music based on physiological signals may vary from person to person. Additionally, the classification experiments conducted on data partitioned by personality dimensions and music preferences illustrated that classification based on physiological signals might be more effective for users with certain personality traits or genre preferences, such as being agreeable or preferring Electronica and Rock music. However, these results were confounded with the sample size, as the number of users in each personality category was negatively correlated with performance measures, further indicating variability across users and suggesting that individual-based analysis might be more fruitful for exploiting physiological signals. The results reported in this paper demonstrate the potential of physiological sensing techniques in emotion-aware MIR. This opens up a number of possibilities in future MIR systems and services, such as recommending music based on users' current physiological measures and maintaining mood-based playlists which can be adjusted in real time based on changes in physiological signals, etc.

Future studies will be conducted to further investigate in which circumstances physiological signals are more effective in predicting emotion responses to music. Combinations of factors will be considered such as the matching between music preferences and the music pieces being listened to. In the next stage of our research, we will also explore the effect of incorporating music features (e.g., acoustic, emotion, occasion, etc.) in the prediction as well as users' behavioral logs recorded in the user experiment. Besides, the experiment in this study was run in laboratory settings. To achieve a higher level of ecological validity, future experiments can be extended to the everyday environment of the participants and for longer time spans.

7. ACKNOWLEDGEMENT

The authors acknowledge the support given by the National Natural Science Foundation of China: 61703357, and the Research Grants Council of the HKSAR Government, #T44-707/16/N, under the Theme Based Research Scheme. We also thank the anonymous reviewers for their helpful suggestions.

8. REFERENCES

- [1] B. M. Appelhans and L. J. Luecken, "Heart rate variability as an index of regulated emotional

- responding,” *Review of General Psychology*, Vol. 10, No. 3, pp. 229–240, 2006.
- [2] I. Arapakis, I. Konstas, and J. M. Jose: “Using facial expressions and peripheral physiological signals as implicit indicators of topical relevance,” *Proc. of the 17th ACM international conference on Multimedia*, pp. 461–470, 2009
- [3] O. Barral, M. J. A. Eugster, T. Ruotsalo, M. M. Spapé, I. Kosunen, N. Ravaja, S. Kaski, and G. Jacucci: “Exploring peripheral physiology as a predictor of perceived relevance in information retrieval,” *Proc. of the 20th International Conference on Intelligent User Interfaces*, pp. 389–399, 2015.
- [4] M. Barthet, G. Fazekas, A. Allik, and M. Sandler: “Moodplay: an interactive mood-based musical experience,” *Proc. of the Audio Mostly 2015 on Interaction with Sound*, p. 3, 2015.
- [5] Y. Benjamini, Y. Hochberg: “Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing,” *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 57, No. 1, pp. 289–300, 1995.
- [6] W. Boucsein: “Electrodermal Activity,” *Plenum Series in Behavioral Psychophysiology and Medicine*, Plenum Press, 1992.
- [7] J. Broekens, A. Pronker, and M. Neuteboom: “Real time labeling of affect in music using the affect button”. *Proc. of the 3rd international workshop on Affective interaction in natural environments*, pp. 21–26, 2010.
- [8] G. Chanel, C. Rebetez, M. Bétrancourt, and T. Pun: “Boredom, engagement and anxiety as indicators for adaptation to difficulty in games,” *Proc. of the 12th international conference on Entertainment and media in the ubiquitous era*, pp. 13–17, 2008
- [9] F. Dabek, J. Healey, and R. Picard: “A new affect-perceiving interface and its application to personalized music selection,” *Proc. from the 1998 Workshop on Perceptual User Interfaces*, 1998
- [10] D. L. Davies and D. W. Bouldin: “A Cluster Separation Measure,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-1, No. 2, pp. 224–227, Apr. 1979.
- [11] J. J. Deng, C. H. C. Leung, A. Milani, and L. Chen: “Emotional states associated with music: Classification, prediction of changes, and consideration in recommendation,” *ACM Transactions on Interactive Intelligent Systems*, Vol. 5, No. 1, pp. 1–36, Mar. 2015
- [12] A. Edwards and D. Kelly: “Engaged or Frustrated: Disambiguating Emotional State in Search,” *Proc. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 125–134, 2017.
- [13] M. Garbarino, M. Lai, D. Bender, R. W. Picard, and S. Tognetti: “Empatica E3—A wearable wireless multi-sensor device for real-time computerized biofeedback and data acquisition,” *Proc. of the 4th International Conference on Wireless Mobile Communication and Healthcare*, pp. 39–42, 2014
- [14] S. D. Gosling, P. J. Rentfrow, and W. B. Swann: “A very brief measure of the Big-Five personality domains,” *Journal of Research in Personality*, Vol. 37, No. 6, pp. 504–528, Dec. 2003.
- [15] S. Holm: “A simple sequentially rejective multiple test procedure,” *Scandinavian Journal of Statistics* pp. 65–70, 1979.
- [16] X. Hu, N. Kando, “Evaluation of Music Search in Casual-Leisure Situations,” *Proc. of Search for Fun Workshop at the Information Interaction in Context conference (IliX)*, 2014.
- [17] X. Hu, J. Lee, D. Bainbridge, K. Choi, P. Organisciak and J. Downie, “The MIREX grand challenge: A framework of holistic user-experience evaluation in music information retrieval”, *Journal of the Association for Information Science and Technology*, Vol. 68, no. 1, pp. 97–112, 2017.
- [18] X. Hu, V. Sanghvi, B. Vong, P. J. On, C. Leong, and J. Angelica. “Moody: A web-based music mood classification and recommendation system”, *Proc. of 9th International Conference on Music Information Retrieval*, Philadelphia, U.S. 2008.
- [19] M. S. Hussain, O. AlZoubi, R. A. Calvo, and S. K. D’Mello: “Affect detection from multichannel physiology during learning sessions with AutoTutor,” *International Conference on Artificial Intelligence in Education*, pp. 131–138, 2011.
- [20] M. S. Hussain, R. A. Calvo, and F. Chen: “Automatic cognitive load detection from face, physiology, task performance and fusion during affective interference,” *Interacting with Computers*, Vol. 26, No. 3, pp. 256–268, Jun. 2013.
- [21] M. Iwanaga and Y. Moroki: “Subjective and Physiological Responses to Music Stimuli Controlled Over Activity and Preference,” *Journal of Music Therapy*, Vol. 36, No. 1, pp. 26–38, Mar. 1999.
- [22] J. H. Janssen, E. L. van den Broek, and J. H. D. M. Westerink, “Tune in to your emotions: a robust personalized affective music player,” *User Modeling and User-Adapted Interaction*, Vol. 22, No. 3, pp. 255–279, Oct. 2011.

- [23] P. Kenealy: "Validation of a music mood induction procedure: Some preliminary findings," *Cognition & Emotion*, Vol. 2, No. 1, pp. 41–48, Mar. 1988.
- [24] S. Khalfa, P. Isabelle, B. Jean-Pierre, and R. Manon: "Event-related skin conductance responses to musical emotions in humans," *Neuroscience Letters*, Vol. 328, No. 2, pp. 145–149, Aug. 2002.
- [25] S. Khusro, Z. Ali, and I. Ullah: "Recommender Systems: Issues, Challenges, and Research Opportunities," *Information Science and Applications (ICISA) 2016*, pp. 1179–1189, 2016.
- [26] A. M. Kiviniemi, H. Hintsala, A. J. Hautala, T. M. Ikaheimo, J. J. Jaakkola, S. Tiinonen, T. Seppanen, and M. P. Tulppo: "Impact and management of physiological calibration in spectral analysis of blood pressure variability," *Frontiers in Physiology*, Vol. 5, Dec. 2014.
- [27] F. F. Kuo, M. F. Chiang, M. K. Shan, and S. Y. Lee: "Emotion-based music recommendation by association discovery from film music," *Proc. of the 13th annual ACM international conference on Multimedia*, pp. 507-510, 2005.
- [28] Y. Moshfeghi and J. M. Jose: "An effective implicit relevance feedback technique using affective, physiological and behavioral features," *Proc. of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pp. 133-142, 2013.
- [29] S. C. Müller and T. Fritz: "Stuck and frustrated or in flow and happy: Sensing developers' emotions and progress," *Proc. of the 37th International Conference on Software Engineering IEEE*, Vol 1, 2015.
- [30] N. Oliver, L. Kregor-Stickles: "PAPA: Physiology and Purpose-Aware Automatic Playlist Generation," *Proc. of 7th International Conference on Music Information Retrieval*, pp. 250-253, Victoria, Canada. 2006
- [31] R. W. Picard, E. Vyzas, and J. Healey: "Toward machine emotional intelligence: analysis of affective physiological state," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 10, pp. 1175–1191, 2001.
- [32] J. A. Russell: "A circumplex model of affect," *Journal of Personality and Social Psychology*, Vol. 39, No. 6, pp. 1161–1178, 1980.
- [33] L. Su, C. C. M. Yeh, J.-Y. Liu, J.-C. Wang, and Y.-H. Yang: "A Systematic Evaluation of the Bag-of-Frames Representation for Music Information Retrieval," *IEEE Transactions on Multimedia*, Vol. 16, No. 5, pp. 1188–1200, Aug. 2014.
- [34] T. F. M. Ter Bogt, J. Mulder, Q. A. W. Raaijmakers, and S. Nic Gabhainn: "Moved by music: A typology of music listeners," *Psychology of Music*, Vol. 39, No. 2, pp. 147–163, Aug. 2010.
- [35] A. J. Viera and J. M. Garrett: "Understanding interobserver agreement: the kappa statistic," *Physical Therapy*, Mar. 2005.
- [36] W. Wu, Y. Gil, and J. Lee: "Combination of Wearable Multi-Biosensor Platform and Resonance Frequency Training for Stress Management of the Unemployed Population," *Sensors*, Vol. 12, No. 12, pp. 13225–13248, Sep. 2012.
- [37] Y. H. Yang and Y. C. Teng: "Quantitative Study of Music Listening Behavior in a Smartphone Context," *ACM Transactions on Interactive Intelligent Systems*, Vol. 5, No. 3, pp. 1–30, Sep. 2015.

MUSIC MOOD DETECTION BASED ON AUDIO AND LYRICS WITH DEEP NEURAL NET

Rémi Delbouys Romain Hennequin Francesco Piccoli
Jimena Royo-Letelier Manuel Moussallam
Deezer, 12 rue d'Athènes, 75009 Paris, France
research@deezer.com

ABSTRACT

We consider the task of multimodal music mood prediction based on the audio signal and the lyrics of a track. We reproduce the implementation of traditional feature engineering based approaches and propose a new model based on deep learning. We compare the performance of both approaches on a database containing 18,000 tracks with associated valence and arousal values and show that our approach outperforms classical models on the arousal detection task, and that both approaches perform equally on the valence prediction task. We also compare the *a posteriori* fusion with fusion of modalities optimized simultaneously with each unimodal model, and observe a significant improvement of valence prediction. We release part of our database for comparison purposes.

1. INTRODUCTION

Music Information Retrieval (MIR) has been an ever growing field of research in recent years, driven by the need to automatically process massive collections of music tracks, an important task to, for example, streaming companies. In particular, automatic music mood detection has been an active field of research in MIR for the past twenty years. It consists of automatically determining the emotion felt when listening to a track.¹ In this work, we focus on the task of multimodal mood detection based on the audio signal and the lyrics of the track. We apply deep learning techniques to the problem and compare our approach to classical feature engineering-based ones on a database of 18,000 songs labeled with a continuous arousal/valence representation. This database is built on the Million Song Dataset (MSD) [2] and the Deezer catalog. To our knowledge this constitutes one of the biggest datasets for multimodal mood detection ever proposed.

¹ We use the words emotion and mood interchangeably, as done in the literature (see [15]).

1.1 Related work

Music mood studies appeared in the first half of the 20th century, with the work of Hevner [7]. In this work, the author defines groups of emotions and studies classical music works to unveil correlations between emotions and characteristics of the music. A first indication that music and lyrics should be jointly considered when analyzing musical mood came from a psychological study exposing independent processing of these modalities by the human brain [3]. For the past 15 years, different approaches have been developed with a wide range of datasets and features. An important fraction of them was put together by Kim et al. in [15]. Li and Ogihara [18] used signal processing features related to timbre, pitch and rhythm. Tzanetakis et al. [28] and Peeters [22] also used classical audio features, such as Mel-Frequency Cepstral Coefficients (MFCCs), as input to a Support Vector Machine (SVM). Lyrics-based mood detection was most often based on feature engineering. For example, Yang and Lee [31] resorted to a psycholinguistic lexicon related to emotion. Argamon et al. [1] extracted stylistic features from text in an author detection task. Multimodal approaches were also studied several times. Laurier et al. [16] compared prediction level and feature level fusion, referred to as late and early fusion respectively. In [26], Su et al. developed a sentence level fusion. An important part of the work based on feature engineering was compiled into more complete studies, among which the one from Hu and Downie [9] is one of the most exhaustive, and compares many of the previously introduced features.

Influenced by advances in deep learning, notably in speech recognition or machine translation, new models began to emerge, based on fewer feature engineering. Regarding audio-based methods, the Music Information Retrieval Evaluation eXchange (MIREX) competition [5] has monitored the evolution of the state of the art. In this framework, Lidy et al. [19] have shown the promise of audio-based deep learning. Recently, Jeon et al. [14] presented the first multimodal deep learning approach using a bimodal convolutional recurrent network with a binary mood representation. However, they neither compared their work to classical approaches, nor evaluated the advantage of their mid-level fusion against simple late fusion of unimodal models. In [12], Huang et al. resorted to deep



Boltzmann machines to unveil early correlations between audio and lyrics, but their method was limited by the incompleteness of their dataset, which made impossible the use of temporally local layers, e.g. recurrent or convolutional ones. To our knowledge, there is no clear answer as to whether feature engineering yields better results than more end-to-end systems for the multimodal task, probably because of the lack of easily accessible large size datasets.

1.2 Mood representation

A variety of mood representations have been used in the literature. They either consist of monolabel tagging with either simple tags (e.g. in [9]), clusters of tags (e.g. in the MIREX competition) or continuous representation. In this work, we resort to the latter option. Russell [24] defined a 2-dimensional continuous space of embedding for emotions. A point in this space represents the valence (from negative to positive mood) and arousal (from calm to energetic mood) of an emotion. This representation was used multiple times in the literature [12, 27, 29], and presents the advantage of being satisfyingly exhaustive. It is worth noting that this representation has been validated by embedding emotions in a 2-dimensional space based on their co-occurrences in a database [10]. Since we choose this representation we formulate mood estimation as a 2-dimensional regression problem based on a track's lyrics and/or audio.

1.3 Contributions of this work

We study end-to-end lyrics-based approaches to music mood detection and compare their performance with classical lyrics-based methods performance, and give insights on the performing architectures and networks types. We show that lyrics-based networks show promising results both in valence and arousal prediction.

We describe our bimodal deep learning model and evaluate the performance of a mid-level fusion, compared to unimodal approaches and to late fusion of unimodal predictions. We show that arousal is highly correlated to the audio source, whereas valence requires both modalities to be predicted significantly better. We also see that the latter task can be notably improved by resorting to mid-level fusion.

Finally, we compare our model to traditional feature engineering methods and show that deep-learning-based approaches outperform classical models, when it comes to multimodal arousal detection, and we show that both systems are equally performing on valence prediction. For future comparison purposes, we also release part of our database consisting of valence/arousal labels and corresponding song identifiers.

2. CLASSICAL FEATURE ENGINEERING-BASED APPROACHES

We compare our model to classical approaches based on feature engineering. These methods were iteratively deepened over the years: for audio-based models, a succes-

sion of works [18, 22, 28] indicated the top performing audio features for mood detection tasks ; for lyrics-based approaches, a series of studies [1, 10, 31] investigated a wide variety of text-based features. Finally, fusion methods were also studied multiple times [9, 16, 29]. Hu and Downie compiled and deepened these works in a series of papers [8–10], which is the most accomplished feature-engineering-based approach of the subject. We reimplement this work and compare its performance to ours. This model consists in the choice of the optimal weighted average of the predictions of two unimodal models: an SVM on top of MFCCs, spectral flux, rolloff and centroid, for audio; and an SVM on top of basic, linguistic and stylistic features (n-grams, lexicon-based features, etc.) for lyrics.

3. DEEP LEARNING-BASED APPROACH

We first explore unimodal deep learning models and then combine them into a multimodal network. In each case, the model simultaneously predicts valence and arousal. Inputs are subdivided in several segments for training, so that each input has the same length. Output is the average of the predictions computed by the model on several segments of the input. For the bimodal models, subdivision of audio and lyrics requires synchronization of the modalities.

3.1 Audio only

We use a mel-spectrogram as input, which are 2-dimensional. We choose a convolutional neural network (ConvNet) [17], the architecture is shown in Fig. 1 (a). It is composed of two consecutive 1-dimensional convolution layers (convolutions along the temporal dimension) with 32 and 16 feature maps of size 8, stride 1, and max pooling of size 4 and stride 4. We resort to batch normalization [13] after each convolutional layer. We use two fully connected layers as output to the network, the intermediate layer being of size 64.

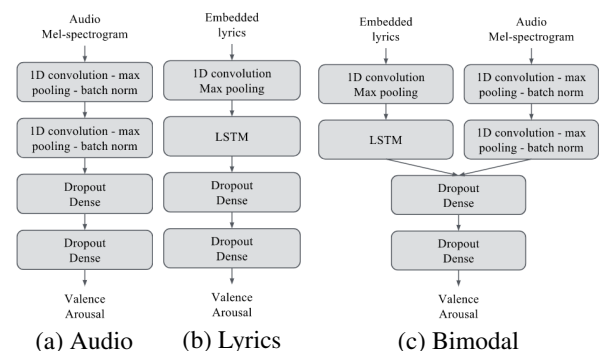


Figure 1. Architecture of unimodal and bimodal models

3.2 Lyrics only

We use a word embedding as input to the network, i.e. each word is embedded in a continuous space and the vectors corresponding to each word are stacked, the input being consequently 2-dimensional. We choose to resort to

Model name	Description
CBOW	Continuous bag-of-words: random forest on top of means of input words embedding
GRU	Single Gated Recurrent Unit (GRU) [4], size 40, dense layers of size 64 and 2, preceded by dropout layers of parameter 0.5
LSTM	Single Long Short-Term Memory (LSTM) [6], size 80, dense layers of size 64 and 2, preceded by dropout layers of parameter 0.5
biLSTM	Single LSTM, size 40, dense layers of size 64 and 2, preceded by dropout layers of parameter 0.5
2LSTMs	Two LSTM layers, of size 40, dense layers of size 64 and 2, preceded by dropout layers of parameter 0.5
ConvNet+LSTM	Convolutional layer with 16 features maps of size (2,2), stride 1, max-pooling of size 2, stride 2, an LSTM layer of size 40 and dense layers of size 32 and 2, preceded by dropout layers of parameter 0.5
2ConvNets+2LSTMs	Two convolutional layers with 16 features maps of size (2,2), stride 1, max-pooling of size 2, stride 2, two LSTM layers of size 40 and dense layers of size 32 and 2, preceded by dropout layers of parameter 0.5

Table 1. Description of lyrics-based models.

a word2vec [21] embedding trained on 1.6 million lyrics, as first results seemed to indicate that this specialized embedding performs better than embedding pretrained on an unspecialized, albeit bigger, dataset. We compare several architectures, with recurrent and convolutional layers. One of them is shown in Fig. 1 (b). We also compare this approach with a simple continuous bag-of-words method that acts as a feature-free baseline. The models that were tested are described in Table 1.

3.3 Fusion

For the fusion model, we reuse the unimodal architecture from which we remove the fully connected layers and concatenate the outputs of each network. On top of this concatenation, we use two fully connected layers with an intermediate vector length of size 100. This architecture is presented in Fig. 1(c). This allows for detection of more complex correlations between modalities. We choose to compare this with a simple late fusion, which is a weighted average of the outputs of the unimodal models, the weight being grid-searched. The mid-level fusion model is referred to as `middleDL` and the late fusion model as `lateDL`.

4. EXPERIMENT

4.1 Dataset

The MSD [2] is a large dataset commonly used for MIR tasks. The tracks are associated with tags from LastFM², some of which are related to mood. We apply the procedure described by Hu and Downie in [11] to select the tags that are akin to a mood description. We then make use of the dataset published by Warriner et al. [30] which associates 14,000 English words with their embedding in Russell’s valence/arousal space. We use it for embedding pre-

viously selected tags into the valence/arousal space. When several tags are associated with the same track, we retain the mean of the embedding values. Finally, we normalize the database by centering and reducing valence and arousal. It would undoubtedly be more accurate to have tracks directly labeled with valence/arousal values by humans, but no database with sufficient volume exists. An advantage of this procedure is its applicability to different mood representations, and thus to different existing databases.

The raw audio signal and lyrics are not provided in the MSD. Only features are available, namely MFCCs for audio, word-counts for lyrics. For this reason, we use a mapping between the MSD and the Deezer catalog using the song metadata (song title, artist name, album title) and have then access to raw audio signals and original lyrics for a part of the songs. As a result, we collected a dataset of 18,644 annotated tracks. We note that lyrics and audio are not synchronized. Automatic synchronization being outside of the scope of this work, we resort to a simple heuristic for audio-lyrics alignment. It consists of aligning both modalities proportionally based on their respective length, i.e. for a certain audio segment, we extract words from the lyrics that are at the corresponding location relatively to the length of the lyrics. We release the labels, along with Deezer song identifiers, MSD identifiers, artist and track name³. More data can be retrieved using the Deezer API⁴. Unfortunately, we cannot release the lyrics and music, due to rights restrictions.

We train the models on approximately 60% of the dataset, and validate their parameters with another 20%. Each model is then tested on the remaining 20%. We refer to these three sets as training, validation and test set, respectively. We split the dataset randomly, with the constraint that songs by the same artist must not appear in two different sets (since artist and moods may be correlated).

4.2 Implementation details

For audio, we use a mel-spectrogram as input to the network, with 40 mel-filters and 1024 sample-long Hann window with no overlapping, with a sampling frequency of 44.1kHz, computed with YAAFE [20]. We use data augmentation, that was investigated for audio and proven useful in [25], in order to grow our dataset. First, we decide to extract 30 second long segments from the original track. The input of the network is consequently of size 40*1292. We choose to sample seven extracts per track: we draw them uniformly from the song. We also use pitch shifting and lossy encoding, which are transformations with which emotion is invariant, and get three extra segments per original sample. In the end, we get a 28-fold increase in the size of the training set.

For lyrics, the input word embedding was computed with gensim’s implementation of word2vec [23] and we used 100-dimensional vectors. We use data augmentation

² <http://www.last.fm/>

³ https://github.com/deezer/deezer_mood_detection_dataset

⁴ <https://developers.deezer.com/api>

mode	model	valence	arousal
audio	CA	0.118	0.197
	ConvNet	0.179	0.235
lyrics	CA	0.140	0.032
	CBOW	0.080	0.031
	LSTM	0.117	0.027
	GRU	0.106	0.017
	biLSTM	0.076	0.017
	2LSTMs	0.128	0.024
	ConvNet+LSTM	0.134	0.026
bimodal	2ConvNets+2LSTMs	0.127	0.022
	CA	0.219	0.216
	LateDL	0.194	0.235
	middleDL	0.219	0.232

Table 2. R^2 scores of the different tested approaches.

for lyrics as well by extracting seven 50-word segments from each track. Consequently, the input of each neural network is of size 100×50 .

4.3 Results

We present the results and compare in particular deep learning approaches with classical ones. The results are presented in Tab. 2 and 3. In the latter, CA refers to classical models (described in Sect. 2).

Unimodal approaches. The results of each unimodal model are given in Table 2. For lyrics-based ones, we have tested several models without feature engineering. The highest performing method, on both validation and test set, is based on both recurrent and convolutional layers. In the following, we choose this model as the one to be compared with classical models.

For both unimodal models, one can see a similar trend for classical and deep learning approaches: lyrics and audio achieve relatively similar performance on valence detection, whereas audio clearly outperforms lyrics when it comes to arousal prediction. This is unsurprising, as arousal is closely related to rhythm and energy, which are essentially induced by the audio signal. On the contrary, valence is explained by both lyrics and audio, indicating that the positivity of an emotion can be conveyed through the text as well as through the melody, the harmony, the rhythm, etc. Similar observations were made by Laurier et al. [16], where angry and calm songs were classified significantly better by audio than by lyrics, and happy and sad songs were equally well-classified by both modalities. This is consistent with our observations, as happy and sad emotions can be characterized by high and low valence, and angry and calm emotions by high and low arousal.

When looking more closely at the results, one can observe that deep learning approaches are much higher performing than classical ones when it comes to prediction based on audio. On the contrary, classical lyrics-based models are higher performing than our deep learning model, in particular when it comes to valence detection, which is the most informative task for the study on lyrics only (as stated above). The reason can be that classical sys-

tems resort to several emotion related lexicons designed by psychological studies. On the contrary, classical audio feature engineering for mood detection does not make use of such external resources curated by experts.

Late fusion analysis. As stated earlier, the late fusion consists of a simple optimal weighted average between the prediction of both unimodal models. We resort to a grid-search on the value of the weighting between 0 and 1. The result for the reimplementation of traditional approaches and for our model is presented in Table 3. One can observe a similar phenomenon for both classical models and ours. In both cases, the fusion of the modalities does not significantly improve arousal detection performance compared to audio-based models. It is as predicted, as we saw that audio-based models perform significantly better than lyrics-based ones. For deep learning models, using lyrics in addition to audio in a late fusion scheme leads to no improvement, so there is no gain added by using lyrics. When it comes to valence detection, both modalities are valuable: in both approaches, the top performing model is a relatively balanced average of unimodal predictions. Here also, these observations generalize to valence/arousal what was observed on the emotions happy, sad, angry and calm in [16]. Indeed, based on this study, not only are lyrics and audio equally performant for predicting happy and sad songs, but they are also complementary, so that fused models can achieve notably better accuracies. However, predicting angry and calm songs is not improved when using lyrics in addition to audio.

Bimodal approaches comparison. Bimodal method performances are reported in Table 2. Several interesting remarks can be made based on these results. First of all, one can notice that if one compares late fusion for both approaches, arousal detection is outperformed by deep learning systems, as the corresponding unimodal approach based on audio is more performant, and we have seen that lyrics-based arousal detection is in both cases performing poorly. On the contrary, late fusion for valence detection yields better results for classical systems. In this case, the lack of performance of lyrics-based methods relying on deep learning is not compensated for by a slightly improved audio-based performance.

However, when it comes to mid-level fusion presented in paragraph 3.3, there is a clear improvement for valence detection. It seems to indicate that there might be earlier correlations between both modalities, that our model is able to detect. Concerning arousal detection, the capacity of the network to unveil such correlations seems useless: we have seen that our lyrics-based model is not able to bring additional information to the audio-based model.

This performing fusion, along with more accurately predicted valence thanks to audio, is sufficient for achieving similar performance to classical approaches, without the use of any external data designed by experts. Interestingly, both models remain useful, as long as they learn complementary information. For valence detection, an optimized weighted average of the predictions of both models yields the performance presented in Table 4. We can see

	coefficient*	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Feature engineering approaches	valence	0.133	0.163	0.186	0.201	0.211	0.211	0.207	0.192	0.174	0.147	0.112
	arousal	0.034	0.081	0.121	0.152	0.178	0.199	0.211	0.217	0.218	0.212	0.201
Deep learning approaches	valence	0.118	0.136	0.152	0.165	0.175	0.182	0.186	0.188	0.187	0.183	0.177
	arousal	0.025	0.065	0.102	0.135	0.164	0.19	0.212	0.231	0.246	0.257	0.265

Table 3. R^2 scores of the late fusion of unimodal models for classical approaches and deep learning approaches, for different values of weighting. *This coefficient is the weight of the audio prediction. The weight of the lyrics prediction is its complementary to one.

modalities	BWC*	CA and DL mean	CA	DL
audio	0.7	0.193	0.118	0.179
lyrics	0.5	0.177	0.140	0.134
fused	0.5	0.243	0.219	0.219

Table 4. R^2 scores of the optimal weighted mean of classical and deep learning approaches for valence prediction for different modalities. *BWC: best weighting coefficient. This coefficient is the optimal weight of the deep learning-based prediction. CA and DL respectively refers to classical approaches and deep learning methods.

a significant gain obtained for a balanced average of both predictions, indicating that both models have different applications, in particular when it comes to lyrics-based valence detection.

5. CONCLUSION AND FUTURE WORK

We have shown that multimodal mood prediction can go without feature engineering, as deep learning-based models achieve better results than classical approaches on arousal detection, and both methods perform equally on valence detection. It seems that this gain of performance is the results of the capacity of our model to unveil and use mid-level correlations between audio and lyrics, particularly when it comes to predicting valence, as we have seen that for this task, both modalities are equally important.

The gain of performance obtained when using this fusion instead of late fusion indicates that further work can be done for understanding correlations between both modalities, and there is no doubt that a database with synchronized lyrics and audio would be of great help to go further. Future work could also rely on a database with labels indicating the degree of ambiguity of the mood of a track, as we know that in some cases, there can be significant variability between listeners. Such databases would be particularly helpful to go further in understanding musical emotion. Temporally localized label in sufficient volume can also be of particular interest. Future work could also leverage unsupervised pretraining to deep learning models, as unlabeled data can be easier to find in high volume. We also leave it as a future work to pursue improvements of lyrics-based models, with deeper architectures or by optimizing word embeddings used as input. Studying and optimizing in detail ConvNets for music mood detection offers the opportunity to temporally localize zones responsible for the valence and arousal of a track, which could be

of paramount importance to understand how music, lyrics and mood are correlated. Finally, by learning from feature engineering approaches, one could use external resources designed by psychological studies to improve significantly the prediction accuracy, as indicated by the complementarity of both approaches.

6. ACKNOWLEDGMENTS

The authors kindly thank Geoffroy Peeters and Gabriel Meseguer Brocal for their insights as well as Matt Mould for his proof-reading. The research leading to this work benefited from the WASABI project supported by the French National Research Agency (contract ANR-16-CE23-0017-01).

7. REFERENCES

- [1] Shlomo Argamon, Marin Šarić, and Sterling S Stein. Style mining of electronic messages for multiple authorship discrimination: first results. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 475–480. ACM, 2003.
- [2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR*, 2011.
- [3] Mireille Besson, Frederique Faita, Isabelle Peretz, A-M Bonnel, and Jean Requin. Singing in the brain: Independence of lyrics and tunes. *Psychological Science*, 9(6):494–498, 1998.
- [4] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.
- [5] J Stephen Downie. The music information retrieval evaluation exchange (mirex). *D-Lib Magazine*, 12(12):795–825, 2006.
- [6] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. 1999.
- [7] Kate Hevner. Experimental studies of the elements of expression in music. *The American Journal of Psychology*, 48(2):246–268, 1936.

- [8] Xiao Hu, Kahyun Choi, and J Stephen Downie. A framework for evaluating multimodal music mood classification. *Journal of the Association for Information Science and Technology*, 2016.
- [9] Xiao Hu and J Stephen Downie. Improving mood classification in music digital libraries by combining lyrics and audio. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 159–168. ACM, 2010.
- [10] Xiao Hu and J Stephen Downie. When lyrics outperform audio for music mood classification: A feature analysis. In *ISMIR*, pages 619–624, 2010.
- [11] Xiao Hu, J Stephen Downie, and Andreas F Ehmann. Lyric text mining in music mood classification. *American music*, 183(5,049):2–209, 2009.
- [12] Moyuan Huang, Wenge Rong, Tom Arjannikov, Nan Jiang, and Zhang Xiong. Bi-modal deep boltzmann machine based musical emotion classification. In *International Conference on Artificial Neural Networks*, pages 199–207. Springer, 2016.
- [13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.
- [14] Byungsoo Jeon, Chanju Kim, Adrian Kim, Dongwon Kim, Jangyeon Park, and Jung-Woo Ha. Music emotion recognition via end-to-end multimodal neural networks.
- [15] Youngmoo E Kim, Erik M Schmidt, Raymond Migneco, Brandon G Morton, Patrick Richardson, Jeffrey Scott, Jacquelin A Speck, and Douglas Turnbull. Music emotion recognition: A state of the art review. In *ISMIR*, pages 255–266, 2010.
- [16] Cyril Laurier, Jens Grivolla, and Perfecto Herrera. Multimodal music mood classification using audio and lyrics. In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*, pages 688–693. IEEE, 2008.
- [17] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE, 2010.
- [18] Tao Li and Mitsunori Ogihara. Detecting emotion in music. In *ISMIR*, pages 239–240. Johns Hopkins University, 2003.
- [19] Thomas Lidy and Alexander Schindler. Parallel convolutional neural networks for music genre and mood classification. *MIREX*, 2016.
- [20] Benoit Mathieu, Slim Essid, Thomas Fillon, Jacques Prado, and Gaël Richard. Yaafe, an easy to use and efficient audio feature extraction software. In *ISMIR*, pages 441–446, 2010.
- [21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [22] Geoffroy Peeters. A Generic Training and Classification System for MIREX08 Classification Tasks: Audio Music Mood, Audio Genre, Audio Artist and Audio Tag. In *MIREX*, Philadelphia, United States, September 2008.
- [23] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. <http://is.muni.cz/publication/884893/en>.
- [24] James A Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161, 1980.
- [25] Jan Schluter and Sebastian Bock. Improved musical onset detection with convolutional neural networks. In *Acoustics, speech and signal processing (icassp), 2014 IEEE international conference on*, pages 6979–6983. IEEE, 2014.
- [26] Feng Su and Hao Xue. Graph-based multimodal music mood classification in discriminative latent space. In *International Conference on Multimedia Modeling*, pages 152–163. Springer, 2017.
- [27] George Trigeorgis, Fabien Ringeval, Raymond Brueckner, Erik Marchi, Mihalis A Nicolaou, Björn Schuller, and Stefanos Zafeiriou. Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5200–5204. IEEE, 2016.
- [28] George Tzanetakis. Marsyas submissions to MIREX 2007. In *ISMIR*, 2007.
- [29] Xing Wang, Xiaou Chen, Deshun Yang, and Yuqian Wu. Music emotion classification of chinese songs based on lyrics using tf* idf and rhyme. In *ISMIR*, pages 765–770, 2011.
- [30] Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods*, 45(4):1191–1207, 2013.
- [31] Dan Yang and Won-Sook Lee. Disambiguating music emotion using software agents. In *ISMIR*, volume 4, pages 218–223, 2004.

IDENTIFYING EMOTIONS IN OPERA SINGING: IMPLICATIONS OF ADVERSE ACOUSTIC CONDITIONS

Emilia Parada-Cabaleiro¹ Maximilian Schmitt¹ Anton Batliner¹
Simone Hantke^{1,2} Giovanni Costantini³ Klaus Scherer⁴ Björn W. Schuller^{1,5}

¹ZD.B Chair of Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, Germany

²Machine Intelligence & Signal Processing Group, Technische Universität München, Germany

³Department of Electronic Engineering, University of Rome Tor Vergata, Italy

⁴Department of Psychology, University of Geneva, Switzerland

⁵GLAM – Group on Language, Audio & Music, Imperial College London, UK

emilia.parada-cabaleiro@informatik.uni-augsburg.de

ABSTRACT

The expression of emotion is an inherent aspect in singing, especially in operatic voice. Yet, adverse acoustic conditions, as, e. g., a performance in open-air, or a noisy analog recording, may affect its perception. State-of-the art methods for emotional speech evaluation have been applied to operatic voice, such as perception experiments, acoustic analyses, and machine learning techniques. Still, the extent to which adverse acoustic conditions may impair listeners' and machines' identification of emotion in vocal cues has only been investigated in the realm of speech. For our study, 132 listeners evaluated 390 nonsense operatic sung instances of five basic emotions, affected by three noises (brown, pink, and white), each at four Signal-to-Noise Ratios (-1 dB, -0.5 dB, +1 dB, and +3 dB); the performance of state-of-the-art automatic recognition methods was evaluated as well. Our findings show that the three noises affect similarly female and male singers and that listeners' gender did not play a role. Human perception and automatic classification display similar confusion and recognition patterns: sadness is identified best, fear worst; low aroused emotions display higher confusion.

1. INTRODUCTION

Singing is a channel to communicate emotion that goes beyond culture or time, as shown by a variety of common musical representations across the world over centuries: as, e. g., lullabies [39] (typical expression of parental love) or spiritual chant [20] (typical expression of mystic feelings). In western music, the emotional expression in singing voice is inexorably linked to the *Italian Opera* which has

had, from the XVIII century (through the development of the *belcanto* [38]) till the XIX century (with the advent of the *Melodramma Verdiano* [38]) a focus on the dramatic-emotional interpretation of the opera's characters [10].

The *Opera* was born in Italy at the beginning of the XVII century as an 'entertainment' [12]. Even though Opera is no longer the most common leisure activity, its cultural importance is still shown by thousands of 'opera performances' made every year—6,795 only in Germany for the 2015/2016 season¹; and by thousands of 'opera recordings' available in multi-media libraries—21,054 items only in the *Istituto Centrale per i Beni Sonori ed Audiovisivi* (The National Italian Audiovisual Institute²). Yet, opera may face 'real-world' acoustic degradation, e. g., from open-air performances [3] or from analog recordings [22]. Indeed, improving the acoustics of an opera house is a central topic of sound engineering [2], as well as the application of digital signal processing solutions to the restoration of old recordings [11].

Even though emotion in opera singing has been studied from the perceptual [34], acoustic [32], and automatic recognition [7] point of view, it has not been evaluated so far up to which extent restricted acoustic quality affects the perception and classification of emotion in singing. In this regard, we present a perceptual study (based on a forced-choice categorical [6] and dimensional [28] test), performed by 132 Italian listeners, who evaluated 390 nonsense instances, sung by 6 professional opera singers (3 female), in 5 emotional states (hot anger, elated happiness, depressive sadness, panicked fear, and worried fear), subsequently masked by three noises (white, pink, and brown) at 4 signal-to-noise ratios (-1 dB, -0.5 dB, +1 dB, and +3 dB). The performance of state-of-the-art emotion recognition methods based on a Support Vector Machine classifier and ComParE features [36] is evaluated as well. In Section 2, related work is described; Sections 3 and 4 evaluate the database and the listening test; Section 5 discusses the results for the machine learning approach; finally, Section 6 outlines conclusions and future work.



© Emilia Parada-Cabaleiro, Maximilian Schmitt, Anton Batliner, Simone Hantke, Giovanni Costantini, Klaus Scherer, Björn W. Schuller. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Emilia Parada-Cabaleiro, Maximilian Schmitt, Anton Batliner, Simone Hantke, Giovanni Costantini, Klaus Scherer, Björn W. Schuller. "Identifying Emotions in Opera Singing: Implications of Adverse Acoustic Conditions", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹<http://operabase.com/top.cgi?lang>

²<http://opac2.icbsa.it/vufind/>



Figure 1: The nonsense utterance ‘Ne kal ibam soud molen’ sung in an ascending scale for each emotional state.

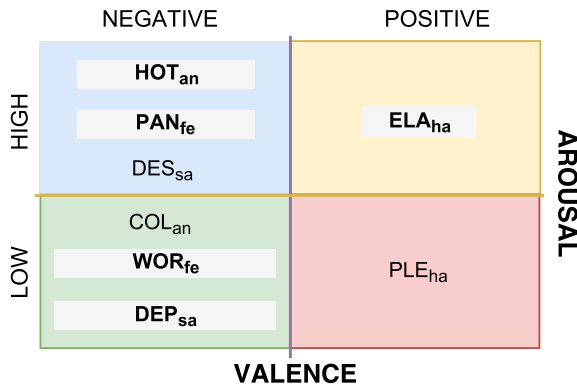


Figure 2: Correspondence between emotion categories and the bi-dimensional model of the five ‘real’ labels, i.e., hot anger (HOT_{an}), elated happiness (ELA_{ha}), depressive sadness (DEP_{sa}), panicked fear (PAN_{fe}), and worried fear (WOR_{fe}), in bold; and the three dimensional ‘distractors’, i.e., cold anger (COL_{an}), pleased happiness (PLE_{ha}), and desperate sadness (DES_{sa}), considered in the perception test.

2. RELATED WORK

Even though emotions are typically expressed through the voice, emotional singing has received little attention compared to emotional speech [29]. Yet, the similarity between both channels (i.e., speech and singing [19, 33]) has recently encouraged researchers to analyse the expression and perception of sung emotional content [4]. Methods typically used in emotional speech research [1] have also been applied to singing—with special attention to the operatic voice—such as acoustic evaluation [23, 27, 32, 37] or perception assessment [15, 16, 34]. Furthermore, in the realm of affective computing, state-of-the-art machine learning techniques, typically used in audio signal processing for speech emotion recognition, have also been applied to the study of the *a cappella* singing voice [7, 40].

In the assessment of emotional speech, it has been shown that listeners’ perception, acoustic feature analysis, and machine learning techniques, are affected by noisy backgrounds [25, 35], which are typical of ‘real-world’ environments and recordings. Yet, although singing mostly takes place in adverse acoustic conditions, the extent to which these may impair a listener’s ability to perceive its inherent emotion, and how the robustness of automatic systems for emotion recognition in singing might be impaired, has not been, to the best of our knowledge, assessed so far.

3. METHODOLOGY

3.1 An Emotional Corpus of a *Cappella* Opera Singing

We took into account a selection of sentences from a dataset of the emotional singing voice [7, 33] in which professional opera singers performed a variety of sentences

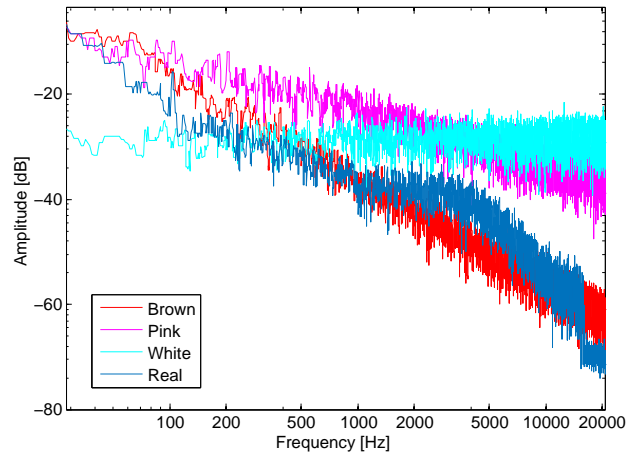


Figure 3: A comparison of the spectral distribution between 0–2 kHz and -80–0 dB, for the brown, pink, white, and real noise.

in different emotional states correlated to several levels of arousal (intensity) and valence (hedonistic value). Since linguistic meaning may influence listener perception of the emotional content, in order to avoid such a bias [31], the nonsense sentence *ne kal ibam soud molen!* has been considered. For a gender-balanced distribution of voice types, six singers have been selected: three females (two sopranos and one mezzosoprano) and three male (two tenors and one countertenor), who produced five times the nonsense sentence with an ascending scale (cf. Figure 1), each time expressing a different emotional state.

Following previous research on the perception of emotion in operatic voice [16], four basic emotions have been considered: anger, with high arousal (intensity), i.e., hot anger; happiness, high aroused, i.e., elated happiness; sadness, low aroused, i.e., depressive sadness; and fear, with both high arousal, i.e., panicked fear, and low arousal, i.e., worried fear (cf. Figure 2). Thus, considering one nonsense sentence, expressed in five emotional states by six singers, 30 ‘clean’ stimuli in total have been employed.

3.2 Manipulation Techniques

The perception of emotion in speech is especially compromised by pink, and to a lesser extent by white and brown noise [25]. In Figure 3, the spectrum of a ‘real’ background noise, digitised from a ‘no-musical fragment’ of an LP recording³, is compared with brown, pink, and white noise. The ‘real’ noise displays higher energy in the lowest frequencies, presenting a negative slope of approximately 6 dB per octave up to 1 kHz, a constant area from 1 kHz to 3 kHz, and a fall of energy of approximately 10 dB per octave above 3 kHz. Its acoustic characteristics makes it most similar to brown noise, which presents a negative

³ Recording of the aria *Vissi d’amore* (Puccini’s *Tosca*), interpreted by Giannina Arangi and produced in 1932 by *Columbia records*.

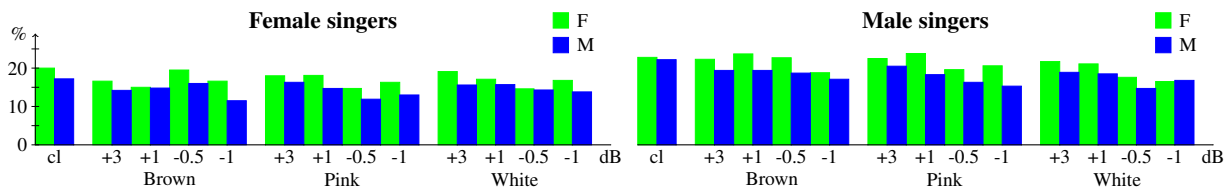


Figure 4: Mean accuracy in % of the ‘real’ emotions (cf. caption Figure 2) perceived by female (F) and male (M) listeners; in clean (cl) conditions, background noises (brown, pink, and white), and 4 SNR (-1 dB, -0.5 dB, +1 dB, and +3 dB); sung by females and males.

slope of around 6 dB per octave ($1/f^2$ noise); slightly similar to pink, with a negative slope of approximately 3 dB per octave ($1/f$ noise); and dissimilar to white, whose flat spectrum presents all the frequencies at the same level. Note that the given comparison aims at exemplifying potential similarities between ‘real’ and ‘artificially generated’ noise; noise from different recordings may display higher similarity with pink, white, or other noise types.

We evaluated listeners’ perception of emotion in adverse acoustic conditions by applying four Signal-to-Noise Ratio (SNR) levels (-1 dB, -0.5 dB, +1 dB, +3 dB) and three noises (brown, pink, white) to the ‘clean’ samples. The noises, normalized to -1 dB, have been artificially generated and mixed (at the specified SNR value) in *Matlab R2014a* [21]. Given 6 singers, 1 sentence sung in 5 emotional states, 3 noises, and 4 applied SNR levels yields $6 \times 1 \times 5 \times 3 \times 4 = 360$ ‘noisy’ samples plus 30 ‘clean’ samples = 390 stimuli in total.

4. PERCEPTION STUDY

4.1 Emotion Measurement

The two prominent models considered to evaluate listeners’ perception of emotional speech, i.e., the categorical [6], which identifies each emotional state with a specific category, and the dimensional [28], which identifies each emotional state within a continuous hyper-space characterised by dimensions—commonly arousal (from low to high) and valence (from negative to positive)—have already been applied to the perceptual evaluation of emotion in singing [16,26]. Yet, which of them would be more suitable to evaluate listeners’ perception of emotion, is still an open question in both the musical domain [5] and speech research [18]. Both models have been taken into account for the perception test, i.e., each of the 4 considered basic emotions—anger, happiness, sadness, and fear (cf. Section 3.1)—has been defined in the bi-dimensional space, by having a level of arousal and valence (cf. Figure 2).

Five of these eight emotional categories (hot anger, elated happiness, depressive sadness, panicked fear, and worried fear), are ‘real’ emotions effectively expressed by the singers in the dataset. The other three (cold anger, pleased happiness, and desperate sadness), so-called ‘distractor labels’ [24]—emotion categories not displayed in the evaluated data, have the purpose to ‘distract’ the listeners by minimising the chances of performing ‘discrimination’ rather than ‘recognition’ [30]. Furthermore, disgust and surprise (the remaining two basic emotions—in addition to anger, fear, sadness, and happiness—amongst those

known as ‘big six’ [6]), have also been considered as ‘distractors’, without indicating a specific dimensional level; we thus present a balanced set of perceptual choices: five ‘real’ emotions and five ‘distractors’.

4.2 Listening test setup

In total, 132 Italian listeners (55 f, 77 m, mean age 20.7 years, standard deviation 2.5 years) took part in the perception study. The participants were all students of the engineering faculty of the ‘Tor Vergata’ university (Rome) and received credits for their participation. To avoid fatigue, the 390 stimulus were similarly distributed into four sessions, each designed to last not longer than 30 minutes. Out of the 132 listeners, 101 had no musical instruction, 27 were self-taught in piano or guitar, 4 had studied in the conservatory—piano (2), flute, and accordion. Their musical interest was mostly in pop (65 listener), rock (45 listeners), and hip-hop (22 listeners); other genres as, e.g., Italian music, heavy-metal, or classic were underrepresented (less than 10 listeners). Since none of them had studied singing or demonstrated interest in opera, we consider them as a unique group of non-experts.

The test was designed as a forced-choice task; the ten emotion categories were presented and the participants could choose one out of them after listening to each stimulus (an initial training was provided). The test was hosted on a browser based interface (accessible from any computer) provided through the gamified crowd-sourcing platform *iHEARu-PLAY* [14]. To ensure a consistent listening environment, the participants were instructed to use earphones. Although the listeners had the possibility of listening to each stimulus indefinitely, they were encouraged to answer spontaneously to the randomized samples.

4.3 Results and discussion

Emotions were identified best in clean conditions; female listeners were slightly more accurate than male; emotions in male voices were somewhat better identified than in female voices (cf. Figure 4). Listeners’ and singers’ gender-related differences turned out to be not significant. In the former case, the biggest distance, i.e., female and male listeners evaluating male voices in pink noise at -1 SNR (21.6% vs 17.6%), corresponds to a p value in Pearson Chi-square of = .47 (way above the conventional threshold for significance of $p < .05$). In the latter case, the biggest distance, i.e., female and male voices perceived by male listeners in clean conditions (17.2 % vs 22.2 %) did not yield a significant difference either ($p = .37$). Thus, the further evaluations will not consider gender.

%	Real emotions					Distractor labels					#
	<i>HOT_{an}</i>	<i>ELA_{ha}</i>	<i>DEP_{sa}</i>	<i>PAN_{fe}</i>	<i>WOR_{fe}</i>	<i>COL_{an}</i>	<i>PLE_{ha}</i>	<i>DES_{sa}</i>	<i>DIS</i>	<i>SUR</i>	
HOT_{an}	26.6	14.7	05.5	03.2	05.2	26.4	9.2	02.8	01.7	04.5	6
ELA_{ha}	13.1	18.4	07.6	04.4	05.9	19.7	18.3	04.1	03.3	05.2	6
DEP_{sa}	01.0	03.6	42.0	03.6	05.7	07.0	12.3	21.4	01.2	02.2	6
PAN_{fe}	09.8	06.1	22.8	06.8	06.6	19.7	12.0	09.6	02.3	04.3	6
WOR_{fe}	12.6	08.8	14.4	08.2	08.5	21.2	14.2	05.1	02.2	04.9	6
total	63.1	51.7	92.3	26.1	31.9	93.9	66.4	43.0	10.7	21.1	30

Table 1: Listeners’ perception (in %) of the clean instances (#), considering ‘real’ emotions and ‘distractors’ (cf. Figure 2, disgust—DIS, and surprise—SUR). Each row gives the ‘reference’, darker cells indicate higher % ; listeners’ and singers’ gender is not considered.

%	<i>HOT_{an}</i>	<i>ELA_{ha}</i>	<i>DEP_{sa}</i>	<i>PAN_{fe}</i>	<i>WOR_{fe}</i>	mean
cl	26.6	18.4	42.0	06.8	08.5	20.5
br	13.7	10.9	42.9	04.8	06.5	15.8
pi	12.9	09.5	45.1	05.2	11.5	17.0
wh	10.0	09.1	49.7	04.8	08.5	16.4

Table 2: Perception accuracy (in %) of *HOT_{an}*, *ELA_{ha}*, *DEP_{sa}*, *PAN_{fe}*, and *WOR_{fe}* (cf. Figure 2), in clean (cl) and noisy background: brown (br), pink (pi), white (wh) at -1 dB SNR. Mean accuracy is given; each row gives results for 30 instances.

%	<i>HOT_{an}</i>	<i>ELA_{ha}</i>	<i>DEP_{sa}</i>	<i>PAN_{fe}</i>	<i>WOR_{fe}</i>	<i>COL_{an}</i>
cl	63.1	51.7	92.3	26.1	31.9	93.9
br	36.3	41.4	112.9	26.3	32.4	109.1
pi	44.1	37.9	138.1	26.7	27.7	107.2
wh	36.3	34.8	125.1	25.6	28.8	117.7

Table 3: Sum of columns (in %) ‘perceived as’ for the ‘real’ emotions: *HOT_{an}*, *ELA_{ha}*, *DEP_{sa}*, *PAN_{fe}*, *WOR_{fe}*; the ‘distractor’ *COL_{an}* (cf. Figure 2), in clean (cl) and -1 dB SNR background: brown (br), pink (pi), white (wh); each row encodes 30 instances.

The results for clean conditions show that the emotional state most accurately perceived is *DEP_{sa}* (42.0%), followed by *HOT_{an}* (26.6%), and *ELA_{ha}* (18.4%); worse recognised were *WOR_{fe}* (08.5%) and *PAN_{fe}* (06.8%). *HOT_{an}* was mainly confused with *COL_{an}*, *ELA_{ha}* with *PLE_{ha}*, and *DES_{sa}* with *DEP_{sa}* (cf. Table 1), suggesting that listeners discriminate better between two different emotions than between two arousal levels of the same emotion. The ‘distractors’ DIS and SUR have been rarely chosen (less than 5.5 %). Confusion between different emotions within the same arousal level took mostly place between *HOT_{an}* vs *ELA_{ha}* (high arousal) and *WOR_{fe}* vs *COL_{an}* (low arousal); this can be explained by the acoustic similarities between them. In Figure 5, the *Chroma*⁴ representation of emotional singing performed by a female singer (soprano) displays that *HOT_{an}* and *DEP_{sa}* are expressed differently. *HOT_{an}*, as shown in acted speech [13], is expressed through articulated prosody, acoustically characterised by a strong decay in amplitude and lower slope declinations, which is displayed by a richer spectrum on partials with less differences between the energy of low and high frequencies. *DEP_{sa}*, on the contrary, is expressed through sustained amplitude for each note, which concentrates more energy in F0 and less in higher harmonics. *ELA_{ha}* presents a spectrum and articulation at mid point between the previous ones.

As expected (apart from a rare exceptions in the perception of female voices at -0.5 dB SNR in brown noise), listeners’ accuracy decreases with the increment of noise (cf. Figure 4), i. e., higher SNR (-1 dB and -0.5 dB) yielded lower accuracy. By evaluating the perception of emotion in clean and -1 dB SNR conditions, (cf. Table 2), *HOT_{an}* and *ELA_{ha}* were affected most by noise, *DEP_{sa}* less, *WOR_{fe}* and *PAN_{fe}* were perceived similarly to clean background. The three noises affected perception in a similar way: brown slightly more (15.8 % mean accu-

racy), pink less (17.0 % mean accuracy). Yet, the higher level of accuracy in pink and white noises is due to an improvement—caused by an increment in the confusion towards low aroused emotions—in the accuracy of *DEP_{sa}*, rather than to a lower detriment in the overall accuracy. This phenomenon relates to an acoustic ‘flattening’ by the noise of the characteristics typical of each emotion, causing perception as sustained, with lower energy, and attenuated articulation, i. e., similarly to low aroused emotions. Indeed, the *chromogram* for *HOT_{an}*, *ELA_{ha}*, and *DEP_{sa}*, masked by pink noise at -1 dB (cf. Figure 6), displays comparable acoustic representation for the three emotions.

To evaluate such phenomena, for each confusion matrix—obtained by the perception in clean and -1 dB SNR conditions—the sum of the columns has been computed, by that counting for each emotion all the responses ‘identified as’ (cf. ‘total’ in Table 1). Confirming previous findings [25], the confusion in background noise mostly increases for the low aroused emotions *DEP_{sa}* and *COL_{an}*, and decreases for the high aroused *HOT_{an}* and *ELA_{ha}* (cf. Table 3). No meaningful differences are displayed for the other emotions across conditions.

5. AUTOMATIC RECOGNITION

5.1 Methods

We employed state-of-the-art methods for emotion recognition of vocal cues by applying a Support Vector Machine (SVM) classifier with linear kernel, from the open-source toolkit LIBLINEAR [9], and the ComParE 2013 challenge features set [36], extracted with OPENSIMILE [8]. Since our goal is to evaluate how background noises may affect the classification performance in general, only state-of-the-art methods for automatic recognition of emotion in the operatic voice [7] have been taken into account.

For speaker independence, we split the 390 instances into three sets (A, B, and C), considering for each 130 in-

⁴ Chroma features have been extracted by OPENSIMILE [8].

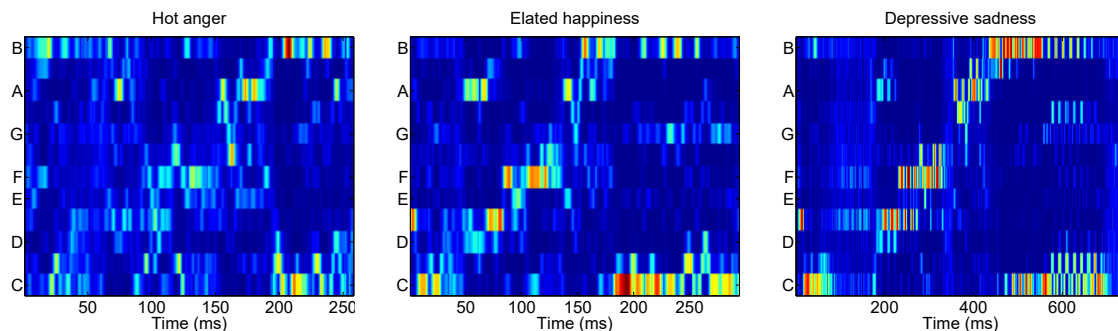


Figure 5: Chroma representation of the instances expressing: Hot_{an} , ELA_{ha} , and DEP_{sa} (from left to right); sung by one of the soprano. The y axis gives the C natural scale; the x axis the time in milliseconds. Dark blue indicates the lower level of energy, red the higher.

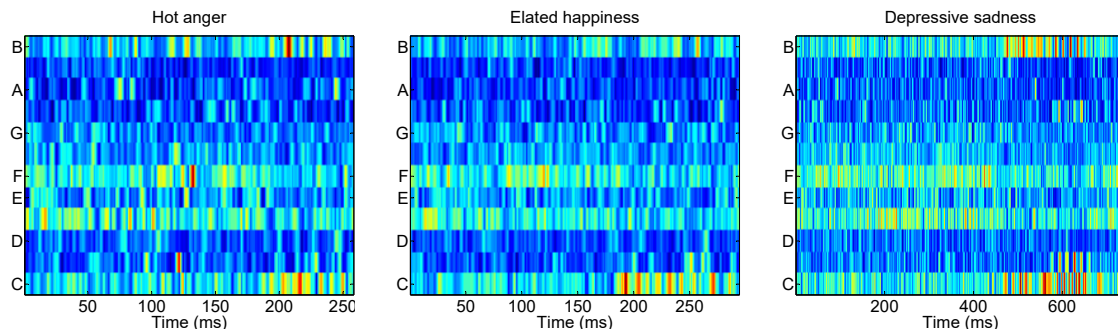


Figure 6: Chroma representation of the instances given in Figure 5 masked by pink noise at -1 dB SNR.

stances sung by two different singers (one female and one male), and performing the experiments in two phases—development and test. For the development phase we considered one set as training (e.g., A), and another as test (e.g., B); 30 levels of complexity (from 2^{30} to 2^0) have been tested to optimise the SVM performance. In the test phase, we merged the sets A and B for training and considered the set C as test; the complexity which achieved best results in the development phase was taken into account as optimisation parameter for the SVM. This procedure was carried out with the six possible permutations between the three sets, and the results were averaged.

We performed binary classification on five classes, i.e., each class was recognised against the other four. In the training phase, the minority class was upsampled to match the sample size of the remaining classes together; for each noise, all the SNR were considered together. We employed the whole ComParE 2013 features set [36], encompassing 6374 acoustic features in total: 64 low-level descriptors—LLD, and several functionals [7], in four sub-sets: mel-frequency cepstral coefficients—mfcc (1,400 features), spectrum (4,300), prosody (183), and voice quality (390).

5.2 Results and discussion

The classification of five classes (cf. Table 4) mirrors the perception findings (cf. Table 2) for all the feature sets: DEP being classified best, HOT and ELA in between, and PAN and WOR worse. The mfcc sub-set performs best, showing the highest Unweighted Average Recall (UAR), i.e., the mean average of the recall per class over the six permutations. In order to visualise these re-

%	HOT	ELA	DEP	PAN	WOR	UAR
ComParE	26.3	28.2	77.6	07.0	18.6	31.5
mfcc	34.6	30.1	71.8	07.7	26.3	34.1
spec	26.9	25.0	82.0	03.8	12.2	30.0
prosody	17.3	22.4	48.1	08.3	21.1	23.5
vq	34.6	27.6	53.2	11.5	14.7	28.3

Table 4: Test classification accuracy and Unweighted Average Recall (UAR) in % for the ‘real’ emotions (HOT, ELA, DEP, PAN, WOR, cf. Figure 2), considering the four conditions—clean and the three noises—together, for each feature set: ComParE, mfcc, spectrum (spec), prosody, and voice quality (vq).

%	ComParE	mfcc	spec	prosody	vq	mean
cl	35.0	40.0	36.6	23.3	25.0	32.0
br	31.6	35.4	30.8	23.7	28.7	30.0
pi	32.0	36.2	28.3	22.9	25.4	29.0
wh	30.0	29.1	29.1	23.7	31.6	28.7

Table 5: UAR for test in % for each feature set (cf. caption of Table 4), in each condition: clean (cl), brown (br), pink (pi), white (wh). In noisy background the 4 SNR are considered together.

sults, in Figure 7, a 2-dimensional *Non-Metrical Multi-Dimensional Scaling* (NMDS, [17]) solution is given. It shows a non-metrical visual representation of the optimal distances between the evaluated categories. DEP, since best recognised—thus classified as different—is more distant to the other classes in all the emotional constellations.

The feature set with the best performance (mfcc, cf. Table 4) displays an arousal related pattern, the high aroused emotions (HOT, ELA, PAN), clustered together, the low aroused (WOR, DEP) more distant. This may relate to the level of energy: higher in the former, lower in the latter (cf. Figure 5). The decline in UAR goes together with the condensation of the emotions in the 2-dim space, as

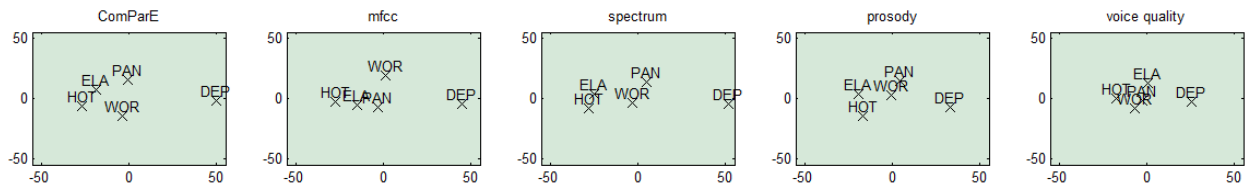


Figure 7: 2-dim NMDS solution for the five feature sets in the classification of the five ‘real’ emotions, considering the 390 instances (cf. caption of Table 4). Kruskal’s stress is given as a measure of fit for 2-dim and 1-dim solution respectively: ComParE (6.3e-07, 4.0e-05); mfcc (3.4e-07, 0.1); spectrum (1.3e-17, 7.5e-16); prosody (6.9e-07, 9.2e-07); voice quality (1.3e-16, 4.8e-05).

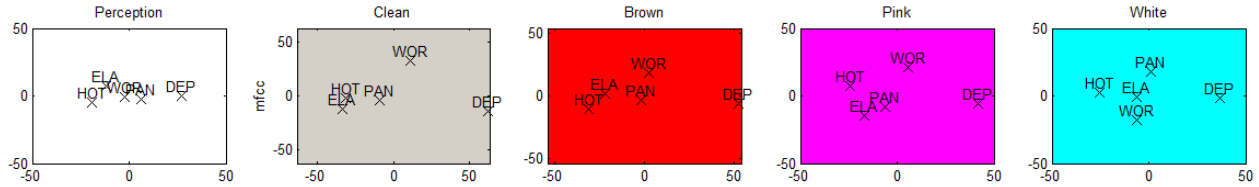


Figure 8: 2-dim NMDS solution for listeners’ perception (in clean condition) and classification (in clean and noisy conditions) with mfcc features, of the five ‘real’ emotions. Kruskal’s stress is given for 2-dim and 1-dim solution respectively: Perception (4.3e-17, 6.6e-07); Clean (1.2e-16, 7.3e-07); Brown (1.3e-16, 0.02), Pink (3.9e-07, 1.3e-05); White (6.9e-07, 3.0e-04).

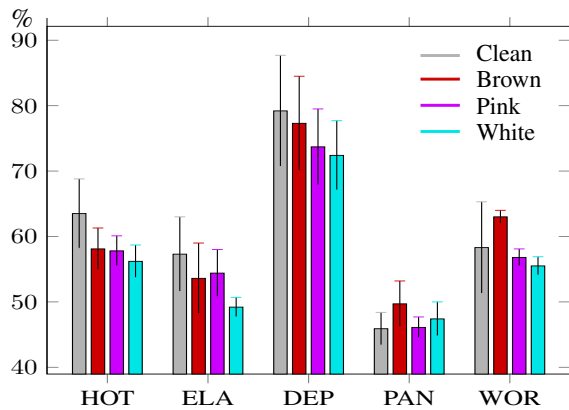


Figure 9: UAR and std in % for binary classification, i. e., each emotion against the other four, in the four conditions (cf. caption of Figure 8), for the mfcc sub-set.

prominently shown for the voice quality sub-set (cf. Figure 7). As for listeners’ perception (cf. Table 2), for mfcc, ComParE, and to some extent for spectral features, highest accuracy was achieved in clean condition, medium in pink and brown, lower in white (cf. Table 5). Prosodic and voice quality features performed worst, which relates to both the considered musical instances and to the operatic technique. On the one hand, the sung melodic contour is the same for all utterances and emotions; thus, there are no degrees of freedom for pitch leftover for the marking of emotions. On the other hand, opera singing is characterised by the ‘projection’ of the voice—a high control of articulation (and by that, mfcc configurations), and a weak use of different voice qualities when expressing emotions, in contrast, for instance, to modern actors or pop singers.

In Figure 8, an NMDS visualisation for perception (in clean condition) and mfcc classification (in the different backgrounds) is given. The confusion in the perceptual constellation relates mainly to the low accuracies achieved in the listening test, which is given mostly by the use of ‘distractors’. As shown in Table 5, classification in clean background yields the highest UAR, which is visually mirrored by the arousal-related pattern previously described, i. e., high aroused emotions clustered together, low aroused

distant (DEP more, WOR less); this is more or less preserved for brown and pink noise but not for white noise with lowest UAR, cf. Table 5.

The binary classification (cf. Figure 9) confirms again the perceptual findings (cf. Table 2): DEP best recognised, HOT and ELA at a medium level, PAN worse. WOR is better classified than perceived, which relates to the spread of the listeners’ responses motivated by the ‘distractor’ COL_{an}. Indeed, WOR—having the same arousal—was mainly misclassified by the listeners as COL_{an}, thus decreasing the perception accuracy of the former. White noise seems to affect binary classification more which might suggest that higher frequencies (more masked in white noise) could be more relevant for the identification of emotion in singing; lower frequencies (more masked in pink and brown noises), since related to pitch—thus to the melodic contour, which is the same for all the samples—might be less relevant for the emotional understanding in this specific study but not in general.

6. CONCLUSIONS

The present study shows that brown, pink, and white noises affect similarly the perception of emotion in operatic singing: the lower the SNR, the lower the perception. Gender seems not to be an influential factor, neither for singers nor for listeners. In general, perception and classification shows analogous emotional constellations regardless the background, sadness being identified best, fear worst. The use of ‘distractors’ influences listeners’ perception, affecting even more the accuracy of fear, an emotion which seems not to have a typical expression in singing; thus it is worse identified and easily confused. Voice quality features perform worst, mfcc best. In the former, this relates to the voice ‘projection’ inherent to opera (which minimise the differences between emotions), in the latter, to the relevance of energy per band to discriminate between sung emotions. Listeners’ low accuracy suggests that identifying emotion in opera singing may be challenging for non trained subjects; thus, musically trained listeners will be considered in future investigations.

7. ACKNOWLEDGEMENT



This work was supported by the European Unions's Seventh Framework and Horizon 2020 Program under grant agreement No. 338164 (ERC StG iHEARu).

8. REFERENCES

- [1] R. Banse and K. R. Scherer, "Acoustic profiles in vocal emotion expression," *Journal of personality and social psychology*, vol. 70, p. 614, 1996.
- [2] L. Beranek, *Concert halls and opera houses: Music and acoustics*. New York, NY: Springer, 2012.
- [3] M. Cognini, A. Farina, and R. Pompili, "L'acustica dell'anfiteatro romano Arena di Verona," in *Proc. of Acoustics and Recovery of Spaces for Music*. Ferrara, Italy: ACM, 1993, pp. 105–118.
- [4] E. Coutinho, K. R. Scherer, and N. Dikken, "Singing and emotion," in *The Oxford handbook of singing*, G. Welch, D. M. Howard, and J. Nix, Eds. Oxford, UK: OUP, 2014.
- [5] T. Eerola and J. K. Vuoskoski, "A comparison of the discrete and dimensional models of emotion in music," *Psychology of Music*, vol. 39, no. 1, pp. 18–49, 2011.
- [6] P. Ekman, "Expression and the nature of emotion," *Approaches to Emotion*, vol. 3, pp. 19–344, 1984.
- [7] F. Eyben, G. L. Salomão, J. Sundberg, K. R. Scherer, and B. W. Schuller, "Emotion in the singing voice – a deeper look at acoustic features in the light of automatic classification," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 1, pp. 1–9, 2015.
- [8] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: The Munich versatile and fast open-source audio feature extractor," in *Proc. of ACM MM*, 2010, pp. 1459–1462.
- [9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *JMLR*, vol. 9, pp. 1871–1874, 2008.
- [10] M. García and D. V. Paschke, *A complete treatise on the art of singing*. New York, NY: Da Capo, 1975.
- [11] S. Godsill, P. Rayner, and O. Cappé, "Digital audio restoration," in *Applications of digital signal processing to audio and acoustics*, M. Kahrs and K. Brandenburg, Eds. Boston, MA: Springer, 2002, pp. 133–194.
- [12] D. J. Grout and C. V. Palisca, *A history of western music*. New York, NY: Norton, 2001.
- [13] M. Guzman, S. Correa, D. Muñoz, and R. Mayerhoff, "Influence on spectral energy distribution of emotional expression," *Journal of Voice*, vol. 27, pp. 129–139, 2013.
- [14] S. Hantke, F. Eyben, T. Appel, and B. Schuller, "iHEARU-PLAY: Introducing a game for crowdsourced data collection for affective computing," in *Proc. of WASA*. Xi'an, China: IEEE, 2015, pp. 891–897.
- [15] P. Howes, J. Callaghan, P. Davis, D. Kenny, and W. Thorpe, "The relationship between measured vibrato characteristics and perception in western operatic singing," *Journal of Voice*, vol. 18, pp. 216–230, 2004.
- [16] S. Jansens, G. Bloothoof, and G. de Krom, "Perception and acoustics of emotions in singing," in *Proc. of Eurospeech*. Rhodes, Greece: ISCA, 1997, pp. 2155–2158.
- [17] J. Kruskal and M. Wish, *Multidimensional Scaling*. London, U.K.: Sage University, 1978.
- [18] P. Laukka, "Vocal expression of emotion: discrete-emotions and dimensional accounts," Ph.D. dissertation, Acta Universitatis Upsaliensis, 2004.
- [19] S. R. Livingstone, K. Peck, and F. A. Russo, "Acoustic differences in the speaking and singing voice," in *Proc. of Meetings on Acoustics*, Montreal, QC, 2013, pp. 1–5.
- [20] I. W. Mabbett, "Buddhism and music," *Asian Music*, vol. 25, no. 1/2, pp. 9–28, 1993.
- [21] I. Mathworks, "Matlab: R2014a," Natick, 2014.
- [22] M. Mauch and S. Ewert, "The audio degradation toolbox and its application to robustness evaluation," in *Proc. of ISMIR*, Curitiba, Brazil, 2013, pp. 83–88.
- [23] V. P. Morozov, "Emotional expressiveness of the singing voice: The role of macrostructural and microstructural modifications of spectra," *Logopedics Phoniatrics Vocology*, vol. 21, pp. 49–58, 1996.
- [24] I. R. Murray and J. L. Arnott, "Implementation and testing of a system for producing emotion-by-rule in synthetic speech," *Speech Comm.*, vol. 16, pp. 369–390, 1995.
- [25] E. Parada-Cabaleiro, A. Baird, A. Batliner, N. Cummins, S. Hantke, and B. Schuller, "The Perception of Emotions in Noisified Non-Sense Speech," in *Proc. of Interspeech*. Stockholm, Sweden: ISCA, 2017, pp. 3246–3250.
- [26] E. Parada-Cabaleiro, A. Baird, A. Batliner, N. Cummins, S. Hantke, and B. W. Schuller, in *Proc. of DLFM*. ACM, 2017, pp. 29–36.
- [27] E. Rapoport, "Emotional expression code in opera and lied singing," *JNMR*, vol. 25, pp. 109–149, 1996.
- [28] J. A. Russell, "A circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39, pp. 1161–1178, 1980.
- [29] K. R. Scherer, "Vocal communication of emotion: A review of research paradigms," *Speech Comm.*, vol. 40, pp. 227–256, 2003.
- [30] —, "Vocal communication of emotion: A review of research paradigms," *Speech Comm.*, vol. 40, pp. 227–256, 2003.
- [31] K. R. Scherer, R. Banse, and H. G. Wallbott, "Emotion inferences from vocal expression correlate across languages and cultures," *Journal of Cross-Cultural Psychology*, vol. 32, pp. 76–92, 2001.
- [32] K. R. Scherer, J. Sundberg, B. Fantini, S. Trznadel, and F. Eyben, "The expression of emotion in the singing voice: Acoustic patterns in vocal performance," *JASA*, vol. 142, pp. 1805–1815, 2017.
- [33] K. R. Scherer, J. Sundberg, L. Tamarit, and G. L. Salomão, "Comparing the acoustic expression of emotion in the speaking and the singing voice," *Computer Speech & Language*, vol. 29, pp. 218–235, 2015.
- [34] K. R. Scherer, S. Trznadel, B. Fantini, and J. Sundberg, "Recognizing emotions in the singing voice," *Psychomusicology: Music, Mind & Brain*, vol. 27, pp. 244–255, 2017.
- [35] B. Schuller, D. Arsić, F. Wallhoff, and G. Rigoll, "Emotion recognition in the noise applying large acoustic feature sets," in *Proc. of Speech Prosody*. Dresden, Germany: ISCA, 2006, pp. 276–289.
- [36] B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, F. Wengler, F. Eyben, E. Marchi *et al.*, "The Interspeech 2013 computational paralinguistics challenge: Social signals, conflict, emotion, autism," in *Proc. of Interspeech*. Lyon, France: ISCA, 2013, pp. 148–152.
- [37] H. Siegwart and K. R. Scherer, "Acoustic concomitants of emotional expression in operatic singing: The case of Lucia in *Ardi gli incensi*," *Journal of Voice*, vol. 9, pp. 249–260, 1995.
- [38] J. Stark, *Bel canto: A history of vocal pedagogy*. London, U.K.: University of Toronto Press, 2003.
- [39] S. Trehub, A. Unyk, and L. Trainor, "Adults identify infant-directed music across cultures," *Infant Behavior and Development*, vol. 16, pp. 193–211, 1993.
- [40] B. Zhang, G. Essl, and E. M. Provost, "Recognizing emotion from singing and speaking using shared models," in *Proc. of ACII*. IEEE, 2015, pp. 139–145.

MUSICAL TEXTURE AND EXPRESSIVITY FEATURES FOR MUSIC EMOTION RECOGNITION

Renato Panda

Ricardo Malheiro

Rui Pedro Paiva

CISUC – Centre for Informatics and Systems, University of Coimbra, Portugal

{panda, rsmal, ruipedro}@dei.uc.pt

ABSTRACT

We present a set of novel emotionally-relevant audio features to help improving the classification of emotions in audio music. First, a review of the state-of-the-art regarding emotion and music was conducted, to understand how the various music concepts may influence human emotions. Next, well known audio frameworks were analyzed, assessing how their extractors relate with the studied musical concepts. The intersection of this data showed an unbalanced representation of the eight musical concepts. Namely, most extractors are low-level and related with tone color, while musical form, musical texture and expressive techniques are lacking. Based on this, we developed a set of new algorithms to capture information related with musical texture and expressive techniques, the two most lacking concepts. To validate our work, a public dataset containing 900 30-second clips, annotated in terms of Russell’s emotion quadrants was created. The inclusion of our features improved the F1-score obtained using the best 100 features by 8.6% (to 76.0%), using support vector machines and 20 repetitions of 10-fold cross-validation.

1. INTRODUCTION

Music Emotion Recognition (MER) research has increased in the last decades, following the growth of music databases and services. This interest is associated to music’s ability to “arouse deep and significant emotions”, being “its primary purpose and the ultimate reason why humans engage with it” [1]. Different problems have been tackled, e.g., music classification [2]–[4], emotion tracking [5], [6], playlists generation [7], [8], exploitation of lyrical information and bimodal approaches [9]–[12]. Still, some limitations affect the entire MER field, among which: 1) the lack of public high-quality datasets, as used in other machine learning fields to compare different works; and 2) the insufficient number of emotionally-relevant acoustic features, which we believe are needed to narrow the existing semantic gap [13] and push the MER research forward. Furthermore, both the state-of-the-art research papers

(e.g., [14], [15]) and MIREX Audio Mood Classification (AMC) comparison¹ results from 2007 to 2017 are still not accurate enough in easier classification problems with four to five emotion classes, let alone higher granularity solutions and regression approaches, showing a glass ceiling in MER system performances [13].

Many of the audio features applied currently in MER were initially proposed to solve other information retrieval problems (e.g. MFCCs and LPCs in speech recognition [16]) and may lack emotional relevance. Therefore, we hypothesize that, in order to advance the MER field, part of the effort needs to focus on one key problem: the design of novel audio features that better capture emotional content in music, currently left out by existing features.

This raises the core question we aim to tackle in this paper: can higher-level features, namely expressivity and musical texture features, improve emotional content detection in a song?

In addition, we have constructed a dataset to validate our work, which we consider better suited to the current MER state-of-the-art: avoids overly complex or unvalidated taxonomies, by using the four classes or quadrants, derived from the Russell’s emotion model [17]; does not require a full manual annotation process, by using AllMusic annotations and data², with a simpler human validation, thus reducing resources needed.

We achieved an improvement of up to 7.9% in F1-Score by adding our novel features to the baseline set of state-of-the-art features. Moreover, even when the top 800 baseline features is employed, the result is 4.3% below the one obtained with the top100 baseline and novel features set.

This paper is organized as follows. Section 2 reviews the related work. Section 3 describes the musical concepts and related state-of-the-art audio features. Dataset acquisition, the novel audio features design and classification strategies are also presented. In Section 4, experimental results are discussed. Conclusions and future work are drawn in Section 5.

2. RELATED WORK

Emotions have been a research topic for centuries, leading to the proposal of different emotion paradigms (e.g., categorical or dimensional) and associated taxonomies (e.g.,



© Renato Panda, Ricardo Malheiro, Rui Pedro Paiva.
Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Renato Panda, Ricardo Malheiro, Rui Pedro Paiva. “Musical Texture and Expressivity Features for Music Emotion Recognition”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ <http://www.music-ir.org/mirex/>

² <https://www.allmusic.com/moods>

Hevner, Russell) [17], [18]. More recently, these have been employed in many MER computational systems, e.g., [2]–[7], [9], [12], [19], [20], and MER datasets, e.g., [4], [6], [20].

Regarding emotion in music, it can be viewed as: i) the perceived emotion, identified when listening; ii) emotion felt, representing the emotion felt when listening, which may be different from the perceived; iii) or the emotion transmitted, which is the emotion a performer intended to deliver. This work is focused on perceived emotions, since it is more intersubjective, as opposed to emotion felt, more personal and dependent of context, memories and culture.

As for associations between emotions and musical attributes, many features such as: articulation, dynamics, harmony, loudness, melody, mode, musical form, pitch, rhythm, timbre, timing, tonality or vibrato have been previously linked to emotion [8], [21], [22]. However, many are yet to be fully understood, still requiring further research, while others are hard to extract from audio signals. These musical attributes can be organized into eight different categories, each representing a core concept, namely: dynamics, expressive techniques, harmony, melody, musical form, musical texture, rhythm and tone color (or timbre). Several audio features have been created (hereinafter referred to as standard audio or baseline features) and are nowadays implemented in audio frameworks (e.g. Marsyas [23], MIR toolbox [24] or PsySound [25]). Even though hundreds of features exist, most belong to the same category – tone color, while others were developed to solve previous research problems and thus might not be suited for MER (e.g., Mel-frequency cepstral coefficients (MFCCs) for speech recognition). On the other hand, the remaining categories are underrepresented, with expressivity, musical texture or form nearly absent.

Finally, as opposed to other information retrieval fields, MER researchers lack standard public datasets and benchmarks to compare existent works' adequately. As a consequence, researchers use private datasets (e.g., [26]), or have access only to features and not the actual audio (e.g., [27]). While efforts such as the MIREX AMC task improve the situation, issues have been identified. To begin with, the dataset is private, use in the annual contest only. Also, it uses an unvalidated taxonomy derived from data containing semantic and acoustic overlap [3].

3. METHODS

In this section, due to the abovementioned reasons, we start by introducing the dataset built to validate our work. Following, we detail the proposed novel audio features and emotion classification strategies tested.

3.1 Dataset Creation

To bypass the limitations described in Section 2 we have created a novel dataset based using an accepted and validated psychological model. We decided on Russell's circumplex model [17], which allows us to employ a simple

taxonomy of four emotion categories, based on the quadrants resulting from the division by the arousal and valence (AV) axes).

First, we obtained music data (30-second audio clips) and metadata (e.g., artist, title, mood and genre) from the AllMusic API¹. The mood metadata consisted of several tags per song, from a list of 289 moods. These 289 tags are intersected with the Warriner's list [28] – an improvement on ANEW adjectives list [29], containing 13915 English words with AV ratings according to Russell's model. This intersection results in 200 AllMusic tags mapped to AV, which can be translated to quadrants. Since we considered only songs with three or more mood tags, each song is assigned to the quadrant that has the highest associated number of tags (and at least 50% of the moods are from it).

The AllMusic emotion tagging process is not fully documented, apart from apparently being made by experts [30]. Questions remain on whether these experts are considering only audio, only lyrics or a combination of both. Besides, the 30-second clips selection that represent each song in AllMusic is also undocumented. We observed several inadequate clips (e.g., containing noise such as applause, only speech, long silences from introductions). Therefore, a manual blind validation of the candidate set was conducted. Subjects were given sets of randomly distributed clips and asked to annotate them according to the perceived emotion in terms of Russell's quadrants.

The final dataset was built by removing the clips where the subjects' and AllMusic derived quadrants' annotations did not match. The dataset was rebalanced to contain exactly 225 clips and metadata per cluster, in a total of 900 song entries, which is publicly available in our site².

3.2 Standard or Baseline Audio Features

Marsyas, MIR Toolbox and PsySound3, three state-of-the-art audio frameworks typically used in MER studies, were used to extract a total of 1702 features. This high number is in part due to the computation of several statistical for the resulting time series data. To reduce this and avoid possible feature duplication across different frameworks, first we obtained the weight of each feature to the problem using ReliefF [31] feature selection algorithm. Next, we calculated the correlation between each pair of features, removing the lowest weight one for each pair with a correlation higher than 0.9. This process reduced the standard audio features set to 898 features, which was used to train baseline models. These models were then used to benchmark models trained with the baseline and novel feature sets. An analogous feature reduction procedure was also performed in the novel features set presented in Section 3.3.

3.3 Novel Audio Features

Although being used constantly in MER problems, many of the standard audio features are very low-level, extracting abstract metrics from the spectrum or directly from the audio waveform. Still, humans naturally perceive higher-level musical concepts such as rhythm, harmony, melody

¹ <http://developer.rovicorp.com/docs>

² <http://mir.dei.uc.pt/downloads.html>

lines or expressive techniques based on clues related with notes, intervals or scores. To propose novel features that related to these higher-level concepts we built on previous works to estimate musical notes and extract frequency and intensity contours. We briefly describe this initial step in the next section.

3.3.1 Estimating MIDI notes

Automatic transcription of music audio signals to scores is still and open research problem [32]. Still, we consider that using such existing algorithms, although imperfect, provide important information currently unused in MER.

To this end, we built on works by Salomon et al. [33] and Dressler [34] to estimate predominant fundamental frequencies (f_0) and saliences. This process starts by identifying the frequencies present in the signal at each point in time (sinusoid extraction), using 46.44 msec (1024 samples) frames with 5.8 msec (128 samples) hopsize (hereafter denoted *hop*). Next, the pitches in each of these moments are estimated using harmonic summation (obtaining a pitch salience function). Then, pitch contours are created from the series of consecutive pitches, representing notes or phrases. Finally, a set of rules is used to select the f_0 s that are part of the predominant melody [33]. The resulting pitch trajectories are then segmented into individual MIDI notes following the work by Paiva et al. [35].

Each of the N obtained notes, hereafter denoted as $note_i$, is characterized by: 1) the respective sequence of f_0 s (a total of L_i frames), $f_{0,j,i}, j = 1, 2, \dots, L_i$; the corresponding MIDI note numbers (for each f_0), $mid_{j,i}$; 2) the overall MIDI note value (for the entire note), $MIDI_i$; 3) the sequence of pitch saliences, $sal_{j,i}$; 4) the note duration, nd_i (sec); starting time, st_i (sec); and 5) ending time, et_i (sec). This data is used to model higher level concepts related with expressive techniques, such as vibrato.

In addition to the predominant melody, music typically contains other melodic lines produced by distinct sources. Some researchers have also proposed algorithms to multiple (also known as polyphonic) F_0 contours estimation from these constituent sources. We use Dressler's multi- F_0 approach [34] to obtain a framewise sequence of fundamental frequencies estimates to assess musical texture.

3.3.2 Musical texture features

Previous studies have verified that musical texture can influence emotion in music, either directly or in combination with tempo and mode [36]. However, as stated in Section 2, very few of the available audio features are directly related with this musical concept. Thus, we propose features to capture information related with the musical layers of a song, based on the simultaneous layers in each frame using the multiple frequency estimates described above.

Musical Layers (ML) statistics. As mentioned, various multiple F_0 s are estimated from each audio frame. Then, we define the number of layers in a frame as the number of obtained multiple F_0 s in that frame. The obtained data series, representing the number of musical layers in each instant during the clip, is then summarized using six statistics: mean (MLmean), standard deviation (MLstd), skewness (MLskw), kurtosis (MLkurt), maximum (MLmax) and minimum (MLmin) values. The same

six statistics are applied similarly to the other proposed features.

Musical Layers Distribution (MLD). Here, the number of f_0 estimates in each frame is categorized in one of four classes: i) no layers; ii) a single layer; iii) two simultaneous layers; iv) and three or more layers. The percentage of frames in each of these four classes is computed, measuring, as an example, the percentage of the song identified as having a single layer (MLD1). Similarly, we compute MLD0, MLD2 and MLD3.

Ratio of Musical Layers Transitions (RMLT). These features capture the amount of transitions (changes) from a specific musical layer sequence to another (e.g., ML1 to ML2). To this end, we count consecutive frames having distinct numbers of fundamental frequencies (f_0 s) estimated in each as a transition. The total number of these transitions is normalized by the length of the audio segment (in secs). Additionally, we also compute the length in seconds of the longest audio segment for each of the four musical layers classes.

3.3.3 Expressivity features

Expressive techniques such as vibrato, tremolo and articulation are used frequently by composers and performers, across different genres. Some studies have linked them to emotions [37]–[39], still the number of standard audio features studied that are primarily related with expressive techniques is low.

Articulation Features

Articulation relates to how specific notes are played and expressed together. To capture this, we first detect legato (i.e., connected notes played “smoothly”) and staccato (i.e., short and detached notes), as defined in Algorithm 1. Using this, we classify all the transitions between notes in the song clip and, from them, extract several metrics such as: ratio of staccato, legato and *other* transitions, longest sequence of each articulation type, etc.

ALGORITHM 1 ARTICULATION DETECTION.

1. For each pair of consecutive notes, $note_i$ and $note_{i+1}$:
 - 1.1. Compute the inter-onset interval (IOI , in sec), i.e., the interval between the onsets of the two notes, as: $IOI = st_{i+1} - st_i$.
 - 1.2. Compute the inter-note silence (INS , in sec), i.e., the duration of the silence segment between the two notes, as follows: $INS = st_{i+1} - et_i$.
 - 1.3. Calculate the ratio of INS to IOI ($INS_{to}IOI$), which indicates how long the interval between notes is, compared to the duration of $note_i$.
 - 1.4. Define the articulation between $note_i$ and $note_{i+1}$, art_i , as:
 - 1.4.1. *Legato*, if the distance between notes is less than 10 msec, i.e., $INS \leq 0.01 \Rightarrow art_i = 1$.
 - 1.4.2. *Staccato*, if the duration of $note_i$ is short (i.e., less than 500 msec) and the silence between the two notes is relatively similar to this duration, i.e., $nd_i < 0.5 \wedge 0.25 \leq INS_{to}IOI \leq 0.75 \Rightarrow art_i = 2$.
 - 1.4.3. *Other Transitions*, if none of the abovementioned two conditions was met ($art_i = 0$).

In Algorithm 1, the employed threshold values were set

experimentally. Then, we define the following features:

Staccato Ratio (SR), Legato Ratio (LR) and Other Transitions Ratio (OTR). These features indicate the ratio of each articulation type (e.g., staccato) to the total number of transitions between notes.

Staccato Notes Duration Ratio (SNDR), Legato Notes Duration Ratio (LNDR) and Other Transition Notes Duration Ratio (OTNDR) statistics. These represent statistics based on the duration of notes for each articulation type. As an example, with staccato (SNDR), the ratio of the duration of notes with staccato articulation to the sum of the duration of all notes, as in Eq. 1. For each, the 6 statistics described in Section 3.3.2 are calculated.

$$SNDR = \frac{\sum_{i=1}^{N-1} [art_i = 1] \cdot nd_i}{\sum_{i=1}^{N-1} nd_i} \quad (1)$$

Glissando Features

Glissando is another expressive articulation, which is the slide from one note to another. Normally used as an ornamentation, to add interest to a piece, may be related to specific emotions in music.

We assess glissando by analyzing the transition between two notes, as described in Algorithm 2. This transition part is saved at the beginning of the second note by the segmentation method applied (mentioned in Section 3.3.1) [35]. The second note must start with a climb or descent, of at least 100 cents, which may contain spikes and slight oscillations in frequency estimates, followed by a stable sequence.

ALGORITHM 2 GLISSANDO DETECTION.

1. For each note i :
 - 1.1. Get the list of unique MIDI note numbers, $u_{z,i}$, $z = 1, 2, \dots, U_i$, from the corresponding sequence of MIDI note numbers (for each $f0$), $midl_{j,i}$, where z denotes a distinct MIDI note number (from a total of U_i unique MIDI note numbers).
 - 1.2. If there are at least two unique MIDI note numbers:
 - 1.2.1. Find the start of the steady-state region, i.e., the index, k , of the first note in the MIDI note numbers sequence, $midl_{j,i}$, with the same value as the overall MIDI note, $MIDI_i$, i.e., $k = \min_{1 \leq j \leq L_i, midl_{j,i} = MIDI_i} j$,
 - 1.2.2. Identify the end of the glissando segment as the first index, e , before the steady-state region, i.e., $e = k - 1$.
 - 1.3. Define
 - 1.3.1. gd_i = glissando duration (sec) in note i , i.e., $gd_i = e \cdot hop$.
 - 1.3.2. gp_i = glissando presence in note i , i.e., $gp_i = 1$ if $gd_i > 0$; 0, otherwise.
 - 1.3.3. ge_i = glissando extent in note i , i.e., $ge_i = |f0_{1,i} - f0_{e,i}|$ in cents.
 - 1.3.4. gc_i = glissando coverage of note i , i.e., $gc_i = gd_i / dur_i$.
 - 1.3.5. $gdir_i$ = glissando direction of note i , i.e., $gdir_i = \text{sign}(f0_{e,i} - f0_{1,i})$.
 - 1.3.6. gs_i = glissando slope of note i , i.e., $gs_i = gdir_i \cdot ge_i / gd_i$.

Based on the output of Algorithm 2 we define:

Glissando Presence (GP). A song clip contains glissando if any of its notes has glissando, as in (2).

$$GP = \begin{cases} 1, & \text{if } \exists i \in \{1, 2, \dots, N\} : gp_i = 1 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

If $GP = 1$, we then compute the remaining glissando

features.

Glissando Extent (GE) statistics. Using the glissando extent of each note, ge_i (see Algorithm 2), we compute the 6 statistics (Section 3.3.2) for notes containing glissando.

Glissando Duration (GD) and Glissando Slope (GS) statistics. Similarly to GE, we also compute the same statistics for glissando duration, based on gd_i and slope, based on gs_i (see Algorithm 2).

Glissando Coverage (GC). For glissando coverage, we compute the global coverage, based on gc_i , using (3).

$$GC = \frac{\sum_{i=1}^N gc_i \cdot nd_i}{\sum_{i=1}^N nd_i} \quad (3)$$

Glissando Direction (GDIR). This feature indicates the global direction of the glissandos in a song, (4):

$$GDIR = \frac{\sum_{i=1}^N gp_i}{N}, \text{ when } gdir_i = 1 \quad (4)$$

Glissando to Non-Glissando Ratio (GNGR). This feature represents the ratio of the notes containing glissando to the total number of notes, as in (5):

$$GNGR = \frac{\sum_{i=1}^N gp_i}{N} \quad (5)$$

Vibrato and Tremolo Features

Vibrato and tremolo are expressive technique used in vocal and instrumental music. Vibrato consists in a steady oscillation of pitch in a note or sequence of notes. Its properties are the: 1) the velocity (rate) of pitch variation; 2) amount of pitch variation (extent); and 3) duration. It varies across music styles and emotional expression [38].

Given its possible relevance to MER, we apply the vibrato detection algorithm described in Algorithm 3, which was adapted from [40]. We then compute features such as vibrato presence, rate, coverage and extent.

ALGORITHM 3 VIBRATO DETECTION.

1. For each note i :
 - 1.1. Compute the STFT, $|F0_{w,i}|$, $w = 1, 2, \dots, W_i$, of the sequence $f0_i$, where w denotes an analysis window (from a total of W_i windows). Here, a 371.2 msec (128 samples) Blackman-Harris window was employed, with 185.6 msec (64 samples) hopsize.
 - 1.2. Look for a prominent peak, $pp_{w,i}$, in each analysis window, in the expected range for vibrato. In this work, we employ the typical range for vibrato in the human voice, i.e., [5, 8] Hz [40]. If a peak is detected, the corresponding window contains vibrato.
 - 1.3. Define:
 - 1.3.1. vp_i = vibrato presence in note i , i.e., $vp_i = 1$ if $\exists pp_{w,i}$; $vp_i = 0$, otherwise.
 - 1.3.2. WV_i = number of windows containing vibrato in note i .
 - 1.3.3. vc_i = vibrato coverage of note i , i.e., $vc_i = WV_i / W_i$ (ratio of windows with vibrato to the total number of windows).
 - 1.3.4. vd_i = vibrato duration of note i (sec), i.e., $vd_i = vc_i \cdot d_i$.
 - 1.3.5. $\text{freq}(pp_{w,i})$ = frequency of the prominent peak $pp_{w,i}$ (i.e., vibrato frequency, in Hz).
 - 1.3.6. vr_i = vibrato rate of note i (in Hz), i.e., $vr_i = \sum_{w=1}^{WV_i} \text{freq}(pp_{w,i}) / WV_i$ (average vibrato frequency).
 - 1.3.7. $|pp_{w,i}|$ = magnitude of the prominent peak $pp_{w,i}$ (in cents).
 - 1.3.8. ve_i = vibrato extent of note i , i.e., $ve_i = \sum_{w=1}^{WV_i} |pp_{w,i}| / WV_i$ (average amplitude of vibrato).

Then, we define the following features.

Vibrato Presence (VP). A song clip contains vibrato if any of its notes have vibrato, similarly to (2).

Vibrato Rate (VR) statistics. Based on the vibrato rate value of each note, vr_i (see Algorithm 3), we compute 6 statistics described in Section 3.3.2 (e.g., the vibrato rate weighted mean of all notes with vibrato as in Eq. 6).

$$VRmean = \frac{\sum_{i=1}^N vr_i \cdot vc_i \cdot nd_i}{\sum_{i=1}^N vc_i \cdot nd_i} \quad (6)$$

Vibrato Extent (VE) and Vibrato Duration (VD) statistics. Similarly to VR, these features represent the same statistics for vibrato extent, based on ve_i and vibrato duration, based on vd_i (see Algorithm 3).

Vibrato Notes Base Frequency (VNBF) statistics. As with VR features, we compute the same statistics for the base frequency (in cents) of all notes containing vibrato.

Vibrato Coverage (VC). This represents the global vibrato coverage in a song, based on vc_i , similarly to (3).

High-Frequency Vibrato Coverage (HFVC). Here, the VC is computed only for notes over C4 (261.6 Hz), which is the lower limit of the soprano's vocal range [41].

Vibrato to Non-Vibrato Ratio (VNVR). This feature is defined as the ratio of the notes containing vibrato to the total number of notes, similarly to (5).

An approach similar to vibrato was applied to compute tremolo features. Tremolo can be described as a trembling effect, to a certain degree similar to vibrato but regarding variation of amplitude. Here, instead of using the f0 sequences, the sequence of pitch saliences of each note is used to assess variations in intensity or amplitude. Due to the lack of research regarding tremolo range, we decided to use vibrato range (i.e., 5-8Hz).

3.4 Emotion Classification

Given the high number of features, ReliefF feature selection algorithms [31] were used to rank the better suited ones emotion classification. This algorithm outputs feature weights in the range of -1 to 1, with higher values indicating attributes more suited to the problem. This, in conjunction with the strategy described in Section 3.2, were used to reduce and merge baseline and novel features sets.

For classification we selected Support Vector Machines (SVM) [42] as the machine learning technique, since it has performed well in previous MER studies. SVM parameters were tuned with grid search and a Gaussian kernel (RBF) was selected based on preliminary tests. The experiments were validated with 20 repetitions of 10-fold cross validation [43], where we report the average (macro weighted) results.

4. RESULTS AND DISCUSSION

In this section we discuss the results of our classification tests. Our main objective was to assess the relevance of existing audio features to MER and understand if and how our novel proposed ones improve the current scenario. With this in mind, we start by testing the existing baseline (standard) features only, followed by tests using the com-

bination of baseline and novel, to assess if the obtained results improve and if the differences are statistically significant.

A summary of the classification results is shown in Table 1. The baseline feature set obtained its best result, of 71.7% F1-score, with an extremely high number of features (800). Considering a more reasonable number of features, up to the best 100 according to ReliefF, the best model used the top70, and attained 67.5%. Next, including novel features (with the baseline) increased the best result to 76.0% F1-score using the best 100 features, a considerably lower number (100 instead of 800). This difference is statistically significant (at $p < 0.01$, paired T-test). Interestingly, we observed decreasing results with models using higher number of features, indicating that those extra features might not be relevant but introducing noise.

Classifier	Feature set	# feats.	F1-Score
SVM	baseline	70	67.5% ± 0.05
SVM	baseline	100	67.4% ± 0.05
SVM	baseline	800	71.7% ± 0.05
SVM	baseline+novel	70	74.0% ± 0.05
SVM	baseline+novel	100	76.0% ± 0.05
SVM	baseline+novel	800	73.5% ± 0.04

Table 1. Results of the classification by quadrants.

Of the 100 features used in the best result, 29 are novel, which demonstrates the relevance of adding novel features to MER. Of these, 8 are related with texture, such as the number of musical layers (*MLmean*), while the remaining 21 are expressive techniques such as tremolo, glissando and especially vibrato (12). The remaining 71 baseline features are mainly tone color related (50), with the few others capturing dynamics, harmony, rhythm and melody.

Further analysis to the results per individual quadrant, presented in Table 2, gives us a deeper understanding about which emotions are harder to classify and where the new features were more significant. According to it, Q1 and Q2 obtained a higher result compared to the remaining. This seems to indicate that emotions in songs with higher arousal are easier to differentiate. Also, Q2 result is significantly higher, indicating that it might be markedly distinct from the remaining, explained by the fact that several excerpts from Q2 belong to genres such as punk, hard-core or heavy-metal, which have very distinctive, noise-like, acoustic features. This goes in the same direction as the results obtained in previous studies [44].

Quads	baseline			novel		
	Prec.	Recall	F1-Score	Prec.	Recall	F1-Score
Q1	62.6%	73.4%	67.6%	72.9%	81.9%	77.2%
Q2	82.3%	79.6%	80.9%	88.9%	82.7%	85.7%
Q3	61.3%	57.5%	59.3%	73.0%	69.2%	71.1%
Q4	62.8%	57.9%	60.2%	68.5%	68.6%	68.5%

Table 2. Results per quadrant using 100 features.

Several factors can be thought to explain the lower results in Q3 and Q4 (average of -11.7%). First, a higher number of ambiguous songs exist in these quadrants, containing unclear or contrasting emotions. This is supported

by the low agreement (45.3%) between the subject's and the original AllMusic annotations during the annotation process. In addition, the two quadrants contain songs which share similar musical characteristics, sometimes with each characteristic related to contrasting emotional cues (e.g., a happy melody and a sad voice or lyric). This agrees with the conclusions presented in [45]. As a final point, these similarities may explain why the subjects reported having more difficulty distinguishing valence for songs with low arousal.

The addition of novel features improved the results by 8.6% when considering the top 100 features' results. Novel features seemed more relevant to Q3, with the most significant improvement (by 11.8%), which was before the worst performing quadrant, followed by Q1 (9.6%). On the opposite end, Q2 was already the best performing with baseline features and thus is lower improvement (4.8%).

In addition to assessing the importance of baseline and novel features for quadrants classification, where we identified 29 novel features in the best 100, we also studied the best features to discriminate each specific quadrant from the others. This was done by analyzing specific feature rankings, e.g., the ranking of features that are best to separate Q1 songs from non-Q1 songs (a set containing Q2, Q3 and Q4 annotated as non-Q1). As expected based on former tests, tone color is the most represented concept in the list of the 10 best features for each of the four quadrants. The reason is in part due to being overrepresented in original feature set, while relevant features from other concepts may be missing.

Of the four quadrants, Q2 and Q4 seem to have the most suited features to distinguish them (e.g., features to identify a clip as Q2 vs non-Q2), according to the obtained ReliefF weights. This was confirmed experimentally, where we observed that 10 features or less was enough to obtain 95% of the max score in binary problems for Q2 and Q4, while the top 30 and 20 features, for Q1 and Q3 respectively, were needed to attain the same goal.

Regarding the first quadrant, some of the novel features related with musical texture information were shown to be very relevant. As an example, in the top features, 3 are novel, capturing information related with the number of musical layers and the transitions between different texture types, together with 3 rhythmic features related with events density and fluctuation. Q1 represents happy emotions, which are typically energetic. Associated songs tend to be high in energy and have appealing ("catchy") rhythm. Thus, features related with rhythm, together with texture and tone color (mostly energy metrics) support this. Nevertheless, as stated before the weight of these features to Q1 is low when compared with the top features of other quadrants.

For Q2 the features identified as most suited are related with tone color, such as: roughness - capturing the dissonance in the song; rolloff - measuring the amount of high frequency; MFCCs - total energy in the signal; and spectral flatness measure - indicating how noise-like the sound is. Other important features are related with dynamics, such as tonal dissonance. As for novel features, expressive techniques ones, mainly vibrato, which makes 43% of the top 30 features. Some research supports this association of vibrato and negative energetic emotions such as anger

[46]. Generally, the associations found seem reasonable. After all, Q2 is made of tense, aggressive music, and musical characteristics like sensory dissonance, high energy, and complexity are usually present.

Apart from tone color features (extracting energy information), quadrant 3 is also identified higher level features from concepts such as musical texture, dynamics and harmony and expressive techniques. Namely, the number of musical layers, spectral dissonance, inharmonicity, and tremolos. As for quadrant 4, in addition to tone color features related to spectrum (such as skewness or entropy) or measures of how noise-like is the spectrum (spectral flatness), the remaining are again related with dynamics (dissonance) and harmony, as well as some vibrato metrics. More and better features are needed to better understand and discriminate Q3 from Q4. From our tests, songs from both quadrants share some common musical characteristics such as lower tempo, less musical layers and energy, use of glissandos and other expressive techniques.

5. CONCLUSIONS AND FUTURE WORK

We studied the relevance of musical audio features, proposing novel features that complement the existing ones. To this end, the features available in known frameworks were studied and classified in one of eight musical concepts - dynamics, expressive techniques, harmony, melody, musical form, musical texture, rhythm and tone color. Concepts such as musical form, musical texture and expressive techniques were identified as the ones most lacking available audio extractors. Based on this, we proposed novel audio features to mitigate the identified gaps and break the current glass ceiling. Namely, related with expressive techniques, capturing information related with vibrato, tremolo, glissando and articulation. Also, related with musical texture, capturing statistics regarding the musical layers of a musical piece.

Since no public available dataset fulfilled our needs, a new dataset with 900 clips and metadata (e.g., title, artist, genres and moods), annotated according to the Russell's emotion model quadrants was built semi-automatically, used in our tests and is available to other researchers.

Our experimental tests demonstrated that the novel proposed features are relevant and improve MER classification. As an example, using a similar number of features (100), adding our novel proposed features increased the results by 8.6% (to 76.0%), when compared to the baseline. This result was obtained using 29 novel features and 71 baseline, which demonstrates the relevance of this work.

Additional experiments were conducted to uncovered and better understand relations between audio features, musical concepts and specific emotions (quadrants).

In the future, we would like to study multi-modal approaches and the relation between the voice signal and lyrics, as well as testing the features influence in finer grained categorical and dimensional emotion models. Also, other features (e.g. related with musical form), are still to be developed. Moreover, we would like to derive a more understandable set of knowledge (e.g. rules) of how musical features influence emotion, something that lacks when black-box classification methods such as SVMs are employed.

6. ACKNOWLEDGMENT

This work was supported by the MOODetector project (PTDC/EIA-EIA/102185/2008), financed by the Fundação para Ciência e a Tecnologia (FCT) and Programa Operacional Temático Factores de Competitividade (COMPETE) – Portugal, as well as the PhD Scholarship SFRH/BD/91523/2012, funded by the Fundação para Ciência e a Tecnologia (FCT), Programa Operacional Potencial Humano (POPH) and Fundo Social Europeu (FSE).

7. REFERENCES

- [1] A. Pannese, M.-A. Rappaz, and D. Grandjean, “Metaphor and music emotion: Ancient views and future directions,” *Conscious. Cogn.*, vol. 44, pp. 61–71, Aug. 2016.
- [2] Y. Feng, Y. Zhuang, and Y. Pan, “Popular Music Retrieval by Detecting Mood,” *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, vol. 2, no. 2, pp. 375–376, 2003.
- [3] C. Laurier and P. Herrera, “Audio Music Mood Classification Using Support Vector Machine,” in *Proc. of the 8th Int. Society for Music Information Retrieval Conf. (ISMIR 2007)*, 2007, pp. 2–4.
- [4] Y.-H. Yang, Y.-C. Lin, Y.-F. Su, and H. H. Chen, “A Regression Approach to Music Emotion Recognition,” *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 16, no. 2, pp. 448–457, Feb. 2008.
- [5] L. Lu, D. Liu, and H.-J. Zhang, “Automatic Mood Detection and Tracking of Music Audio Signals,” *IEEE Trans. Audio, Speech Lang. Process.*, vol. 14, no. 1, pp. 5–18, Jan. 2006.
- [6] R. Panda and R. P. Paiva, “Using Support Vector Machines for Automatic Mood Tracking in Audio Music,” in *130th Audio Engineering Society Convention*, 2011, vol. 1.
- [7] A. Flexer, D. Schnitzer, M. Gasser, and G. Widmer, “Playlist Generation Using Start and End Songs,” in *Proc. of the 9th Int. Society of Music Information Retrieval Conf. (ISMIR 2008)*, 2008, pp. 173–178.
- [8] O. C. Meyers, “A Mood-Based Music Classification and Exploration System,” MIT Press, 2007.
- [9] R. Malheiro, R. Panda, P. Gomes, and R. P. Paiva, “Emotionally-Relevant Features for Classification and Regression of Music Lyrics,” *IEEE Trans. Affect. Comput.*, pp. 1–1, 2016.
- [10] X. Hu and J. S. Downie, “When lyrics outperform audio for music mood classification: a feature analysis,” in *Proc. of the 11th Int. Society for Music Information Retrieval Conf. (ISMIR 2010)*, 2010, pp. 619–624.
- [11] Y. Yang, Y. Lin, H. Cheng, I. Liao, Y. Ho, and H. H. Chen, “Toward multi-modal music emotion classification,” in *Pacific-Rim Conference on Multimedia*, 2008, vol. 5353, pp. 70–79.
- [12] R. Panda, R. Malheiro, B. Rocha, A. Oliveira, and R. P. Paiva, “Multi-Modal Music Emotion Recognition: A New Dataset, Methodology and Comparative Analysis,” in *10th International Symposium on Computer Music Multidisciplinary Research – CMMR’2013*, 2013, pp. 570–582.
- [13] Ò. Celma, P. Herrera, and X. Serra, “Bridging the Music Semantic Gap,” in *Workshop on Mastering the Gap: From Information Extraction to Semantic Representation*, 2006, vol. 187, no. 2, pp. 177–190.
- [14] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck, and D. Turnbull, “Music Emotion Recognition: A State of the Art Review,” in *Proc. of the 11th Int. Society for Music Information Retrieval Conf. (ISMIR 2010)*, 2010, pp. 255–266.
- [15] X. Yang, Y. Dong, and J. Li, “Review of data features-based music emotion recognition methods,” *Multimed. Syst.*, pp. 1–25, Aug. 2017.
- [16] S. B. Davis and P. Mermelstein, “Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*. 1980.
- [17] J. A. Russell, “A circumplex model of affect,” *J. Pers. Soc. Psychol.*, vol. 39, no. 6, pp. 1161–1178, 1980.
- [18] K. Hevner, “Experimental Studies of the Elements of Expression in Music,” *Am. J. Psychol.*, vol. 48, no. 2, pp. 246–268, 1936.
- [19] M. Malik, S. Adavanne, K. Drossos, T. Virtanen, D. Ticha, and R. Jarina, “Stacked Convolutional and Recurrent Neural Networks for Music Emotion Recognition,” in *Proc. of the 14th Sound & Music Computing Conference*, 2017, pp. 208–213.
- [20] A. Aljanaki, Y.-H. Yang, and M. Soleymani, “Developing a benchmark for emotional analysis of music,” *PLoS One*, vol. 12, no. 3, Mar. 2017.

- [21] C. Laurier, O. Lartillot, T. Eerola, and P. Toivainen, "Exploring relationships between audio features and emotion in music," in *Proc. of the 7th Triennial Conf. of European Society for the Cognitive Sciences of Music*, 2009, vol. 3, pp. 260–264.
- [22] A. Friberg, "Digital Audio Emotions - An Overview of Computer Analysis and Synthesis of Emotional Expression in Music," in *Proc. of the 11th Int. Conf. on Digital Audio Effects (DAFx)*, 2008, pp. 1–6.
- [23] G. Tzanetakis and P. Cook, "MARSYAS: a framework for audio analysis," *Organised Sound*, vol. 4, no. 3, pp. 169–175, 2000.
- [24] O. Lartillot and P. Toivainen, "A Matlab Toolbox for Musical Feature Extraction from Audio," in *Proc. of the 10th Int. Conf. on Digital Audio Effects (DAFx)*, 2007, pp. 237–244.
- [25] D. Cabrera, S. Ferguson, and E. Schubert, "'PsySound3': Software for Acoustical and Psychoacoustical Analysis of Sound Recordings," in *Proc. of the 13th Int. Conf. on Auditory Display (ICAD2007)*, 2007, pp. 356–363.
- [26] C. Laurier, "Automatic Classification of Musical Mood by Content-Based Analysis," Universitat Pompeu Fabra, 2011.
- [27] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset," in *Proc. of the 12th Int. Society for Music Information Retrieval Conf. (ISMIR 2011)*, 2011, pp. 591–596.
- [28] A. B. Warriner, V. Kuperman, and M. Brysbaert, "Norms of valence, arousal, and dominance for 13,915 English lemmas," *Behav. Res. Methods*, vol. 45, no. 4, pp. 1191–1207, Dec. 2013.
- [29] M. M. Bradley and P. J. Lang, "Affective Norms for English Words (ANEW): Instruction Manual and Affective Ratings," *Psychology*, vol. Technical, no. C-1, p. 0, 1999.
- [30] X. Hu and J. S. Downie, "Exploring Mood Metadata: Relationships with Genre, Artist and Usage Metadata," in *Proc. of the 8th Int. Society for Music Information Retrieval Conf. (ISMIR 2007)*, 2007, pp. 67–72.
- [31] M. Robnik-Šikonja and I. Kononenko, "Theoretical and Empirical Analysis of ReliefF and RReliefF," *Mach. Learn.*, vol. 53, no. 1–2, pp. 23–69, 2003.
- [32] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic music transcription: challenges and future directions," *J. Intell. Inf. Syst.*, vol. 41, no. 3, pp. 407–434, 2013.
- [33] J. Salamon and E. Gómez, "Melody Extraction From Polyphonic Music Signals Using Pitch Contour Characteristics," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 20, no. 6, pp. 1759–1770, 2012.
- [34] K. Dressler, "Automatic Transcription of the Melody from Polyphonic Music," Ilmenau University of Technology, 2016.
- [35] R. P. Paiva, T. Mendes, and A. Cardoso, "Melody Detection in Polyphonic Musical Signals: Exploiting Perceptual Rules, Note Salience, and Melodic Smoothness," *Comput. Music J.*, vol. 30, no. 4, pp. 80–98, Dec. 2006.
- [36] G. D. Webster and C. G. Weir, "Emotional Responses to Music: Interactive Effects of Mode, Texture, and Tempo," *Motiv. Emot.*, vol. 29, no. 1, pp. 19–39, Mar. 2005.
- [37] P. Gomez and B. Danuser, "Relationships between musical structure and psychophysiological measures of emotion," *Emotion*, vol. 7, no. 2, pp. 377–387, May 2007.
- [38] C. Dromey, S. O. Holmes, J. A. Hopkin, and K. Tanner, "The Effects of Emotional Expression on Vibrato," *J. Voice*, vol. 29, no. 2, pp. 170–181, Mar. 2015.
- [39] T. Eerola, A. Friberg, and R. Bresin, "Emotional expression in music: contribution, linearity, and additivity of primary musical cues," *Front. Psychol.*, vol. 4, p. 487, 2013.
- [40] J. Salamon, B. Rocha, and E. Gómez, "Musical genre classification using melody features extracted from polyphonic music signals," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 81–84.
- [41] A. Peckham, J. Crossen, T. Gebhardt, and D. Shrewsbury, *The Contemporary Singer: Elements of Vocal Technique*. Berklee Press, 2010.
- [42] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.
- [43] R. O. Duda, P. E. (Peter E. Hart, and D. G. Stork, *Pattern classification*. Wiley, 2000.

- [44] G. R. Shafron and M. P. Karno, "Heavy metal music and emotional dysphoria among listeners.," *Psychol. Pop. Media Cult.*, vol. 2, no. 2, pp. 74–85, 2013.
- [45] Y. Hong, C.-J. Chau, and A. Horner, "An Analysis of Low-Arousal Piano Music Ratings to Uncover What Makes Calm and Sad Music So Difficult to Distinguish in Music Emotion Recognition," *J. Audio Eng. Soc.*, vol. 65, no. 4, 2017.
- [46] K. R. Scherer, J. Sundberg, L. Tamarit, and G. L. Salomão, "Comparing the acoustic expression of emotion in the speaking and the singing voice," *Comput. Speech Lang.*, vol. 29, no. 1, pp. 218–235, Jan. 2015.

SHARED GENERATIVE REPRESENTATION OF AUDITORY CONCEPTS AND EEG TO RECONSTRUCT PERCEIVED AND IMAGINED MUSIC

André Ofner

Sebastian Stober

Research Focus Cognitive Sciences, University of Potsdam, Germany

{ofner, sstober}@uni-potsdam.de

ABSTRACT

Retrieving music information from brain activity is a challenging and still largely unexplored research problem. In this paper we investigate the possibility to reconstruct perceived and imagined musical stimuli from electroencephalography (EEG) recordings based on two datasets. One dataset contains multi-channel EEG of subjects listening to and imagining rhythmical patterns presented both as sine wave tones and short looped spoken utterances. These utterances leverage the well-known speech-to-song illusory transformation which results in very catchy and easy to reproduce motifs. A second dataset provides EEG recordings for the perception of 10 full length songs. Using a multi-view deep generative model we demonstrate the feasibility of learning a shared latent representation of brain activity and auditory concepts, such as rhythmical motifs appearing across different instrumentations. Introspection of the model trained on the rhythm dataset reveals disentangled rhythmical and timbral features within and across subjects. The model allows continuous interpolation between representations of different observed variants of the presented stimuli. By decoding the learned embeddings we were able to reconstruct both perceived and imagined music. Stimulus complexity and the choice of training data shows strong effect on the reconstruction quality.

1. INTRODUCTION

Studying the human brain's response to music gained a lot of attention in recent years. Many studies in the field rely on electroencephalography (EEG) recordings, as they provide better temporal resolution than other techniques, such as functional magnetic resonance imaging (fMRI). Previous research suggests that a listener's brain response is modulated in correlation to the perceived auditory stimuli on many different levels and that these modulations can be detected within EEG. One of these effects is the correlation between the frequency and magnitude of neural oscillation patterns, which are modulated by accents and rhythmical patterns in music [3, 20, 21]. Other studies indicate that tracking auditory attention towards a specific sound source in EEG recordings is possible [1, 30].

EEG data has been used to research event-related potentials (ERPs) as a repeatable and distinguishable response to aspects

of perceived music. The characteristic brain activity patterns underlying ERPs can be specific, for example, to the structure of musical events, such as note onsets or rhythm and pitch patterns [19, 24]. Other ERPs are related to the timbre of sound and can be modulated even by differences within timbre, such as changes in harmonics [17, 25]. While many ERP components show similar activation across subjects, studies suggest that some are caused by more fine-grained aspects of music, especially within trained musicians [25]. These brain activity patterns extend over the temporal, spatial and frequency domain of the EEG signal.

Motivated by the existence of such features, EEG recordings have been used in several music information retrieval studies based on EEG, such as perceived rhythm or tempo classification [28]. First attempts have been made to reconstruct the loudness envelope of perceived and imagined musical stimuli, but with unsatisfying accuracy [22, 26, 27]. Some of these studies use deep neural networks for classification and regression and the achieved results hint at their usefulness in exploring the complex brain signal. However, the power of employed networks is restricted by size and their general application exclusively to EEG signal denoising or classification. Outside from research on music cognition, recent studies have shown the possibility to use generative models to reconstruct perceived visual stimuli both from fMRI and EEG recordings [4, 10]. Generative models learn to encode a meaningful internal latent representation of a given signal. In addition, they contain a decoding part to either reconstruct the input or another signal that is extractable from the internal latent variable. A recent study has demonstrated the possibility to learn such shared latent embeddings for EEG recordings of music perception and use them as a continuous semantic space representation of the audio [23].

This suggests that a more elaborate generative model could learn a shared encoding of music and brain signals, leading to a conjoint representation of those auditory concepts that are perceived and processed by the brain. As previous research suggests, these concepts span a spectrum of complexity, starting on the level of the subject-specific manifestation and meaning of specific ERP responses to high-level semantic or emotional meaning of music. Therefore, they provide the necessary information to reconstruct the musical stimuli as they are perceived or imagined. Based on this motivation, we propose a generative multi-view model that makes use of deep neural networks to encode and decode spatio-temporal brain signal using a latent embedding. This embedding is simultaneously used to reconstruct and classify music presented and imagined during EEG recording. In this paper we introduce our view



© André Ofner, Sebastian Stober. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** André Ofner, Sebastian Stober. "Shared generative representation of auditory concepts and EEG to reconstruct perceived and imagined music", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

on auditory concepts and the suggested method. We describe two datasets of EEG recordings during music perception and imagination that are used for training and evaluation. Furthermore, we perform model introspection to demonstrate the possibility of interpolating between musically meaningful points within the learned latent space. Finally, we suggest possible ways to extend the framework to include multi-modal processing and learning high level musical concepts.

2. AUDITORY CONCEPTS

Our approach relies on three assumptions for auditory concepts:

1. Coupled auditory and conceptual processing
2. Shared neural representation of music perception and imagination
3. Hierarchical structure of music

Firstly, we assume that there is a tight coupling between auditory and conceptual processing [12]. Several studies suggest that auditory stimuli are processed in a conceptual system that is shared with other modalities, such as visual perception [31]. Furthermore, music processing is based on concepts inherent to the auditory stimuli as well as on external factors, such as visual and social environment or musical training [7]. Secondly, following the ideas of embodied cognition, we assume that the human conceptual system is essentially grounded in perception and that through its interplay with action and cognitive states, music perception at least partially shares conceptual and neural representation with musical imagination [11]. Previous research suggests that auditory concept formation can be traced back to specific ERPs and that the magnitude of some ERP component can be controlled by the presence of an auditory concept in the listeners mind [29]. Thirdly, we follow the idea that music is essentially hierarchical in structure and that auditory concepts equivalently exist on a spectrum of abstraction levels, reflecting and augmenting this structure. They can range from concepts related to single sounds or rhythm to concepts within the emotional or aesthetic processing of music. Together with the previous two assumptions this means that basic elements of perceptual musical processing, such as ERPs related to note onset expectancy, are influenced by their integration into conceptual processing. Music cognition and concept formation can be highly subjective, stimulus-driven as well as context-dependent, e.g. on visual and social aspects of a performance [18]. For these reasons, we hypothesize that a simultaneous retrieval of auditory concepts from multiple sources aids the reconstruction of the processed stimuli while further deepening our understanding of music cognition.

3. RELATED WORK

Various approaches exist to learning a shared embedding from two or more datasets. One method is Canonical Correlation Analysis (CCA) [8]. CCA is non-probabilistic and enables the extraction of linear components to optimize the correlations between two multivariate datasets. CCA in combination with convolutional neural networks has recently been used by Raposo

et al. to learn a shared semantic space between audio and EEG signal [23]. Based on CCA, Fujiwara et al. have introduced Bayesian Canonical Correlation Analysis (BCCA), a probabilistic interpretation of CCA [5]. However, BCCA still contains linear observation models, while EEG data is very complex and noisy and requires non-linear computation. To surpass this limitation, Deep Canonically Correlated Autoencoders (DCCAes) were proposed by Wang et al. [32]. DCCAes maximize the correlation between the latent embeddings of two separate autoencoders, but do not enable cross-reconstruction between their inputs. While this problem is solved by correlational neural networks (CorrNets), the unregularized latent embeddings of both DCCAe and CorrNet are prone to overfitting, especially in combination with the representational power of non-linear observation models [2]. For these reasons, we follow the suggestion of Wang et al. to use a deep, generative and probabilistic latent variable interpretation of CCA called Deep Variational Canonical Correlation Analysis (VCCA) [32]. A similar approach tailored specifically to a missing view reconstruction for visual stimuli in fMRI data has successfully been demonstrated recently [4]. Here, we show that we can derive a general multi-view generative model capable of joint EEG and stimulus processing that allows multi-modal learning from physiological data as well as directly from the stimuli. To our knowledge, no comparable framework for EEG-based audio stimulus reconstruction or for shared auditory concept learning exists.

4. DATASETS AND PREPROCESSING

We use two datasets, the OpenMIIR speech and the Naturalistic Music EEG Dataset - Tempo (NMED-T) dataset. They are similar in experimental setup but differ in focus and size.

4.1 OpenMIIR speech dataset

One dataset contains EEG of subjects listening to and imagining four rhythmical patterns presented both as sine wave tones and short looped spoken utterances. It stems from the Open Music Imagery Information Retrieval (OpenMIIR) initiative [28] and features four different catchy and easy to reproduce motifs superimposed on a constant metronome click. We refer to it as "OpenMIIR speech dataset". The trials are annotated for containing either speech or sine wave tones and can be used to train and evaluate model performance for the perception and imagination of the same rhythmical trials within two timbres. The metronome clicks serve as cues that are present during perception as well as imagination. The main intention behind this dataset is to reduce stimulus complexity as far as possible while still retaining enough musical structure for building and evaluating models. This dataset contains data from seven subjects with normal hearing and no history of brain injury. It was recorded with 64 EEG channels, horizontal and vertical Electrooculography (EOG) channels sampled at 512 Hz. All perception stimuli have equal tempo and duration of 12 s. Presentation was done in randomized order after 2 s of metronome clicks. They were immediately followed by another 12 s of metronome cues. Participants were asked to imagine the perceived stimulus directly after presentation using these subsequent cue clicks. The concatenated

perception-imagination trials sum up to 26 s of recorded EEG data for each trial. As each trial was presented 6 times, this sums up to a total of 96 presented trials. In total, the dataset contains about 2500 s (42 min) of EEG recordings per subject. We performed common-practice preprocessing steps using the MNE-python toolbox by Gramfort et al. including manual bad channel removal and interpolation after visual inspection [6]. All EEG data was bandpass filtered between 0.5 and 50 Hz. Extended Infomax Independent Component Analysis (ICA) was used to remove EEG artifacts using the EOG signal.

4.2 NMED-T dataset

The NMED-T dataset provides EEG recordings for the perception of 10 naturalistic full length songs. The songs are in Western musical tradition, have durations between 4:30 and 5:00 min in length and contain vocals. They are real-world musical works with pronounced rhythmical properties. 125 channel EEG at 1 kHz sampling rate was recorded for all of the 20 subjects with normal hearing and no history of brain injury. We used the preprocessed version of the dataset, which features EEG down-sampled to 125 Hz and bandpass filtered between 0.3 and 50 Hz. Ocular and cardiac artifacts were removed using the additional EOG channels with ICA after manual bad channel removal. A more detailed description of the preprocessed dataset can be found in [15].

Subjects in both experiments were not required to have musical training, nor did they execute a particular task during listening or imagination. All EEG channels were normalized to zero mean and range [-1, 1]. For training, EEG data was split into excerpts of 1 s length, resulting in 512 samples (OpenMIIR) and 125 samples (NMED-T) length.

We computed Mel spectrograms of audio targets at full sample-rate of 44100 Hz using the librosa library [16] with 64 frequency bands between 0 and 2000 Hz, FFT window size of 2048 and hop length of 1024. Furthermore, we generated loudness envelopes for each stimulus using Hilbert transform of the scipy library at the full sample rate [9]. We then down-sampled the Mel spectrograms and loudness envelopes to the sample rates of the EEG (512 Hz for OpenMIIR and 125 Hz for NMED-T) before splitting into excerpts of 1 s length.

5. LEARNING SHARED REPRESENTATIONS OF AUDIO AND BRAIN SIGNAL

We propose an adaptation of VCCA as proposed by Wang et al. [32] to perform multi-view learning on audio and EEG signal by defining EEG and audio to be two views that can be generated independently from a shared latent embedding z :

$$p(\text{audio}, \text{eeg}, z) = p(z)p(\text{audio}|z)p(\text{eeg}|z). \quad (1)$$

As we are essentially interested in the auditory information within EEG signal, we formulate a default model with a single encoder, which processes EEG. Here, z is a learnable space of auditory concepts which are contained implicitly both in the audio and the EEG signal and which generate significant parts of both views. Following the VCCA principle, we project both audio and EEG signal into the shared space z . By declaring the prior $p(z)$, $p(\text{audio} | \text{eeg})$, and $p(\text{eeg} | z)$

to be Gaussian, we ensure that the projections $E[z | \text{audio}]$ and $E[z | \text{eeg}]$ of the maximum likelihood solution are in the same space as the projections through CCA. As we deal with the reconstruction of complex EEG data, we parametrize the mean of $p_{\Theta}(\text{eeg} | z)$ with deep neural networks (DNNs) and apply the same procedure for the mean of $p_{\Theta}(\text{audio} | z)$. The approximate posterior $q_{\phi}(z | \text{eeg})$ is optimized by a third DNN. Training the VCCA model is done in analogy to Variational Autoencoders (VAEs) with variational inference by sampling from $q_{\phi}(z | \text{eeg})$. Optimizing the lower bound of the log likelihood $L(\text{eeg}, \text{audio}; \theta, \phi)$ with stochastic backpropagation is done by optimizing the reconstruction loss of audio and EEG decoder and the Kullback-Leibler (KL) divergence between the learned $q_{\phi}(z | \text{eeg})$ and $p(z)$ using the reparameterization trick [14].

5.1 Multimodal data and additional views

This model can be extended to arbitrary amount of decoders to reconstruct multiple views, as long as they are dependent mainly of a shared latent variable. Here, we use several decoders to reconstruct different aspects of the audio signal: Mel spectrograms of the audio stimuli, their loudness envelope as well as an additional decoder to classify the trial types. Based on our retrieval intention, here we focus on the learned embedding and the reconstructed Mel spectrograms. We use the remaining decoders to enhance the training quality. Similarly, we can add additional encoders, if they represent data based on the latent variable, by making use of additional private latent variables introduced with the VCCA model. They store only the view-specific aspects of additional input, e.g. from other biological modalities, such as fMRI, audio or EEG signal during imagination. Figure 1 shows an example of the modified VCCA architecture with one EEG decoder and two audio decoders. Here, we test the model with a single EEG encoder and multiple decoders.

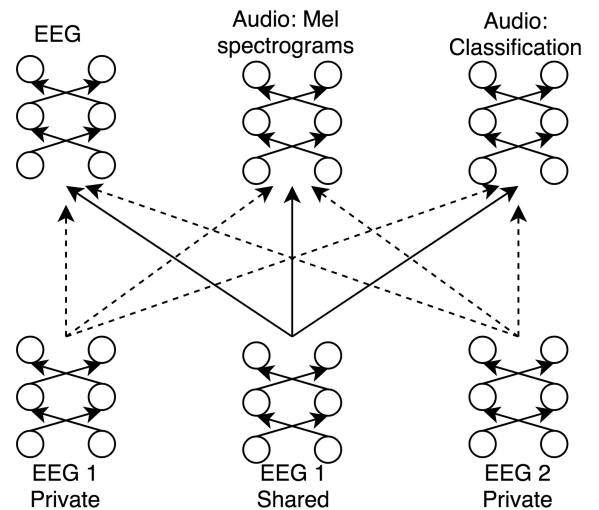


Figure 1. VCCA architecture for shared auditory concept and EEG representation learning. Latent variables parametrized by optional private encoders are indicated with dashed lines.

5.2 EEG encoder architectures

Both NMED-T and OpenMIIR speech EEG encoders featured 4 convolutional layers with filter numbers linearly ascending from 64 to 512 per layer. Convolution was performed on two dimensional inputs. Each column of the input represented the same linear concatenation of EEG channels for a single sample within the inputs of 1 s length. This resulted in inputs of size 512×64 for the OpenMIIR speech and 125×125 channels for the NMED-T inputs. The kernel size was set to $[2 \times 2]$ for all layers. Here and for all further kernel dimensions, we define the first index to be within the channel domain (or frequency for spectrograms) and the second within the temporal domain. Each convolutional layer was followed by 30 % dropout.

5.3 EEG and audio decoder architectures

We used similar EEG decoder architectures for both datasets. The OpenMIIR speech EEG decoder featured 6 hidden deconvolution layers with three layers of 16 and another three layers of 32 filters. The kernel size was set uniformly to $[2 \times 16]$ with stride 2 except for a $[2 \times 1]$ kernel in the third layer with stride 1. A final dense output layer consisted of 512×64 units. The decoder for the NMED-T dataset followed the same deconvolution architecture, except for kernels with dimension of $[4 \times 16]$ and $[4 \times 1]$ instead of $[2 \times 16]$ and $[2 \times 1]$. A final dense layer consisted of 125×125 units. Both OpenMIIR speech and NMED-T decoders for Mel spectrograms consisted of four layers: Two deconvolution layers of 32 filters and two layers with 64 filters. As the length of Mel spectrograms mirrors those of the EEG excerpts, but in combination with a frequency resolution of 64 bins, the final dense layer featured 512×64 and 125×64 units respectively. The kernel dimensions were set to $[4 \times 8]$ uniformly, except for the fourth deconvolution layer of the OpenMIIR speech decoder, with a $[2 \times 8]$ kernel. The decoder for loudness envelope reconstruction consisted of a bidirectional LSTM layer with 128 hidden units, followed by a dense layer of size equal to the length of the audio excerpt. Finally, the decoder used for classification of the OpenMIIR speech dataset consisted of two hidden dense layers with 32 filters and a dense output layer of 1 unit. All internal units used Rectified Linear Unit (ReLU) activations, all output units had sigmoid activation. The size of the latent embedding was 128 units.

5.4 VCCA training and prediction

The extended VCCA model was trained both intra-subject and cross-subject in an end-to-end fashion purely on the perception trials using Adam optimization with a constant learning rate of 0.0001 [13]. For both datasets we used 60 % of available perception trials for training and another 20 % for validation. The remaining 20 % and the imagination trials were used for testing. All trials were shuffled randomly before training. For tests on imagination data, we evaluated both imagination trials whose corresponding perception trials were included in the training as well as entirely unknown trials. All models were trained up to saturation of the Mel spectrogram reconstruction loss, between 1000-2000 epochs. Reconstruction loss was computed as the mean squared error between reconstructions and targets.

5.5 Introspection

After training we inspected the learned latent space by linearly interpolating between multiple existing EEG inputs extracted either from the training or testing dataset. This way, we received embeddings for the given inputs as well as a fixed number of embeddings that connect them in the learned projection space. We then used the model to reconstruct the Mel spectrogram and EEG signal for the embeddings.

6. QUALITATIVE ANALYSIS OF MUSICAL STIMULUS RECONSTRUCTION

6.1 Perceived stimulus reconstruction

We were able to use the modified VCCA model to reconstruct the Mel spectrograms of perceived audio within both datasets at various levels of accuracy. Figure 2 shows exemplary reconstructions of speech and sine wave tone patterns for intra-subject training and testing on both trial types of the OpenMIIR speech dataset. The reconstructions are characterized by rhythmic and timbral alignment with the target. In some cases we noticed erroneous temporal shifts of the whole predicted rhythmic pattern within a reconstructed excerpt. Additional tests with smaller window sizes lead to a decrease in amount and size of such errors, while increasing the amount of false positive predictions of both sine wave and speech patterns. In some cases speech and sine wave patterns were mixed up, but still with correct temporal alignment of note onset positions between target and predictions. Figure 3 shows reconstructions after training on all subjects of the OpenMIIR speech dataset. Multi-subject training lead to results with improved temporal alignment of targets and predictions. Here, in more cases the two timbres (sine wave and speech pattern) were confused. This indicates that the correct prediction of the timbre is more subject-specific than the temporal and rhythmic aspects. Increasing the amount of training data for both trials enhanced the overall reconstruction quality, training only on the speech trials still lead to temporally meaningful reconstructions of the sine wave tone patterns. We found the stimulus reconstruction quality to be best when including 4 subjects for cross-subject training and testing.

Increasing the amount of dropout within the EEG decoder (up to 40 %) turned out to be crucial for reconstructions of comparable quality for trials in subjects that were excluded entirely from the training procedure. Training with randomized window start positions and using overlapping overlapping windows proved to enhance the reconstruction quality. This suggests that Mel spectrogram reconstruction quality for this dataset is limited by the amount of available training data.

Compared to the OpenMIIR dataset, the NMED-T dataset provided more training data with increased target complexity. The reconstructions showed different characteristic in visual inspection. Often times, the timbre reconstruction dominated the reconstruction of temporal aspects, especially in parts that featured multiple instruments or singing voice. In fewer cases, but within all songs, the onsets of percussion, speech or other sounds were reconstructed. For all trained models, timbre reconstruction was visible after around 500 epochs, while temporal aspects were learned at later stages. Figure 4 provides examples for reconstructed excerpts of the perceived full-length

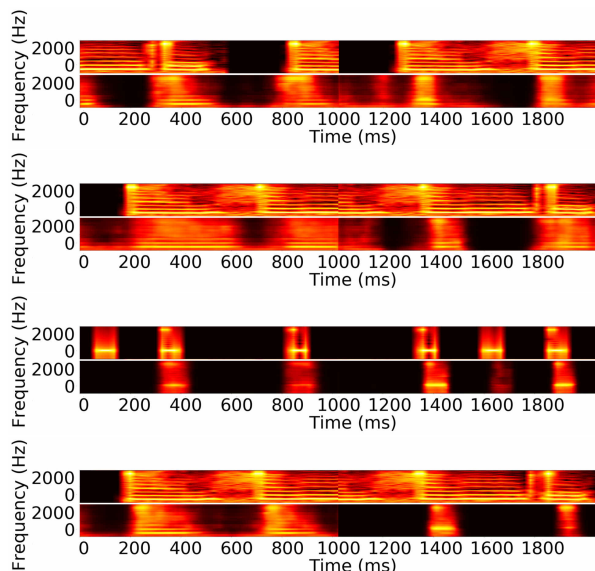


Figure 2. Mel spectrogram reconstructions of perceived rhythmic trials for the VCCA model trained on subject 'P13' of the OpenMIIR speech dataset. Target stimuli are presented above their reconstructions.

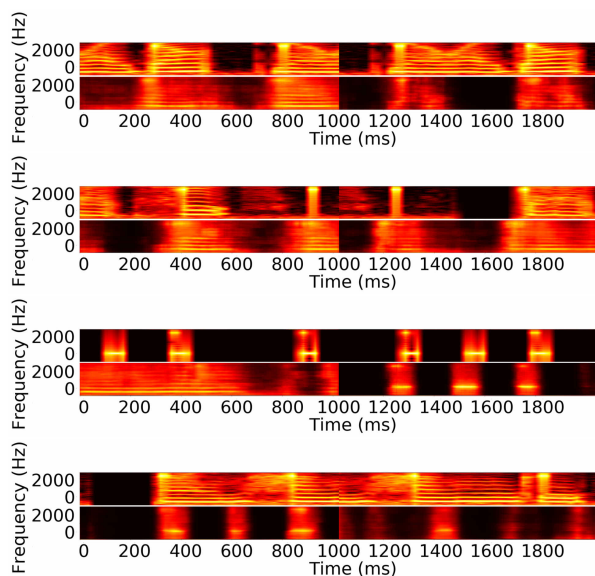


Figure 3. Mel spectrogram reconstructions of perceived rhythmic trials for the VCCA model trained on all subjects of the OpenMIIR speech dataset. Target stimuli are presented above their reconstructions.

songs contained in the NMED-T dataset. We found no substantial difference in the quality of reconstructions within subjects included into training and those from subjects excluded during training. This might be due to the small amount and long duration of 10 stimuli in combination with a single presentation per stimulus. Increasing the dropout rate after each convolutional layer in the EEG encoder over 30 % increased the models tendency to reconstruct temporal aspects, such as percussion onsets. Training sets with a larger amount of subjects generally

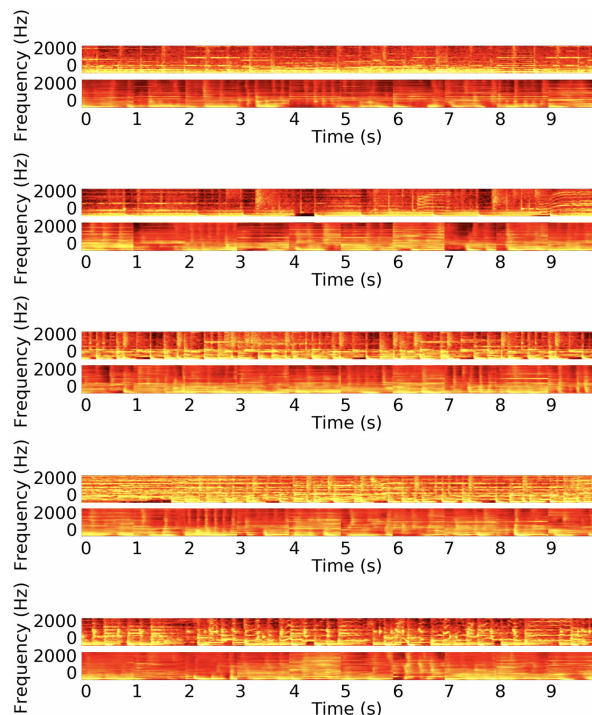


Figure 4. Excerpts of reconstructed Mel spectrograms from the NMED-T dataset. The target stimuli are shown above their reconstructions. The two top rows are based on training on all subjects. The three bottom rows are based on training on 10 subjects and testing on subjects that were excluded during training.

improved reconstruction quality. Furthermore, the introduction of overlapping EEG input windows increased the amount of reconstructed temporal features. Models trained for more than 2000 epochs showed more sparse reconstruction within the frequency domain. This indicates that adding more data and increasing training length can further increase the reconstruction quality for naturalistic music. Often times, the size of temporal misalignments was equal at all positions within reconstructed excerpts. This indicates that the reconstruction quality is dependent of the window size. Future work could test this assumption by simultaneously training on EEG or audio excerpts of various sizes within different encoders of the VCCA model. This would furthermore allow the representation of the latent concepts to include contexts of various size. For example, in the audio domain, such contexts could range from single note onsets to changes in song structure.

6.2 Imagined stimulus reconstruction

VCCA models trained on perceptual OpenMIIR speech data could be applied to imagination trial reconstruction. The reconstructed stimuli showed the same typical rhythmic patterns and could be divided into speech and sine wave predictions. However, the correct rhythmic predictions were less often visible and more blurry. It is important to note that the imagination was performed superimposed on a constant metronome click. This means, that only the difference

between the rhythmical structure and timbre was based on pure imaginative processes, while there were still perceptual cues for temporal alignment. Models trained on multi-subject perceptual data showed less blurry reconstructions. Adding private encoders with imagination based EEG signal did not cause a visible increase in reconstruction quality.

7. QUALITATIVE ANALYSIS OF LEARNED AUDITORY CONCEPTS

We found musically meaningful representations of the OpenMIIR speech stimuli in the latent space of models trained intra-subject as well as cross-subject. Both EEG signal from training and testing subsets could be used to produce continuous interpolation. Processing EEG inputs from both testing and training data sets and using the target audio stimuli as validation, we found continuous representation across the temporal, rhythmical and timbral domain. For any given input, we could change the temporal position of the rhythmical pattern as well as the timbre (within speech and sine wave tones). Furthermore, the latent space enabled interpolation between metronome clicks and the sine wave tones of increased loudness. However, this difference was found to a lesser degree with data of the test set. Figure 5 (a) shows an example for the interpolation between 3 embeddings based on EEG inputs of the OpenMIIR speech training data set. Here, interpolation was done while simultaneously shifting the temporal position of the rhythmical pattern within the reconstructed excerpt. The non-syncopated excerpt was further interpolated into its representation with speech signal. Figure 5 (b) shows topographic projections of the brain activity reconstructed for each embedding that was computed in Subfigure (a). For the sake of clarity we show six topographic plots out of the total amount of 512 per embedding. Qualitative comparison of the EEG signal with the original inputs indicated that overfitting the EEG data is not possible when we stop training when the audio reconstruction loss is saturated. For other use cases, higher quality EEG reconstructions could be achieved with different training procedures, such as unsupervised EEG reconstruction pretraining. Models with smaller latent embeddings sizes (e.g. 8 units) did still produce meaningful and continuous interpolations, but with more blurring across the temporal and frequency domains. The model forces EEG and audio to be shared even in these smaller latent spaces. The neuroscientific meaningfulness of the EEG reconstructions might further be validated in future work, for example with shared fMRI representation in private encoders.

8. CONCLUSIONS

In this paper, we presented the application of a multi-view generative model for shared auditory concept learning and musical stimulus reconstruction from EEG signals. We showed that the model can learn representations of simple rhythm and timbre related concepts that are shared in audio and EEG data. Furthermore, we could see first successes in approaching naturalistic music and imagined stimulus reconstruction. The presented framework is designed to be expandable to additional modalities, such as fMRI data, or additional reconstruction

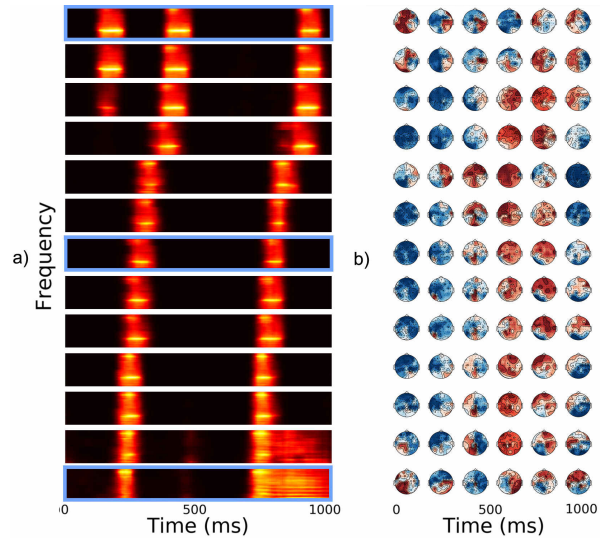


Figure 5. (a) Reconstructed Mel spectrograms after interpolation in the learned latent space learned for Subject 'P13' of the OpenMIIR speech dataset. Embeddings that correspond to real EEG inputs are framed. (b) Topographic visualization of the reconstructed temporal brain activity. Each row represents the brain activity reconstructed for the embedding in the same row of Subfigure (a).

targets, such as emotional aspects of music cognition. In combination with the ability to perform introspection on the shared representation of stimuli and electrophysiological responses, the model can be an aid for future EEG based music information retrieval and research in music cognition.

9. ACKNOWLEDGMENTS

The OpenMIIR speech dataset was kindly shared by the Music and Neuroscience Lab at Western University in London, Ontario. The authors especially would like to thank Avital Sternin who recorded the data. This research has been funded by the Federal Ministry of Education and Research of Germany (BMBF).

10. REFERENCES

- [1] A. Aroudi, B. Mirkovic, M. De Vos, and S. Doclo. Auditory attention decoding with eeg recordings using noisy acoustic reference signals. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 694–698. IEEE, 2016.
- [2] S. Chandar, M. M. Khapra, H. Larochelle, and B. Ravindran. Correlational neural networks. *Neural computation*, 28(2):257–285, 2016.
- [3] L. K. Cirelli, D. Bosnyak, F. C. Manning, C. Spinelli, C. Marie, T. Fujioka, A. Ghahremani, and L. J. Trainor. Beat-induced fluctuations in auditory cortical beta-band activity: using eeg to measure age-related changes. *Frontiers in psychology*, 5:742, 2014.
- [4] C. Du, C. Du, and H. He. Sharing deep generative representation for perceived image reconstruction from human brain activity. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 1049–1056. IEEE, 2017.
- [5] Y. Fujiwara, Y. Miyawaki, and Y. Kamitani. Modular encoding and decoding models derived from bayesian canonical correlation analysis. *Neural computation*, 25(4):979–1005, 2013.
- [6] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, et al. Meg and eeg data analysis with mne-python. *Frontiers in neuroscience*, 7:267, 2013.
- [7] K. Hoenig, C. Müller, B. Herrnberger, E.-J. Sim, M. Spitzer, G. Ehret, and M. Kiefer. Neuroplasticity of semantic representations for musical instruments in professional musicians. *NeuroImage*, 56(3):1714–1725, 2011.
- [8] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.
- [9] E. Jones, T. Oliphant, and P. Peterson. Scipy: open source scientific tools for python. 2014.
- [10] I. Kavasidis, S. Palazzo, C. Spampinato, D. Giordano, and M. Shah. Brain2image: Converting brain signals into images. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1809–1817. ACM, 2017.
- [11] M. Kiefer and L. W. Barsalou. 15 grounding the human conceptual system in perception, action, and internal states. *Action science: Foundations of an emerging discipline*, page 381, 2013.
- [12] M. Kiefer, E.-J. Sim, B. Herrnberger, J. Grothe, and K. Hoenig. The sound of concepts: four markers for a link between auditory and conceptual brain systems. *Journal of Neuroscience*, 28(47):12224–12230, 2008.
- [13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [14] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [15] S. Losorelli, D. T. Nguyen, J. P. Dmochowski, and B. Kaneshiro. Nmed-t: A tempo-focused dataset of cortical and behavioral responses to naturalistic music. ISMIR, 2017.
- [16] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.
- [17] M. Meyer, S. Baumann, and L. Jancke. Electrical brain imaging reveals spatio-temporal dynamics of timbre perception in humans. *Neuroimage*, 32(4):1510–1523, 2006.
- [18] N. Moran. Social implications arise in embodied music cognition research which can counter musicological individualism. *Frontiers in psychology*, 5:676, 2014.
- [19] R. Näätänen and T. Picton. The n1 wave of the human electric and magnetic response to sound: a review and an analysis of the component structure. *Psychophysiology*, 24(4):375–425, 1987.
- [20] S. Nozaradan, I. Peretz, M. Missal, and A. Mouraux. Tagging the neuronal entrainment to beat and meter. *Journal of Neuroscience*, 31(28):10234–10240, 2011.
- [21] S. Nozaradan, I. Peretz, and A. Mouraux. Selective neuronal entrainment to the beat and meter embedded in a musical rhythm. *Journal of Neuroscience*, 32(49):17572–17581, 2012.
- [22] A. Ofner. Reconstruction of perceived and imagined music from eeg recordings with deep neural networks. In *Proceedings of the MEI: CogSci Conference 2017*, page 53.
- [23] F. Raposo, D. M. de Matos, R. Ribeiro, S. Tang, and Y. Yu. Towards deep modeling of music semantics using eeg regularizers. *arXiv preprint arXiv:1712.05197*, 2017.
- [24] R. S. Schaefer, P. Desain, and P. Suppes. Structural decomposition of eeg signatures of melodic processing. *Biological psychology*, 82(3):253–259, 2009.
- [25] A. Shahin, L. E. Roberts, C. Pantev, L. J. Trainor, and B. Ross. Modulation of p2 auditory-evoked responses by the spectral complexity of musical sounds. *Neuroreport*, 16(16):1781–1785, 2005.
- [26] A. Sternin, S. Stober, J. Grahn, and A. Owen. Tempo estimation from the eeg signal during perception and imagination of music. In *1st International Workshop on Brain-Computer Music Interfacing/11th International*

Symposium on Computer Music Multidisciplinary Research (BCMI/CMMR15), 2015.

- [27] S. Stober, D. J. Cameron, and J. A. Grahm. Using convolutional neural networks to recognize rhythm stimuli from electroencephalography recordings. In *Advances in neural information processing systems*, pages 1449–1457, 2014.
- [28] S. Stober, A. Sternin, A. M. Owen, and J. A. Grahm. Towards music imagery information retrieval: Introducing the openmiir dataset of eeg recordings from music perception and imagination. In *ISMIR*, pages 763–769, 2015.
- [29] D. Stuss, A. Toga, J. Hutchison, and T. Picton. Feedback evoked potentials during an auditory concept formation task. In *Progress in brain research*, volume 54, pages 403–409. Elsevier, 1980.
- [30] M. S. Treder, H. Purwins, D. Miklody, I. Sturm, and B. Blankertz. Decoding auditory attention to instruments in polyphonic music using single-trial eeg classification. *Journal of neural engineering*, 11(2):026009, 2014.
- [31] R. Vigo, M. Barcus, Y. Zhang, and C. Doan. On the learnability of auditory concepts. *The Journal of the Acoustical Society of America*, 134(5):4064–4064, 2013.
- [32] W. Wang, X. Yan, H. Lee, and K. Livescu. Deep variational canonical correlation analysis. *arXiv preprint arXiv:1610.03454*, 2016.

EXPLORING MUSICAL RELATIONS USING ASSOCIATION RULE NETWORKS

Renan de Padua^{1,2}

Verônica Oliveira de Carvalho³

Solange de Oliveira Rezende¹

Diego Furtado Silva⁴

¹ Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo, São Carlos, Brazil

² Data Science Team, Itaú-Unibanco, São Paulo, Brazil*

³ Instituto de Geociências e Ciências Exatas – Universidade Estadual Paulista, Rio Claro, Brazil

⁴ Departamento de Computação – Universidade Federal de São Carlos, São Carlos, Brazil

padua@icmc.usp.br, veronica@rc.unesp.br, solange@icmc.usp.br, diegofs@ufscar.br

ABSTRACT

Music information retrieval (MIR) has been gaining increasing attention in both industry and academia. While many algorithms for MIR rely on assessing feature subsequences, the user normally has no resources to interpret the significance of these patterns. Interpreting the relations between these temporal patterns and some aspects of the assessed songs can help understanding not only some algorithms' outcomes but the kind of patterns which better defines a set of similarly labeled recordings. In this work, we present a novel method to assess these relations, constructing an association rule network from temporal patterns obtained by a simple quantization process. With an empirical evaluation, we illustrate how we can use our method to explore these relations in a varied set of data and labels.

1. INTRODUCTION

Digital music repositories and streaming music services have become increasingly popular in the last decades. Along with this growth, algorithms to automatically organize, navigate, and search on music collections are more and more necessary. For this reason, Music information retrieval (MIR) has been gaining considerable attention in both industry and academia.

There is a multitude of MIR methods that rely on assessing subsequences of features. In other words, these methods extract features from the audio in a sliding window fashion and use successive subsets of these features to take decisions over the data. One example is the music genre classification, in which a common approach is to aggregate

features from subsequences to obtain a more robust set of features [2, 8]. Moreover, Silva et al. [12] showed how assessing distances between subsequences can be used as a subroutine for different MIR tasks, from cover recognition to visualization.

In this work, we propose the use of a novel category of association rules networks to support understanding the relations between sequential patterns and labels which describe our data. The exploration of these association rules may provide insights on what kind of pattern defines one label, which may have implications on musicology or other MIR tasks.

For instance, consider the genre as the target label. If one pattern (or a group of patterns) happens with high confidence for only one label, this may help to explain the characteristics which define that genre. Also, it may guide us to understand how to improve music classification algorithms. Besides, our method helps us to find patterns shared between different labels. This kind of relation can be used, for example, to improve music recommendation systems, as well as provides insights on the musical influences between different labels.

Figure 1 illustrates one example of relations found by our method. It represents that, for a given dataset labeled with genre information, when the pattern indexed by 10 appears in a recording, we can say that recording belongs to the label "classical" with 100% of confidence. Also, if the patterns 23 and 4 happens in the same recording, it belongs to the label "classical" with 94% of confidence. The patterns correspond to quantized subsequences of features – Mel-Frequency Cepstrum Coefficients (MFCC), in this case – and can be assessed visually or by listening to the music excerpt that generated it.

In this paper, we introduce algorithms to represent association rules in a graph, aiming to provide a visual tool to understand the relations between the features that compose it. Then, we apply our method on different datasets, described by varied classes of labels, to demonstrate how to use this representation to understand the relations between features and labels, as well as which patterns link two different labels.

* The opinions expressed in this article are those of the authors and do not necessarily reflect the official policy or position of the Itaú-Unibanco.



© Renan de Padua, Vernica Oliveira de Carvalho, Solange Rezende, Diego Furtado Silva. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Renan de Padua, Vernica Oliveira de Carvalho, Solange Rezende, Diego Furtado Silva. "Exploring Musical Relations Using Association Rule Networks", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

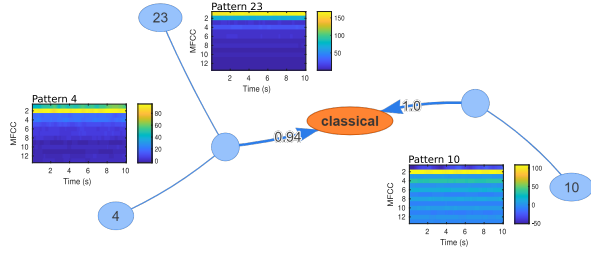


Figure 1: Example of association rule network

The remainder of this paper is organized as following. Section 2 introduces the main concepts of association rules and association rules networks, accompanied by related work. The method presented in this work is presented in Section 3. Section 4 presents our experimental evaluation. Finally, Section 5 concludes this work.

2. BACKGROUND AND RELATED WORK

The association rules were first proposed by Agrawal et. al. [1]. The goal of the proposed approach was identifying, in supermarket buying transactions, what were the items that customers used to buy together. This analysis was made aiming to help the supermarket owners to organize their stock in order to raise the sales of some specific items. To understand how association rules discovery works, we first define some concepts related to it.

Definition 1 Given a set of items I , a set of transaction T consisting of subsets of I , an association rule is the relation $A \rightarrow B$, where A and B are subsets of I and $A \cap B = \emptyset$.

A is called *antecedent* (or Left-Hand side - LHS) and B is called *consequent* (or Right-Hand side - RHS). The association rule can be read as: “given that A happened, B also happens in $c\%$ of the cases”, where $c\%$ is the association rule confidence. Support ($s\%$) is another important measure in the association rule, that describes the percentage of transactions in which all the items of the rule appear.

Definition 2 The support $\sigma(A)$ of a subset $A \subset I$ is defined by the percentage of transactions that contain all the items presented in A .

Definition 3 The confidence of a rule $A \rightarrow B$ is given by the percentage of transactions that contain all the items in A that also contain all the items in B . The confidence is calculated by $\frac{\sigma(A \cup B)}{\sigma(A)}$.

Also, the Lift is a widely used measure to assess the association rule quality. It evaluates if the items on the LHS are positively or negatively dependent with the items on RHS, or if these sets are independent.

Definition 4 The lift value of a rule $A \rightarrow B$ is given by the probability of A and B happen together divided by the probability of A times the probability of B , calculated by $\frac{\sigma(A \cup B)}{\sigma(A)\sigma(B)}$.

The Association Rule Network (ARN) was proposed by Chawla et. al. [5] and extended by Pandey et. al. [10] and by Chawla [4]. The ARN models all the association rules that are directly or indirectly correlated to an specific item (called objective item) in a directed acyclic graph (DAG), pruning all the other rules that are not interesting in the objective item context. According to Pandey et. al. [10], the ARN modeling is capable of pruning the rules into a specific context, defined by the selected objective item.

According to Thulasiraman et. al. [14], a graph $G = (V, E)$ consists of two sets: a finite set of vertices V and a finite set of edges E . Each vertex represents an object in the graph and an edge represents a link between two vertices. Also, it is possible to define the graph $G = (V, E, W)$, consisting of three sets: the V and E sets remain the same, while the W set represents the weight of the edges in the graph G . In a graph that does not have weights, the W may have 1 where the connection exists and 0 where it does not exists. If the edges are ordered, i.e., the edges are identified as “from” vertex and “to” vertex, then it is said that the graph is directed because its edges contain a direction. A Directed Acyclic Graph (DAG), is a particular type of graph that contains no cycles.

Definition 5 We say that a directed graph contains cycles if given a graph G containing N vertices V , the graph has a path that goes from v_x to v_y and there is also a path from v_y to v_x .

An example of an ARN with objective item “ G ” is presented on Figure 2. In this example, the following rules were modeled: $A \rightarrow D$, $B \rightarrow D$, $B \rightarrow C$, $C \& D \rightarrow E$ and $E \& F \rightarrow G$. All the other extracted rules were pruned because they were not interesting in the context of the G item exploration.

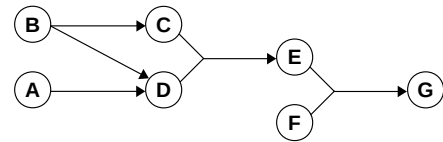


Figure 2: Example of ARN with objective item = G , adapted from Pandey et. al. [10].

The final ARN is a directed acyclic graph that flows to the objective item, i. e., all elements on the graph have directed connections that leads to an objective item. This graph models the association rules that better explains the occurrence of the selected objective item. The modeling can be used to build a hypothesis, based on the correlation among the items in the database and the objective item that the user wants to understand.

The ARN algorithm can be described in 3 steps, described as follows.

Step A Given a database D , a minimum support, and a minimum confidence value, we extract the association rules using an algorithm, like apriori [1]. The RHS must have size 1.

Step B Considering all the items in the association rules' RHS, the user selects one item to represent the objective item.

Step C Models all the rules that have the objective item in the RHS (considered level 0) or already modeled on other levels. The modeling must fulfill the 2 restrictions: 1 - The LHS of the rule is not present in the network or 2 - the LHS level is equal the RHS level + 1.

3. EXTRACTING AND ASSOCIATING PATTERNS

The proposed method relies on two main steps, which we describe in details in this section. The first one extracts frame-level features from the audio and quantize them to create a limited dictionary of patterns. With this procedure, we transform the recordings in our database in a transaction-like representation, where each recording is represented by the patterns presented in it. This representation is used in the second step of our method.

The second one extracts and selects the best rules to describe all the labels on the dataset. We extract the association rules, prune the rules which do not present interesting knowledge on the labels context and build a DAG, explaining how the patterns correlates to the labels.

3.1 Extracting and Quantizing Subsequence Patterns

As aforementioned, the first step of our method relies on associating each recording to one or more temporal patterns in a bag-of-patterns representation. For this, we split this phase into different intermediate steps to create a dictionary and, then, associate each subsequence of features from each recording to a codeword in this dictionary.

Initially, we extract frame-level Mel-Frequency Cepstrum Coefficients (MFCC) and Constant-Q chromagram from the raw audio. For this, we used the LibROSA package for music and audio analysis [9]. Since our main purpose is not comparing different parameter settings of each feature extraction procedure, we applied the default parameters defined by the tool. We chose these features since they represent distinct characteristics of music. Specifically, the MFCC and chromagram are intrinsic to timbre and pitch information, respectively.

To associate each feature vector to a pattern, we first need to create a dictionary. For this, we applied the simple k-Means clustering algorithm on subsequences of the feature vectors. The centroid of each cluster defines one codeword, i.e., the prototype of a temporal pattern. For the sake of memory and time efficiency, we only used one-third of the subsequences to estimate the centroids.

Once the codewords are defined, we associate all feature subsequences of each song with codewords, according to their proximity. In other words, for each recording, we find the nearest centroid of each subsequence of features and annotate it. At the end of this step, each recording is described by the set of codewords that appear in its subsequences. In the case of repetition, we remove these recurrences.

Although this step relies on defining the number of codewords, we leave details regarding this to Section 4.

3.2 Extended Association Rule Network

The *Extended Association Rule Network* (ExARN) aims to aid the user to understand the data and build a hypothesis from that data. The objective is to model the association rules in a graph, explaining the correlation among the attributes in the database according to a set of attributes selected by the user. For instance, consider the contact lens database¹, which is aimed to automatically prescribes contact lenses to patients. In this case, the user may be interested not in the classification, but in understanding which are the patient's characteristics that define which kind of lens will be prescribed.

The ExARN is conceptually different from associative classification algorithms and decision trees, which build the model only based on the classes, ignoring all other correlations present on the database. The ExARN explores a set of previously extracted association rules and, then, searches for the best rules to describe the set of attributes defined by the user. Also, it has some interesting properties: i) it is built on a DAG, which means that there are no cycles on the network, ii) it is built on levels, every rule has the LHS on level x and the RHS on level $x + 1$, for example, an objective attribute will mandatorily be on the RHS and on level 0, all the attributes on LHS that contains that attribute on RHS will be on level 1 and so on.

The ExARN is built in three steps. The first step consists of the association rule mining phase. The only restriction added to this step, if compared to a conventional association rule mining, is that the rules must an RHS with size 1. This restriction was added, so each rule explains only one attribute, reducing the complexity for the user to explore the result.

The second step is the objective attribute definition. This step will guide the entire exploration, as it will define the objective attributes which the network will be built from. The user must select the attributes that will be explored. This selection must be done considering also the possibility that these attributes have a common cause to be explored or refuted.

The last step consists of the ExARN construction. This step is responsible for getting all the rules that are directly or indirectly related to the objective attributes and model them following the ExARN restrictions. The ExARN building is done recursively. First, all the attributes selected as objective attributes are modeled in the graph on the level 0. Then, all the rules that the LHS' attributes are not in the graph and have the RHS' attributes on level 0 are modeled on the network. The same process is done to all the attributes on level 1, then to attributes on level 2 and so on. Until there are no more rules to be modeled. The ExARN can be defined as follows:

Definition 6 Given a set of association rule R , containing rules with RHS of size 1, and a set of objective attributes

¹ <https://archive.ics.uci.edu/ml/datasets/lenses>

Z with size ≥ 2 , the ExARN is a DAG that models all the rules related to the items on Z , such as:

1. Each vertex models a rule $r \in R$.
2. From any point of the network, it is always possible to reach at least 1 vertex representing an attribute from Z .
3. Given a vertex $v \in \text{ExARN}$, such as $v \notin Z$. There is no path from any item on Z to v .

The ExARN presents a wider exploration if compared to the presented ARN because it allows the exploration of 2 or more objective items at once. That way, the user might discover which patterns are interesting in the context of a single objective item, also discovering which patterns are interesting for a set of objective items.

4. EXPERIMENTAL EVALUATION

In this section, we describe the experimental evaluation performed to assess the ExARN in different scenarios of music data. We note that we made a supplementary website² where we make available source codes and detailed results, as well interactive visualizations of the networks presented in this section and some audio excerpts to exemplify some of the mentioned patterns.

4.1 Rule discovery and association setup

After extracting the subsequence patterns, the database pattern extraction begins. The sequence described here was applied to all the databases.

First, the association rules were extracted. We mined the association rules using the *arules* package in R³. This step needs the definition of some parameters. The support value, which is a threshold of minimum occurrence was set to 1%. This value was chosen because the databases were divided into more than 10 different labels, so each subpattern will have a maximum occurrence of $\frac{1}{\text{numLabels}}$ on each label. Defining the minimum support to 1% will remove only the subpatterns that rarely happens. The other parameter is called confidence, which can be defined in terms of posterior probability as: $\text{Conf}(A \rightarrow B) = P(B|A)$. We defined the minimum confidence on 25%, which means that the B must happen in, at least, one-quarter of the occurrence of A .

To make sure that the association rules are positively dependent, we applied a filter using the lift measure. The threshold was defined at 2, as rules with lift value ≥ 1 are considered to have a positive dependency. We selected the value 2 instead of 1 as this value discards the rules that are on the edge of the measure. Then, we applied the ExARN algorithm over the association rules, considering all the labels as the objective items

4.2 Datasets

The datasets used in our experimental evaluation aim to provide us a diversity of characteristics and labels. For this reason, we used diversified datasets and for one of them, we used different labels for the same bag-of-patterns.

One of the most common labels in MIR datasets is the genre. We evaluated our method in this context using the GTZAN dataset [15]. This database is composed of 1000 thirty-second tracks, perfectly balanced in ten genres.

Another way to categorize music data is according to the artist who recorded it. We also evaluated our method in this scenario, using the Artist20 dataset [7]. This dataset contains 1413 songs performed, as its name suggests, by 20 artists mostly of pop or rock music. The number of recordings is not balanced among the artists.

Finally, we assessed the FMA dataset [6]. Moreover, we took the fact that many of the recordings in these databases are associated with “social” features provided by Echonest⁴ to evaluate our method on varied labels for the same data. Specifically, we applied our method targeting seven distinct labels: acousticness, danceability, energy, instrumentalness, liveness, speechiness, and valence. In order to transform these continuous features in class-like values, we discretized the features in five equally spaced intervals, representing low, mid-low, mid, mid-high, and high levels of each characteristic. As we used the default small portion of this data and only kept information from the recordings associated with Echonest features, we ended with 1023 tracks.

4.3 On the Impacts of the Codebook’s Size

The quantizing phase of our method has one parameter that affects the results of our method. The number of clusters to create the dictionary, i.e., the number of codewords, have a direct impact on the confidence. Particularly, the higher the number of codewords, the lower the confidence of the rules. Conversely, the lower the number of clusters, the higher is the confidence.

As we experimented with 25, 50, and 100 codewords in each dataset, we stick our analysis on the lower value. However, we notice that a high number of codewords may be more appropriate for datasets with a high number of labels. Otherwise, the intersection between the labels would be too high to find meaningful rules. In this paper, the higher number of assessed labels is 20.

There is another characteristic of using fewer codewords regarding the interpretability of the results. As the codewords are centroids, if we use too few clusters the codewords will look “blurry” or few informative. However, we noticed that it does not hamper the rule discovery and the music excerpts that were associated with each pattern can listen to a better understanding of what that pattern represents. Also, once the ExARN is computed, we can break a pattern A in more parts, B and C for instance, in a procedure similar to the Bisect k-Means approach [13]. With this operation, we turn the patterns more specific.

²<https://sites.google.com/view/music-exarn>

³available at <https://cran.r-project.org/web/packages/arules/index.html>

⁴<http://the.echonest.com/>

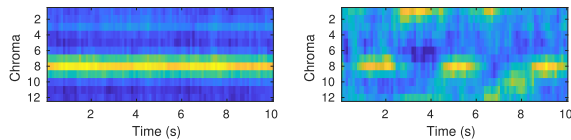


Figure 3: Examples of patterns in different resolutions, given by the number of codewords in the dictionary: 25 (left) and 100 (right)

Then, instead of reading the association regarding A , one may read it regarding “ B or C .” Figure 3 illustrates two chroma patterns obtained from different number of code-words.

4.4 Results and Discussion

In this section, we present some of the results obtained by our method. For simplicity, we split it into sections regarding each analyzed dataset and the target labels of each of them. Specifically, the results on GTZAN, Artist20, and FMA are presented in the sections regarding genre, artist, and the “social features” from Echonest.

We acknowledge that interpreting the patterns and, as consequence, the meaning of some rules, only using textual and static graphical elements is a difficult matter. For this reason, we make available on our website interactive visualizations of ARNs obtained in our experiments, as well as some music excerpts that are representative of relevant patterns.

We note that association rules discovery is an unsupervised task and, therefore, there is no quantitative evaluation measure to assess the quality of these rules. The only way to objectively evaluate the value of the learned rules would be use it as an intermediate step of an algorithm to perform other task, such as classification or recommendation systems. We leave this as an intention for future work.

4.4.1 Genre

Using MFCC, we found a few interesting rules that associate patterns with some of the target labels. One example is the one illustrated by Figure 1. We also found similar associations to other genres. Specifically, for metal, reggae, and jazz. The latter two, however, with lower confidences (around 33%).

The most interesting relations in this dataset come from the patterns shared between distinct labels. Figure 4 presents the entire network for this dataset when associating patterns representing 10 seconds of audio.

Some of the patterns are associated with several genres in the presented network. These patterns are not suitable for differing the characteristics of each genre. However, we commonly see music elements that are used in songs belonging to different genres. So, this kind of multiple relations was expected. For instance, we observed a pattern associated with the genres disco, pop, and hip-hop.

On the other hand, we found patterns that link pairs of genres. These patterns directly associate two genres that

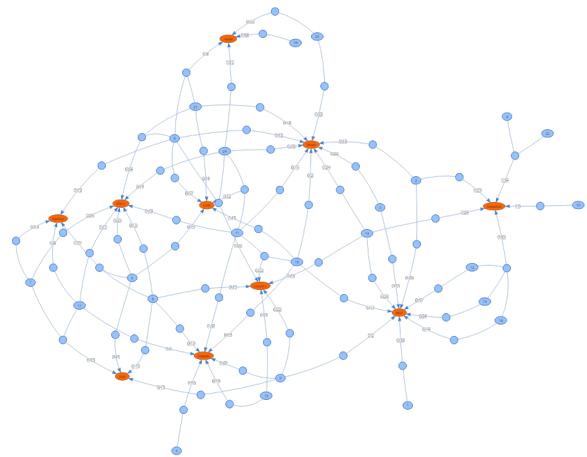


Figure 4: ExARN obtained by associating MFCC patterns from 10 seconds of audio in the GTZAN dataset

have somehow similar timbral information. It may help explain the influence between genres or the mutual influences of each pair. One example of two genres linked by this kind of association is the pair metal and blues. Another interesting relation regards the genres classical and jazz. They have two patterns that are common for both. However, they usually happen together (i.e. in the same recording) in classical pieces but separately in jazz songs. We noticed that we did not achieved interesting associations when using chroma features in this case.

4.4.2 Artist

Is there any link between Metallica and Roxette regarding tonal patterns in their songs? The answer is “yes, there is Tori Amos.” Using subsequences of chroma vectors representing 10 seconds in the Artists20 dataset, we found that these three artist have sets of four tonal patterns each that are confidentially linked to each of them. Moreover, Tori Amos shares one of its patterns with Metallica and another one with Roxette. Figure 5 illustrates these relations.

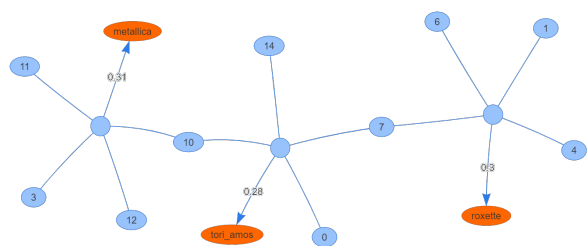


Figure 5: Subset of the rules obtained from chroma patterns in the Artist20 dataset

This kind of relation is commonly seen when using MFCC as the input features in this dataset. For instance, when applying the ExARN algorithm on five seconds excerpts, we found rules with (at least) the minimum sup-

port for seven artists. When using ten seconds excerpts, we found rules for ten artists. In all of these cases, there is at least one timbral pattern for each artist which links it to another one.

We notice that these links are relevant since they are not trivial. In other words, if a timbral or chromatic pattern is present in many songs of several artists, the rules containing it would have a very low support. Therefore, these links show how two (or more artists) are musically related each other by patterns that are not so commonly used.

4.5 Echonest Labels

We evaluated the ExARN on seven different labels from Echonest. We found relevant rules in all of them but the *speechiness*. For the other labels, the association rules network demonstrated regularities in their behavior. For instance, when we assess the rules associated to a single label, usually we cannot find association with minimum support for the intermediate values. This may happen because the middle labels are fuzzy. In other words, the assessed patterns can describe solely the high and low characteristics at a minimum support. Figure 6 illustrates this fact regarding the acousticness.

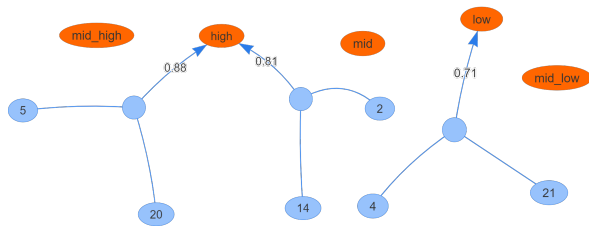


Figure 6: Association rules from MFCC that are not shared by different intervals of acousticness

When we analyze the rules that associate different labels, three main behaviors appear. The first one is not finding patterns which relate different levels of these characteristics. Figure 7 illustrates the second, and most common, behavior. In this case, the extremes are separate into distinct components, i.e., the high and mid-high values are linked by some patterns, similarly to what happens between low and mid-low values.

Finally, in some cases, the labels representing extreme values are directly linked by one or more patterns while some patterns play the rule of “bridges” between these extremes. Figure 8 illustrates this scenario.

5. CONCLUDING REMARKS

In this paper we presented the use of extended association rules networks for exploring the correlation between temporal patterns and labels of music in different scenarios.

To evaluate the meaning of the discovered rules, we presented some reasoning to verify the quality of these rules as a qualitative approach. One example is evaluating the existence of links between labels we consider similar to each other. In other cases, our rules may explicit some relations

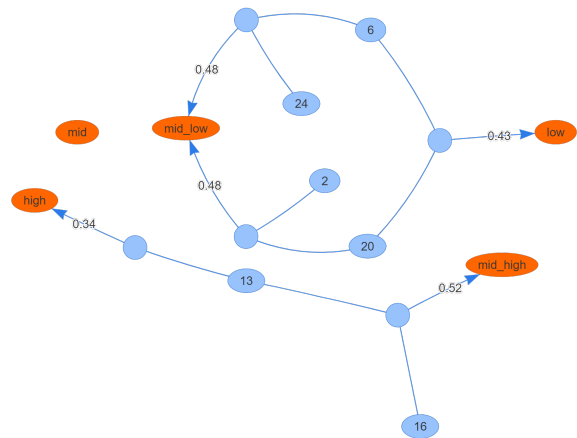


Figure 7: Association rules from MFCC shared by different intervals of energy

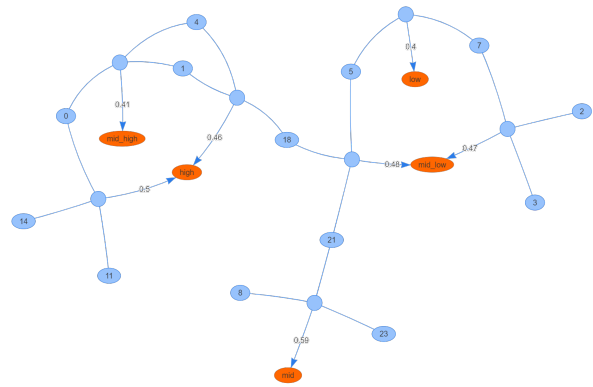


Figure 8: Association rules from chroma features shared by different intervals of energy

that are not obvious. In both cases, studying the patterns that composes such relations can be useful to understand music data in several aspects. For instance, in some cases, we could find interesting relations using chroma vectors in scenarios where these features are not usually considered (e.g. to describe valence and energy).

As future work, we intend to improve the quantization step so we reduce the impact of the codebook generation and we can ignore several patterns, considering them irrelevant. For this, we may evaluate the use of some density-based clustering strategy [3, 11]. Also, we will evaluate the use of ExARN as an intermediate step to improve recommendation systems. Finally, we intent to evaluate if this kind of association rules network can improve the interpretability of music-related learned features.

6. ACKNOWLEDGEMENT

This work was funded by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and São Paulo Research Foundation (FAPESP) by grants #2013/26151-5, #2016/17078-0, and #2018/11755-6.

7. REFERENCES

- [1] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.
- [2] James Bergstra, Norman Casagrande, Dumitru Erhan, Douglas Eck, and Balázs Kégl. Aggregate features and AdaBoost for music classification. *Machine learning*, 65(2-3):473–484, 2006.
- [3] Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 160–172, 2013.
- [4] Sanjay Chawla. Feature selection, association rules network and theory building. In *International Workshop on Feature Selection in Data Mining*, pages 14–21, 2010.
- [5] Sanjay Chawla, Bavani Arunasalam, and Joseph Davis. Mining open source software (OSS) data using association rules network. In *Advances in Knowledge Discovery and Data Mining*, pages 461–466, 2003.
- [6] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. In *International Society for Music Information Retrieval Conference*, 2017.
- [7] Daniel P. W. Ellis. Classifying music audio with timbral and chroma features. In *International Society for Music Information Retrieval Conference*, pages 339–340, 2007.
- [8] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *International Society for Music Information Retrieval Conference*, pages 339–344. Utrecht, The Netherlands, 2010.
- [9] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Python in Science Conference*, pages 18–25, 2015.
- [10] Gaurav Pandey, Sanjay Chawla, Simon Poon, Bavani Arunasalam, and Joseph G. Davis. Association rules network: Definition and applications. *Statistical Analysis and Data Mining*, 1(4):260–279, 2009.
- [11] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.
- [12] Diego F Silva, Chin-Chia M Yeh, Gustavo Enrique de Almeida Prado Alves Batista, and Eamonn Keogh. Simple: Assessing music similarity using subsequences joins. In *International Society for Music Information Retrieval Conference*, pages 23–29, 2016.
- [13] Michael Steinbach, George Karypis, Vipin Kumar, et al. A comparison of document clustering techniques. In *ACM SIGKDD Workshop on Text Mining*, pages 525–526, 2000.
- [14] K. Thulasiraman and M. N. S. Swamy. *Graphs: Theory and Algorithms*. 1992.
- [15] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

Session D

Corpora and voice

A CROWDSOURCED EXPERIMENT FOR TEMPO ESTIMATION OF ELECTRONIC DANCE MUSIC

Hendrik Schreiber

tagtraum industries incorporated

hs@tagtraum.com

Meinard Müller

International Audio Laboratories Erlangen

meinard.mueller@audiolabs-erlangen.de

ABSTRACT

Relative to other datasets, state-of-the-art tempo estimation algorithms perform poorly on the *GiantSteps* Tempo dataset for electronic dance music (EDM). In order to investigate why, we conducted a large-scale, crowdsourced experiment involving 266 participants from two distinct groups. The quality of the collected data was evaluated with regard to the participants' input devices and background. In the data itself we observed significant tempo ambiguities, which we attribute to annotator subjectivity and tempo instability. As a further contribution, we then constructed new annotations consisting of tempo distributions for each track. Using these annotations, we re-evaluated two recent state-of-the-art tempo estimation systems achieving significantly improved results. The main conclusions of this investigation are that current tempo estimation systems perform better than previously thought and that evaluation quality needs to be improved. The new crowdsourced annotations will be released for evaluation purposes.

1. INTRODUCTION

Estimation of a music piece's *global tempo* is a classic *music information retrieval* (MIR) task. It is often defined as estimating the frequency with which humans tap along to the beat. A necessary precondition for successful global tempo estimation is the existence of a stable tempo as it often occurs in rock, pop, or dance music. To evaluate a tempo estimation system one needs the system itself, a dataset with suitable tempo annotations, and one or more metrics. One such dataset, named *GiantSteps Tempo*, has been released by Knees et al. in 2015 [6]. It was created by scraping a forum that let listeners discuss Beatport¹ songs with wrong tempo labels. Scraping was done via a script and 15% of the labels were manually verified. All 664 tracks in the dataset belong to the umbrella genre electronic dance music (EDM) with its subgenres trance, drum-and-bass, techno, etc. Since its release, several academic and

commercial tempo estimation systems have been tested against the dataset (e.g. [12]). As is common for datasets annotated with only a single tempo per track, the two metrics *Accuracy1* and *Accuracy2* were used. *Accuracy1* is defined as the fraction of correct estimates while allowing a tolerance of 4%. *Accuracy2* additionally allows estimates to be wrong by a factor of 2, 3, $\frac{1}{2}$ or $\frac{1}{3}$ (so-called *octave errors*). The highest results reported for the *GiantSteps* dataset are 77.0% *Accuracy1* by the applications NI Traktor Pro 2² (with octave bias 88 – 175) and 90.2% *Accuracy2* by CrossDJ³ (with octave bias 75 – 150).⁴ These results are surprisingly low—the highest reported *Accuracy2* values for other commonly used datasets like *ACM Mirum* [10], *Ballroom* [4], and *GTzan* [13] are greater than 95% [1]. Since EDM is often associated with repeating bass drum patterns and steady tempi [2, 7], it should be comparatively easy to estimate the tempo for this genre. We hypothesize that relatively low accuracy values were achieved for multiple possible reasons. Since the annotations were scraped off a forum for disputed tempo labels, the dataset may contain many tracks that are especially hard to annotate for humans. And if not difficult for humans to annotate, it is conceivable that the tracks are particularly hard for algorithms to analyze. Lastly, if neither humans nor algorithms fail, perhaps some of the scraped annotations are simply wrong.

In this paper we investigate why tempo estimation systems perform so poorly for *GiantSteps Tempo*. To this end, we conducted a large, crowdsourced experiment to collect new tempo data for *GiantSteps Tempo* from human participants. The experiment is described in detail in Section 2. The data is analyzed in Section 3 and used to create a new ground-truth. This ground-truth is then compared to the original ground-truth and used to evaluate two recent algorithms. The results are discussed in Section 4. Finally, in Section 5, we summarize our findings and draw conclusions.

2. EXPERIMENT

In order to generate a new ground-truth for the *GiantSteps Tempo* dataset, we set up a web-based experiment in which

¹ <http://www.beatport.com/>, an online music store



© Hendrik Schreiber, Meinard Müller. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: Hendrik Schreiber, Meinard Müller. “A Crowdsourced Experiment for Tempo Estimation of Electronic Dance Music”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

² <https://www.native-instruments.com/en/products/traktor/dj-software/traktor-pro-2/>

³ <http://www.mixvibes.com/cross-dj-software-mac-pc/>

⁴ More benchmark results are available at <http://www.cp.jku.at/datasets/giantsteps/>

we asked participants to tap along to audio excerpts using their keyboard or touchscreen. The user interface for this experiment is depicted in Figure 1. Since most tracks from the dataset are 2 min long and tapping for the full duration is difficult, we split each track into half-overlapping 30 s segments. Out of the 664 tracks we created 4,640 such segments (in most cases 7 per track). To measure tempo, it is not important for tap and beat to occur at the same time. In contrast to experiments for beat tracking, phase shifts, input method latencies, or anticipatory early tapping—known as *negative mean asynchrony* (NMA)—are irrelevant, as long as they stay constant (see [11] for an overview of tapping and [3,5] for beat tracking). Therefore participants were asked to tap along to randomly chosen segments *as steadily as possible*, over the entire duration of 30 s without skipping beats. To encourage steady tapping, the user interface gave immediate feedback in the form of the mean tempo μ in BPM, the median tempo *med* in BPM, the standard deviation of the *inter-tap-intervals* (ITI) σ in milliseconds, as well as textual messages and emojis (Figure 1). When calculating the standard deviation, the first three taps were ignored, as those are typically of low quality (users have to find their way into the groove). When the standard deviation σ stayed very low, smiles, thumbs up and textual praise were shown. When σ climbed above a certain threshold, the user was shown sad faces and messages like “Did you miss a beat? Try to tap more steadily.” To prevent low quality submissions, users were only allowed to proceed to the next track, once four conditions were met:

1. 20 or more taps
2. Taps cover at least 15 s
3. ITI standard deviation: $\sigma < 50$ ms
4. Median tempo: $50 \leq \text{med} \leq 210$ BPM

While the first three conditions were not explicitly communicated, the instructions made participants aware that the target tempo lies between 50 and 210 BPM. Once all four conditions were met, a large red bar turned green and the *Next* button became enabled. For situations in which the user was not able to fulfill all conditions, the user interface offered a *No Beat* checkbox. Once checked, it allowed users to bypass the quality check and proceed to the next song. It must be noted that there is a tradeoff between encouraging participants to tap well (i.e. steadily) and a bias towards stable tempi. We opted for this design for two reasons. 1) tempo in EDM is usually is very steady [2, 7]. 2) the bias is limited to individual tapping sessions at the segment level, i.e. we can still detect tempo stability problems on the track level by aggregating segment level annotations.

Participants were recruited from two distinct groups: Academics and people interested in the consumer-level music library management system *beaTunes*⁵. We refer to the former group as *academics* and the latter as *beaTunes*. While members of the *academics* group were asked to help

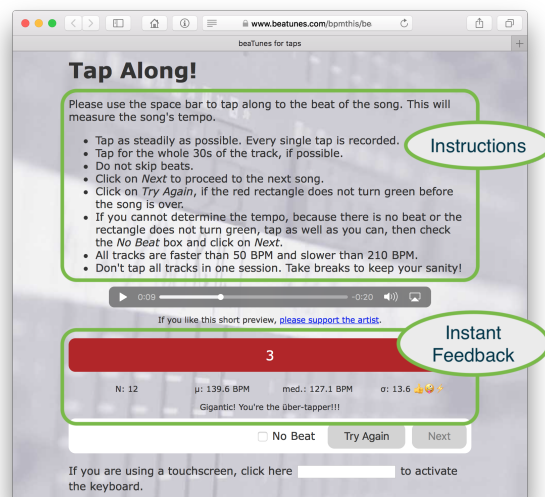


Figure 1: Illustration of the web-based interface used in our experimental user study.

in this experiment via relevant mailing lists without offering any benefits, members of the *beaTunes* group were incentivized by promising a reward license for the *beaTunes* software, if they submitted 110 valid annotations. While it was not explicitly specified what a “valid annotation” is, we attempted to steer people in the right direction using instructions and the instant feedback mechanisms described above (Figure 1).

3. DATA ANALYSIS

Over a period of 21½ months 266 persons participated in the experiment, 217 (81.6%) belonging to *beaTunes* and 49 (18.4%) to *academics*. Together they submitted 18,684 segment annotations (avg = 4.03/segment). We made sure that all segments were annotated at least twice. Since some segments are harder to annotate than others, we monitored submissions and ensured that segments annotated by participants as very different from the original ground-truth—exceeding a tolerance of 4%—were presented to participants more often than others. The vast majority of annotations was submitted by the *beaTunes* group (95.1%). Overall 7.5% of all submissions were marked with *No Beat*. With 7.6% the *No Beat*-rate was slightly higher among members of the *beaTunes* group. Members of *academics* checked *No Beat* only for 5.2% of their submissions. Since the experiment was run in the participant’s web-browser, the browser’s user-agent for each submission was logged by the web-server. Among other information the user-agent contains the name of the participant’s operating system. 17,012 (91.1%) of the submissions were sent from desktop operating systems that are typically connected to a physical keyboard. 1,672 (8.9%) were from mobile operating systems that are usually associated with touchscreens. Participants interested in a reward license, also had to enter name and email. Both datapoints have

⁵ <https://www.beatunes.com/>

Dataset Split	+	-	p -value
$\pm academics$	0.0074	0.0090	$3.11e-29$
$\pm keyboard$	0.0088	0.0095	$9.74e-7$
$beaTunes \pm keyboard$	0.0089	0.0099	$6.71e-10$
$academics \pm keyboard$	0.0074	0.0073	$8.68e-1$

Table 1: Average coefficients of variation \bar{c}_v for dataset splits, *academics* or not, *keyboard* or not, and *keyboard* or not for either *beaTunes* or *academics*. The low p -values indicate a significant difference between the dataset splits.

been removed from the collected data to ensure anonymity.

We analyzed the submitted data to find out whether we can find quality differences between submissions from different participant groups (Section 3.1). Section 3.2 introduces metrics for ambiguity and stability. In Section 3.3, we measure to which extent participants agree on one or multiple tempi for the same segment. Then, in Section 3.4, we take a look at segment annotations aggregated on the track-level. Finally, in Section 3.5, we investigate whether tempo ambiguity is a genre-dependent phenomenon.

3.1 Submission Quality

We wondered how steadily participants tapped and whether some groups of participants tapped more steadily than others. Specifically, are the *beaTunes* submissions as good as the *academics* submissions? We can use the coefficient of variation $c_v = \frac{\sigma}{\mu}$ of each submission’s ITIs as a normalized indicator for how steadily a participant tapped. To remove tapping outliers within a segment, we sort each submission’s ITIs and only keep the central 10 before calculating the c_v . This has the effect of reducing c_v for all submissions. The average c_v for all submissions is $\bar{c}_v = 0.0089$. Assuming a normal distribution, this means that on average 99.7% of all central 10 ITIs lie within $\pm 2.67\%$ ($\equiv 3\sigma$) of their submission’s mean value. Using \bar{c}_v as a measure for the submission quality of different dataset splits, we found that members of *academics* tapped significantly more steadily ($\bar{c}_v = 0.0074$) than members of *beaTunes* ($\bar{c}_v = 0.0090$) (Table 1). To test for significance we used Welch’s t -test. Also, submissions from desktop operating systems that are typically installed on devices connected to a physical keyboard (i.e., no touchscreen) are of significantly higher quality ($\bar{c}_v = 0.0088$) than submissions from devices using iOS or Android as operating system ($\bar{c}_v = 0.0095$). Despite the differences, we found that even the ITIs from the group with the highest \bar{c}_v , i.e., *beaTunes* without keyboard, still lie within only $\pm 2.97\%$ ($\equiv 3\sigma$) of their mean value 99.7% of the time—again assuming a normal distribution. This is well below the tolerance of 4% allowed by *Accuracy1*.

We conclude that the data submitted by *academics* with keyboard is of the highest quality with regard to tempo stability, but find that the data submitted by members of *beaTunes* without keyboard is still acceptable, because the difference in \bar{c}_v is not very large. This may be a direct result of the experiment’s design which did not permit participants to submit highly irregular taps.

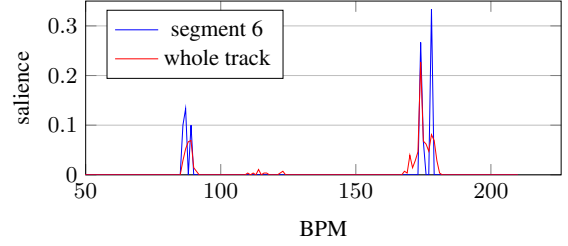


Figure 2: Tempo salience distribution for segment 6 of track ‘Neoteric D&B Mix’ by Polex (Beatport id 4397469). Measured values are: $P(T_{\text{track}}) = 4$, $P(T_{\text{seg6}}) = 2$, $A(T_{\text{track}}) = 0.30$, $A(T_{\text{seg6}}) = 0.40$, and $JSD = 0.24$.

3.2 Tempo Distribution Metrics

How steadily participants tapped does not say anything about whether they tapped along to the true tempo. But since the purpose of the experiment is to create a new ground-truth, we cannot easily verify submissions for correctness. What we can do though, is to measure annotator (dis)agreement both for a segment and for all segments belonging to the same track. To this end, we define some metrics based on tapped tempo distributions. To create such a tapped tempo distribution for a segment, we combine the 10 central ITIs from each of its submissions in a histogram T with a bin width of 1 BPM and then normalize so that $\sum_{i=1}^n T(x_i) = 1$, with n as the number of bins and x_i as the corresponding BPM values. For T we define local peaks as the highest non-zero $T(x_i)$ for all intervals $[x_i - 5, x_i + 5]$. This may include very small peaks. We interpret the BPM values x_i of the histogram’s local peaks as the perceptually strongest tempi and their heights equivalent to their saliences. Per-track tempo distributions are created simply by averaging the 7 segment histograms belonging to a given track. For an example, please see Figure 2.

As a first, very simple indicator for annotator disagreement, we define $P(T)$ as the number of histogram peaks we find in a given tempo distribution T . A high peak count for a single segment $P(T_{\text{seg}})$ indicates annotator disagreement for that segment. This is not necessarily true for the peak count for a track $P(T_{\text{track}})$, since it may also be a sign of tempo instability, i.e., tempo changes or no-beat-sections. Because the peak count P does not say anything about the peaks’ height or salience, it is a relatively crude measure. Therefore we define as second metric the salience ratio between the most salient and the second most salient peak as a measure for ambiguity. More formally, if s_1 is the salience of the highest peak and s_2 the salience of the second highest peak, then the ambiguity $A(T)$ is defined as:

$$A(T) := \begin{cases} 1, & \text{for } P(T) = 0 \\ 0, & \text{for } P(T) = 1 \\ s_2/s_1, & \text{for } P(T) > 1 \end{cases} \quad (1)$$

A value close to 0 indicates low and a value close to 1

high ambiguity. This definition is inspired by McKinney et al. [8] approach to ambiguity, but not identical. Just like P , we can use A for both segment and track tempo distributions. Again, for tracks we cannot be sure of the ambiguity's source.

Finally, we introduce a third metric that focuses more on tempo instability within tracks. Obvious indicators for instabilities are large differences between the tempo distributions of segments belonging to one track. Since we create tapped tempo distributions for each segment in a way that lets us interpret them as probability distributions, we can use the Jensen-Shannon Divergence (JSD) for this purpose, which is based on the Shannon entropy H . With the JSD we measure the difference between the tempo distribution's entropy for the whole track and the average of the individual segment tempo distributions' entropies.

$$H(T) := - \sum_{i=1}^n T(x_i) \log_b T(x_i) \quad (2)$$

$$\text{JSD}(T_1, \dots, T_m) := H\left(\sum_{j=1}^m \frac{1}{m} T_j\right) - \sum_{j=1}^m \frac{1}{m} H(T_j) \quad (3)$$

To allow an easy interpretation of JSD-values, we choose an unusual base for the entropy's logarithm. By setting $b = n$ in (2), we ensure that $0 \leq \text{JSD} \leq 1$. This means, that a JSD-value near 0 indicates a small difference between the tempo distributions for a track's segments. Correspondingly, a JSD-value closer to 1 means that the tempo distributions of a track's segments are very different. To avoid detecting small tempo changes due to annotator disagreements, we convert the segment tempo distributions T to a bin width of 10 BPM before calculating JSD.

3.3 Segment Annotator Agreement

How much do participants agree on a tempo for a given segment? Recall that we have 4,640 segments (and 18,684 annotations for these segments) coming from 664 tracks. As depicted in Figure 3 top, the submissions for more than half the segments (2,500 or 53.9%) have just one peak, i.e., $P(T_{\text{seg}}) = 1$. For 1,514 or 32.6% of all segments we were able to find two peaks, indicating some ambiguity. For 432 segments (9.3%) we found 3 peaks and for 184 segments (4.0%) 4 peaks or more. 10 segments have no peak at all, because they have been marked as *No Beat* in all their submissions. When interpreting these numbers one has to keep in mind that some segments have been annotated by very few participants (Figure 3 bottom). To give an example, while the segments annotated with one peak are based on 3.64 submissions on average, the segments annotated with 6 peaks are annotated with 9.42 submissions per segment. This reflects the fact that we presented difficult segments to participants more often, but could also be caused by increased variability introduced by a higher number of submissions. Because submissions marked as *No Beat* do not show up in this overview unless all submissions for a segment were *No Beat*, we counted the segments for which a majority of submissions were marked with *No Beat*. That was the case for 118 segments (2.5%).

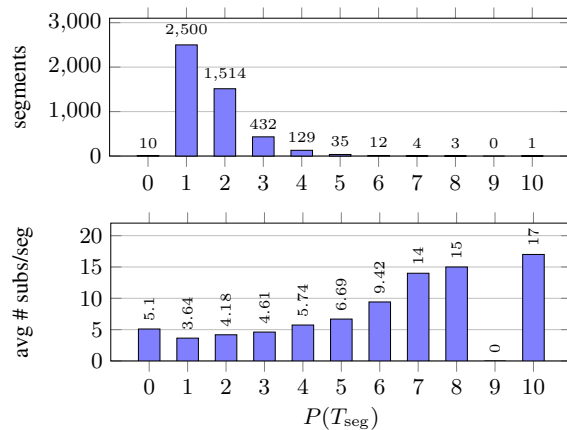


Figure 3: (top) Segments per peak count. (bottom) Average number of submissions per segment by peak count.

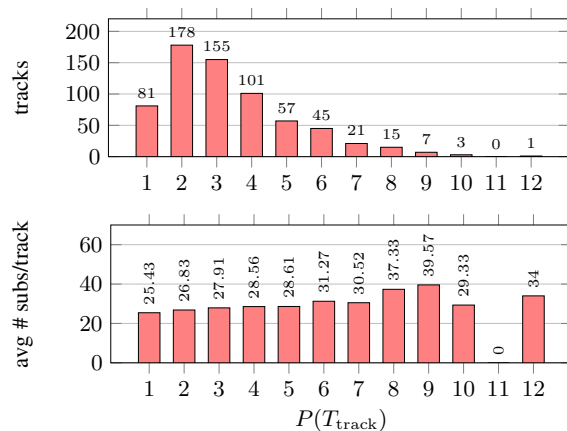


Figure 4: (top) Tracks per peak count. (bottom) Average number of submissions per track by peak count.

As mentioned in Section 3.2, the peak count does not say anything about the peaks' height or salience and is therefore a relatively crude measure. We found that the average ambiguity for all segments is $A(T_{\text{seg}}) = 0.25$ (with standard deviation $\sigma = 0.32$), meaning that on average the highest peak is 4 times more salient than the second highest peak. In other words, we can often observe a peak that is much more salient than others. At the same time, there may also be a second peak with considerable salience.

3.4 Track Annotator Agreement

Just like for the segments, we looked at the number of tracks per peak count. We found only 81 tracks (12.2%) with one peak and 582 tracks (87.8%) with two or more peaks (Figure 4 top). The largest group among the multi-peak tracks are tracks with two peaks (178 or 26.8%). These numbers are much more reliable than the segment peak counts as they are based on at least 25 submissions per track (Figure 4 bottom). Compared to the segments' peak counts we see a larger proportion of tracks with more than one peak. But this does not necessarily mean that the ambiguity A is much higher than for the segments, because

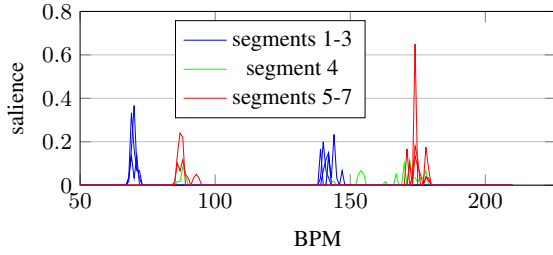


Figure 5: Tempo salience distributions for segments of the track ‘Rude Boy feat. Omar LinX Union Vocal Mix’ by Zeds Dead (Beatport id 1728723). The track’s tempo changes in segment 4, leading to four distinct peaks. With $JSD = 0.44$ its Jensen-Shannon divergence is high.

peak counts do not account for salience and even small local peaks are counted. In fact, we measured an average ambiguity of $\overline{A(T_{track})} = 0.26$ (with standard deviation $\sigma = 0.27$)—almost the same average as for the segments. Therefore we attribute the shift towards more peaks to the much higher number of submissions per item and possible tempo instabilities in the tracks themselves. By tempo instability we mean for example a tempo change in the middle of the track, a quiet section, or no beat at all. Any of these cases inherently lead to more peaks. A typical example for a track with a tempo change is shown in Figure 5.

In an attempt to quantify tempo instabilities in the submissions we calculated the JSD introduced in Section 3.2. The histogram in Figure 6 shows the distribution of tracks per JSD interval with a bin width of 0.05. The average divergence for the whole dataset is $\mu_{JSD} = 0.15$, the standard deviation is $\sigma_{JSD} = 0.11$. To test whether a high JSD correlates with tempo instabilities, we considered all tracks with $JSD > \mu_{JSD} + 2\sigma_{JSD} = 0.375$, resulting in 39 tracks. Performing an informal listening test on these tracks revealed that 3 had no beat, 10 contained a tempo change (e.g. Figure 5), 7 had sections that felt half as fast as other sections (metrical ambiguity), 8 contained larger sections with no discernible beat, 9 were difficult to tap, and 2 had a stable tempo through the whole track. From this result one may conclude that a high JSD is connected to tempo instabilities, but it may also just indicate that a track is difficult to tap. Nevertheless, using JSD helped us find tracks in the *GiantSteps Tempo* dataset that exhibit tempo stability issues. Since 2.5% of the segments were annotated most often with *No Beat*, we wondered whether any tracks have a majority of segments that have predominantly been annotated with *No Beat*, hinting at the absence of not just a local beat (e.g., a sound effect or a silent section), but the lack of a global beat. This is true for 6 tracks, i.e., 0.9% of the dataset. All 6 of them are among the 39 tracks with very high JSD and either have no beat, are very difficult to tap or contain large sections without a beat.

3.5 Ambiguity by Genre

We wondered whether we can confirm findings by McKinney and Moelants [9] that the amount of tempo ambi-

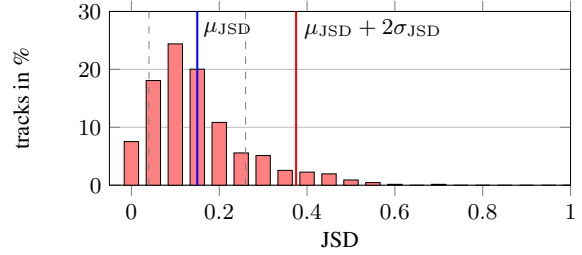


Figure 6: Distribution of tracks in the dataset per JSD interval with a bin width of 0.05. The blue line shows μ_{JSD} and the red line shows $\mu_{JSD} + 2\sigma_{JSD}$.

Genre	$\overline{A(T_{seg})}$	$\overline{A(T_{track})}$
all	0.25	0.26
techno	0.12	0.10
trance	0.17	0.12
drum-and-bass	0.37	0.39
electronica	0.36	0.38
dubstep	0.35	0.43

Table 2: Average ambiguity for the top 5 genres.

guity depends on the genre or musical style. To ensure meaningful results, we considered only the 5 most often occurring genres in the dataset with 54 or more tracks each. We found that the genres techno and trance do not seem to be very affected by ambiguity. More than 65% of their segments are annotated with just one peak. In contrast to that, fewer than 38% of all segments in the genres drum-and-bass, dubstep, and electronica are annotated with just one peak (Figure 7 top). A similar picture presents itself when looking at the average segment ambiguity $\overline{A(T_{seg})}$. As shown in Table 2, it is 0.12 for techno segments and thus much lower than the overall average of 0.25. The same is true for trance (0.17). Contrary to that, the ambiguity values for drum-and-bass (0.37), electronica (0.36) and dubstep (0.35) are all well above the average. We found similar relations for peak counts on the track level (Figure 7 bottom) and the average track ambiguity $\overline{A(T_{track})}$ (Table 2). This strongly supports McKinney and Moelants’ finding that tapped tempo ambiguity is genre-dependent. Perhaps it is even an inherent property.

4. EVALUATION

The tempo histograms for tracks can easily be turned into single tempo per track or two tempi+salience labels. This provides us the opportunity to evaluate the original ground-truth for the *GiantSteps Tempo* dataset by treating it like an algorithm. Since the original annotations are single tempo per track only, we are using *Accuracy1* and *Accuracy2* as metrics. To obtain one tempo value per track from a distribution, we are using just the tempo value with the highest salience. The three tracks without a beat have been removed. We refer to these new annotations as *GSNew* and to the original ones as *GSOrig*. Figure 8 shows the accuracy results for the comparison of *GSOrig* with *GSNew* and reveals a large discrepancy between the two. Only 81.5

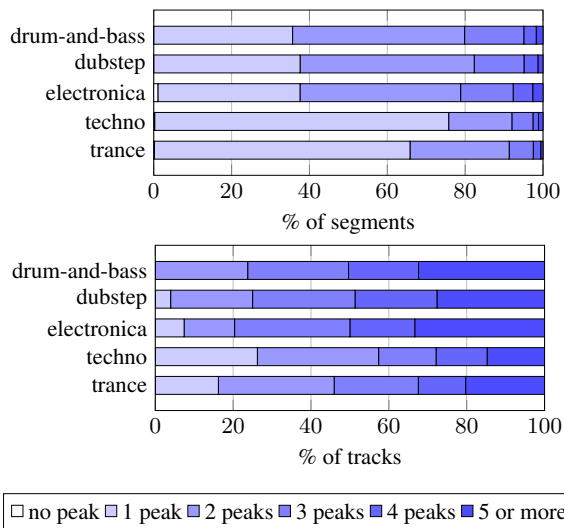


Figure 7: Percentage of segments (top) and tracks (bottom) with a given number of peaks by genre. Drum-and-bass, dubstep, and electronica suffer much more from tapped tempo ambiguity than techno and trance.

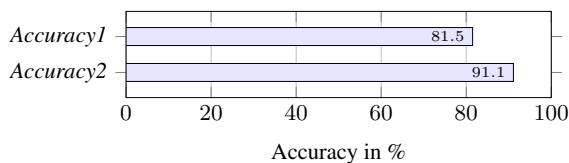


Figure 8: Accuracies measured when comparing GSNew with GSOrig.

of the labels match when using *Accuracy1*, and only 91.1% match when using *Accuracy2*.

Coming back to the original motivation for this paper—the poor performance of tempo estimation systems for *GiantSteps Tempo*—we evaluated the two state-of-the-art algorithms *schreiber* [12] and *böck* [1] with both the old and the new annotations. The algorithms were chosen for their proven performance and conceptual dissimilarity. While *schreiber* implements a conventional onset detection approach followed by an error correction procedure, *böck*’s core consists of a bidirectional long short-term memory (BLSTM) recurrent neural network. Despite their conceptual differences, both algorithms reach considerably higher accuracy values when tested against GSNew (Figure 9). *Accuracy1* increases for *böck* by 5.9 pp (58.9% to 64.8%) and for *schreiber* by 7.1 pp (63.1% to 70.2%). *Accuracy2* shows similar increases, 7.6 pp (86.4% to 94.0%) for *böck* and 6.5 pp (88.7% to 95.2%) for *schreiber*. Remarkably, both *böck* and *schreiber* reach higher *Accuracy2* values for GSNew than the original annotations reached, when compared with GSNew. The increased results for GSNew are much more in line with values reported for other tempo datasets. We therefore believe that this increase and the discrepancy between GSOrig and GSNew are hardly coincidences, but strong indicators for incorrect annotations in GSOrig.

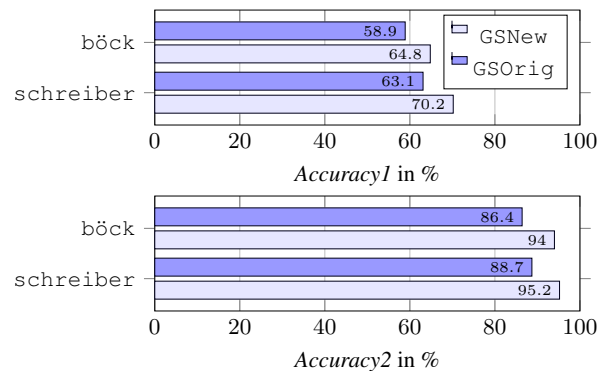


Figure 9: Accuracies for the algorithms *böck* and *schreiber* measured against both GSOrig and GSNew.

5. DISCUSSION AND CONCLUSIONS

In this paper we described a crowdsourced experiment for tempo estimation. We collected 18,684 tapped annotations from 266 participants for electronic dance music (EDM) tracks contained in the *GiantSteps Tempo* dataset. To analyze the data, we used multiple metrics and found that half of the annotated segments and more than half of the tracks exhibit some degree of tempo ambiguity, which may either stem from annotator disagreement or from intra-track tempo instability. This refutes the assumption that it is always easy to determine a single global tempo for EDM. We were able to identify tracks with no tempo at all, no-beat-sections or tempo changes, which raises questions about the suitability of parts of the dataset for the global tempo estimation task. Furthermore, we provided additional evidence for genre-dependent tempo ambiguity. Based on the user-submitted data we derived the new annotations GSNew. The relatively low agreement with the original annotations GSOrig indicates that one of the two ground-truths contains incorrect annotations for up to 8.9% of the tracks (ignoring octave errors). We re-evaluated two recent tempo estimation algorithms against both ground-truths and measured considerably higher accuracies when testing against GSNew. This leads us to the following conclusions: GSOrig contains incorrectly annotated tracks as well as tracks that are not suitable for the global tempo estimation task. The accuracy of state-of-the-art tempo estimation systems is considerably higher than previously thought. And last but not least, as a community, we have to get better at evaluating tempo algorithms in the sense that we need verified, high quality datasets that represent reality with tempo distributions instead of single value annotations. If we cannot accurately measure progress, we have no way of knowing when the task is done.

Datasets

All data is available at http://www.tagtraum.com/tempo_estimation.html.

Acknowledgments

The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits IIS. Meinard Müller is supported by the German Research Foundation (DFG MU 2686/11-1).

6. REFERENCES

- [1] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 625–631, Málaga, Spain, 2015.
- [2] Mark J. Butler. *Unlocking the Groove: Rhythm, Meter, and Musical Design in Electronic Dance Music*. Profiles in popular music. Indiana University Press, 2006.
- [3] Olmo Cornelis, Joren Six, Andre Holzapfel, and Marc Leman. Evaluation and recommendation of pulse and tempo annotation in ethnic music. *Journal of New Music Research*, 42(2):131–149, 2013.
- [4] Fabien Gouyon, Anssi P. Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, 2006.
- [5] Andre Holzapfel, Matthew EP Davies, José R Zapata, João Lobato Oliveira, and Fabien Gouyon. Selective sampling for beat tracking evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(9):2539–2548, 2012.
- [6] Peter Knees, Ángel Faraldo, Perfecto Herrera, Richard Vogl, Sebastian Böck, Florian Hörschläger, and Mickael Le Goff. Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 364–370, Málaga, Spain, October 2015.
- [7] Michaelangelo Matos. Electronic dance music. Encyclopædia Britannica <https://www.britannica.com/art/electronic-dance-music>, last checked 6/9/2018, August 2015.
- [8] Martin F McKinney and Dirk Moelants. Deviations from the resonance theory of tempo induction. In *Proc. Conference on Interdisciplinary Musicology*, Graz, Austria, 2004.
- [9] Martin F McKinney and Dirk Moelants. Ambiguity in tempo perception: What draws listeners to different metrical levels? *Music Perception: An Interdisciplinary Journal*, 24(2):155–166, 2006.
- [10] Geoffroy Peeters and Joachim Flocon-Cholet. Perceptual tempo estimation using GMM-regression. In *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies (MIRUM)*, pages 45–50, New York, NY, USA, 2012. ACM.
- [11] Bruno H. Repp. Sensorimotor synchronization: A review of the tapping literature. *Psychonomic Bulletin & Review*, 12(6):969–992, 2005.
- [12] Hendrik Schreiber and Meinard Müller. A post-processing procedure for improving music tempo estimates using supervised learning. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 235–242, Suzhou, China, October 2017.
- [13] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

COMPUTATIONAL CORPUS ANALYSIS: A CASE STUDY ON JAZZ SOLOS

Christof Weiß¹

Stefan Balke¹

Jakob Abeßer²

Meinard Müller¹

¹ International Audio Laboratories Erlangen, Germany

² Semantic Music Technologies Group, Fraunhofer IDMT, Ilmenau, Germany

christof.weiss@audiolabs-erlangen.de

ABSTRACT

For musicological studies on large corpora, the compilation of suitable data constitutes a time-consuming step. In particular, this is true for high-quality symbolic representations that are generated manually in a tedious process. A recent study on Western classical music has shown that musical phenomena such as the evolution of tonal complexity over history can also be analyzed on the basis of audio recordings. As our first contribution, we transfer this corpus analysis method to jazz music using the Weimar Jazz Database, which contains high-level symbolic transcriptions of jazz solos along with the audio recordings. Second, we investigate the influence of the input representation type on the corpus-level observations. In our experiments, all representation types led to qualitatively similar results. We conclude that audio recordings can build a reasonable basis for conducting such type of corpus analysis.

1. INTRODUCTION

Characterized by keywords such as *systematic musicology* or *computational music analysis*, quantitative and data-driven methods have recently gained importance within musicology. As one central benefit, computational methods enable corpus-based studies on a large scale. Several studies have been conducted recently for different music genres including pop music [13], jazz [1, 6, 9], and Western classical music [2, 17, 21, 24], and also in the field of ethnomusicology [14, 16, 19]. For conducting such corpus studies, a number of different aspects are important. Besides methodological questions such as the musical characteristics under investigation (e. g., melodic, harmonic, or rhythmic aspects), also the way these characteristics are measured, evaluated, and presented matters. Moreover, the corpus itself plays a crucial role. Beyond its size and composition, the representation of the music data constitutes an important aspect. For example, the data can be given as a symbolic transcription [9, 16], as a graphical score [17], or as an audio recording [6, 13, 18].



© Christof Weiß, Stefan Balke, Jakob Abeßer, Meinard Müller. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Christof Weiß, Stefan Balke, Jakob Abeßer, Meinard Müller. “Computational Corpus Analysis: A Case Study on Jazz Solos”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

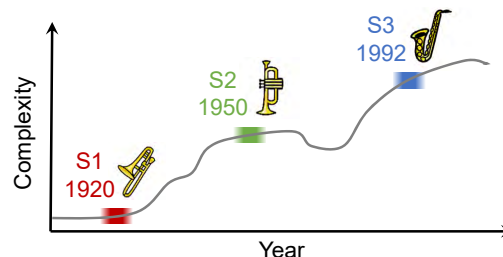


Figure 1. Procedure for mapping feature values from individual solos onto the timeline using the recording years.

In this paper, we investigate the influence of the music representation type on the corpus analysis results. For this purpose, we present a case study for jazz music using the solos contained in the *Weimar Jazz Database* [15]. As an example for a corpus analysis, we investigate the tonal complexity of the jazz solos using a measure introduced in [22]. Inspired by recent work on pop [13] and classical music [21], we apply a visualization technique where quantitative descriptors for individual pieces are mapped onto a timeline as shown in Figure 1. The resulting *evolution curves* [21] allow for studying the evolution of musical phenomena (here: tonal complexity) over history.

As input data for this study, we compare different representations of the jazz solos including a high-quality symbolic transcription of the solo melody as well as the full mix audio recording of the solo section. Furthermore, we investigate intermediate representations, which rely on signal processing techniques [3, 4, 7, 8] for enhancing the presence of the solo instrument and for suppressing accompanying instruments and audio-specific artifacts. Specifically, we consider the approaches proposed in [4, 7]. Though the music representations—as well as the derived features—exhibit a different behavior on the *piece level*, our experiments show that on the *corpus level*, results are qualitatively similar for audio-based procedures and for analyses based on high-quality symbolic transcriptions. Our findings encourage to perform corpus studies on the basis of audio recordings. This opens up new ways for musicological research since audio recordings are available easily without an extensive transcription or annotation process that often needs to be done manually.

The remainder of this paper is structured as follows. First, we describe our music scenario and sketch some musicological hypotheses (Section 2). Second, we detail on

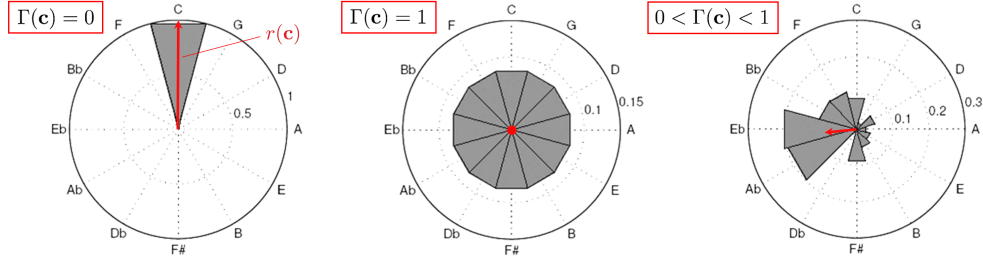


Figure 2. Complexity measure Γ based on the circle of fifths. Values for a sparse chroma vector (left), a flat chroma vector (middle), and a more realistic chroma vector (right) are shown. The red arrows denote the resultant vectors.

our tonal complexity measure and explain its musical implications (Section 3). We then describe the different representations and signal processing techniques we use in this study (Section 4). Next, we describe our corpus analysis strategy and present the experimental results (Section 5). Finally, we discuss the implications of our findings.

2. JAZZ SCENARIO

Within the scope of jazz music, the *Weimar Jazz Database* (WJD) with its 456 manually generated transcriptions of famous jazz solos constitutes a unique dataset [15]. A major benefit of the WJD lies in its clean annotations of the solo melody (fundamental frequency, F_0), which create a controlled environment for systematic experiments. The data served as basis for a number of musicological studies, which mainly focus on performance analysis [1, 6, 9].

Besides the rhythmical aspects of the solos [6] and the melodic phrasing [9], also the played pitch material (scales) can be of musicological interest. In our experiments, we consider this dimension by measuring the tonal complexity of the pitches played by the soloist. We expect to find a lower tonal complexity for solos from the *Chicago Jazz* era (1920s), compared to, for instance, *Bebop* solos from the 1950s. However, there might be some outliers in each period. For example, Chet Baker’s intimate solos will probably obtain lower complexity values than Clifford Brown’s solos—although both perform in the same period.

3. MEASURING TONAL COMPLEXITY

The analysis of music complexity has been an important task within MIR research in the past years. Streich [20] tackled multiple dimensions of this notion denoted as acoustic, timbral, rhythmic, and tonal complexity. Concerning tonality, many studies [5, 12, 20] focus on sequential complexity aspects such as the complexity of chord progressions [5]. As opposed to this, chroma-based complexity measures were introduced in [22], which locally describe the pitch class distribution without explicitly capturing transitional characteristics. Despite their simplicity, these features have shown a high correspondence to an intuitive understanding of music complexity over the course of an individual piece [22]. Beyond that, they have turned out to be useful for classifying music recordings according to style categories [23]. Averaging such complexity features over many pieces provides meaningful and stable

results, which has been shown in a large-scale study of musical evolution in classical music [21]. As one contribution, we transfer this concept to jazz music and show that complexity features also yield meaningful results for this scenario. In contrast to [21], the WJD scenario provides data in different representations (see Section 4), whose influence we want to investigate. Moreover, we have detailed metadata such as the recording year of each solo.

The complexity measures introduced in [22, 23] describe statistical properties of an underlying normalized chroma distribution. Flat distributions result in high complexity values while sharp distributions result in low ones. In [23], several different measures are introduced for this purpose such as entropy-, sparsity-, and flatness-based quantities. Here, we restrict ourselves to one feature that additionally accounts for the tonal relationship of the prominent pitch classes. Following [23], we now summarize the definition of this measure $\Gamma : \mathbb{R}^{12} \rightarrow [0, 1]$. Let $\mathbf{c} = (c_0, c_1, \dots, c_{11})^T \in \mathbb{R}^{12}$ denote a chroma vector with positive entries ($c_n \geq 0$) normalized with respect to the ℓ^1 -norm ($\sum_{n=0}^{11} c_n = 1$). The entries c_n with $n \in [0 : 11]$ indicate the salience of the twelve pitch classes C, C^\sharp , ..., B, respectively. Because of octave invariance, the features show a cyclic behavior so that a transposition in pitch leads to a circular shift.

For computing the complexity $\Gamma(\mathbf{c}) \in [0, 1]$ of a chroma vector $\mathbf{c} \in \mathbb{R}^{12}$, we first re-sort the chroma values to an ordering of perfect fifth intervals (7 semitones) resulting in the vector $\mathbf{c}^{\text{fifth}}$ defined by:

$$c_n^{\text{fifth}} = c_{(n \cdot 7) \bmod 12}. \quad (1)$$

Based on the reordered vector $\mathbf{c}^{\text{fifth}}$, we define the resultant vector $\mathbf{r}(\mathbf{c})$ with a length of

$$r(\mathbf{c}) = \left| \frac{1}{N} \sum_{n=0}^{N-1} c_n^{\text{fifth}} \exp\left(\frac{2\pi i n}{12}\right) \right|. \quad (2)$$

Then, the complexity $\Gamma(\mathbf{c})$ is defined as:

$$\Gamma(\mathbf{c}) = \sqrt{1 - r(\mathbf{c})}. \quad (3)$$

This measure corresponds to the angular deviation and describes the spread of the pitch classes around the circle of fifths. Figure 2 shows the complexity feature and the resultant vector $\mathbf{r}(\mathbf{c})$ (in red) for three input chroma vectors \mathbf{c} . For a sparse vector (left), the complexity is minimal ($\Gamma(\mathbf{c}) = 0$). For a flat vector (middle), we obtain maximal complexity ($\Gamma(\mathbf{c}) = 1$).

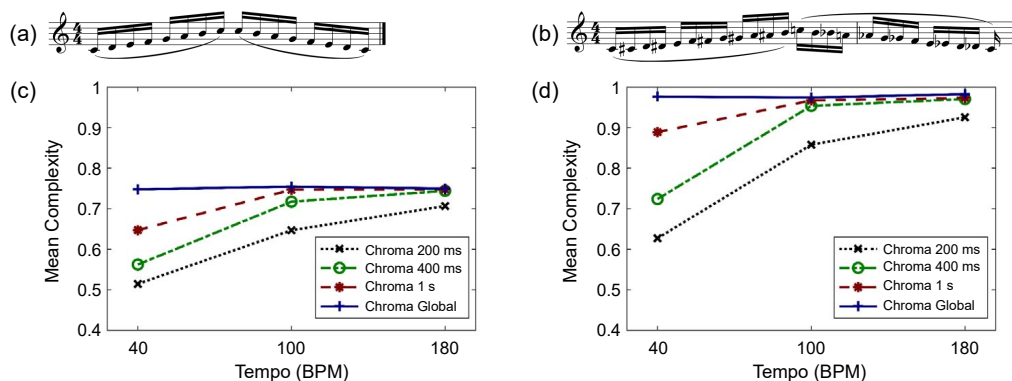


Figure 3. Complexity values for musical scales in several tempi, computed with different window lengths. (a) Diatonic scale. (b) Chromatic scale. (c) Complexity values for the diatonic scale. (d) Complexity values for the chromatic scale.

In this paper, we compute complexity features for jazz solos. Relying on chroma features of the full audio recordings, the features describe the complexity of the overall tonal content—comprising the sounding pitches of the solo instrument as well as the accompanying instruments (e. g., piano, double bass, drums). Since noise-like sounds such as drum hits contribute in an approximately equal fashion to each of the twelve chroma values, this results in an overall increase of complexity. As opposed to the full mix recording, a symbolic transcription of the solo only captures the pitches played by the solo instrument. Since we deal with monophonic solo instruments (mainly saxophone, trumpet, trombone), there is only one non-zero pitch class at a time. Using a small window length (fine-grained resolution) for the chroma features, this results in low complexity values. As soon as we use a larger window length—e. g., by smoothing over several chroma frames—the complexity features are computed from local pitch class histograms and, thus, show mostly non-zero values in case that different pitch classes are played within the analysis interval. Hereby, the feature values depend on the number of pitch classes played but also, on their tonal relationship. Playing many fifth-related pitch classes—such as a diatonic scale—yields a distribution pointing towards a specific direction in the circle of fifths and, thus, results in a rather low complexity value (see Figure 3a and c). For a chromatic scale, in contrast, pitch classes all over the circle of fifths contribute equally resulting in a high complexity value (Figure 3b and d).

Beyond the pitch classes and their relationship, the duration of the notes has a crucial effect on the complexity features. To illustrate this effect, we show in Figure 3 complexity values for scales played in different tempi. For this experiment, we synthesized a diatonic scale and a chromatic scale from music notation software using a saxophone sound. From the generated audio, we computed chroma features in different temporal resolutions. On the basis of these chroma features, we calculated complexity values and averaged these over the full segment. Figures 3c and d show the resulting complexity features for different resolutions and playing tempi. In a higher tempo, more pitch classes are sounding within a window lead-

ing to higher complexity. The absolute complexity values also depend on the analysis window length. The four curves in Figures 3c and d refer to different chroma window lengths of 200 ms, 400 ms, 1 s, and a global chroma histogram, respectively. With larger smoothing windows, we obtain higher complexity values. Using global chroma statistics, the complexity is practically independent of the tempo since it always relies on the same pitch class distribution. For a monophonic input signal, our feature captures the tonal complexity of the melody pitches rather than describing a “melodic complexity,” which usually accounts for further properties such as direction, jumps, melodic intervals. etc. Despite these simplifications, our complexity feature mostly behaves in a musically meaningful way.

4. INPUT DATA AND PRE-PROCESSING

The complexity feature $\Gamma(c)$ can be computed from different pitch class representations. This enables us to compare the feature values for different representation types. Besides symbolic representations with explicit pitch information, we can also use audio-based chromagrams.¹ In our experiments (Section 5), we investigate how the choice of the input representation influences the complexity features (see Figure 4).

Beyond the symbolic transcription (Figure 4a) created in the Jazzomat project (manual F0 annotation of the solo melody), we consider the full mix audio signal (d), as well as two modified audio versions (b, c). For this, we use signal processing methods to suppress components that might affect our harmony analysis. One such method is harmonic-percussive-residual separation (HPRS) [7], which is an extension of the technique presented by Fitzgerald [8]. HPRS aims to decompose a given audio recording into a harmonic component, a percussive component, and a residual component. The residual component captures portions of the audio recording which are neither of harmonic, nor percussive nature, e. g., noise-like signals such as applause or the breathy component of the saxophone sound. For enhancing the tonal parts of the jazz

¹ In contrast to our complexity measure, high-level measures as presented in [5, 20] often require pre-processing steps that involve challenging tasks such as automatic transcription.

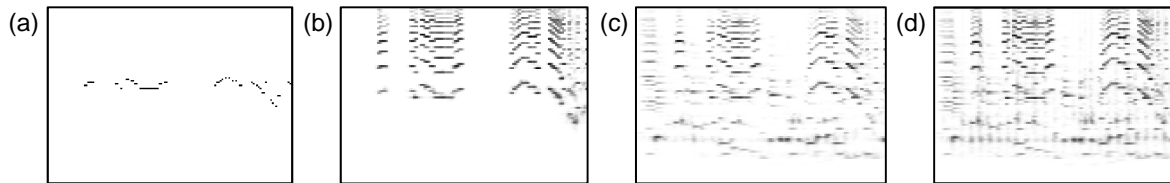


Figure 4. Log-frequency representations of Dexter Gordon’s solo from “Society Red” (excerpt of 14 seconds). (a) Symbolic transcription. (b) Source-separated melody (score-informed). (c) Harmonic–Percussive–Residual separation, harmonic part. (d) Full audio mix.

recordings, we use HPRS and throw away both the residual and the percussive components (see Figure 4c).

Beyond this straight-forward separation, we also use a more sophisticated decomposition. Hereby, we try to extract the solo signal from the full mix via source separation. Similar to previous approaches [10, 11], we make use of score information (F0 trajectories) for the separation into solo instrument and backing track [4]. The fundamental frequency trajectory of the solo instrument is used to construct time-variant masks that follow in principle a comb filter structure covering a certain number of the instrument’s partials. Several post-processing steps ensure that the bandwidth of the single comb spikes covers the range of the individual partials and that interference from transient sound events is attenuated. Due to the score information, the resulting solo track is almost free of background instruments (see Figure 4b). Only signals that overlap the solo instrument’s partials (such as broad-band percussive components) are sometimes perceivable.

From the four representations, we compute pitch class features by summing up energies from different octaves. A comparison of the representation types is interesting since they fundamentally differ from each other in several respects. First, the representations capture different musical parts. Symbolic transcription (a) and source-separated signal (b) only contain the solo instrument, whereas in the other representations, accompaniment is also present. Second, the transcription (a) only contains the fundamental frequency while all other representations also capture overtones. Third, transcription (a) and HPRS-enhancement (c) only capture harmonic information while the separated solo (b) and the full mix (d) also contain residual and percussive components. We will now study how these properties influence a large-scale analysis on the corpus level.

5. CORPUS ANALYSIS

Based on the different types of music representations discussed above, we conduct studies on the tonal complexity of the WJD solos. Inspired by [21], we compute *evolution curves* mapping solo-wise complexity features onto a historical timeline. For this purpose, we use the annotated recording year of each solo. To smooth the curve, we use a soft mapping employing a Gaussian window of size 11 years. Thus, a solo contributes not only to its concrete recording year but also, to a smaller degree, to each 5 years

before and after.² With this technique, the jazz solos distribute over the timeline as shown in Figure 5a. At about 1955, more than 15 solos contribute on average. Around 1932 (beginning of our timeline) and 2002 (end), there are hardly any solos. This means that a solo contributing to these years has a higher influence on the evolution curve.

To investigate the complexity of the jazz solos, we first analyze each solo individually by computing complexity features in one resolution using the global chroma histogram. In Figure 5b and c, we show these complexity values of individual solos as gray crosses. Figure 5b relies on the symbolic transcription and Figure 5c on the HPRS-enhanced audio (harmonic part). We find a broad range of values for most years. Except for the first 15 years, which do not show very high complexity values, there are solos of diverse complexity at all times. Thus, it is hard to find general structures and trends for individual solos. The overall distribution, however, is similar in both figures.

To analyze this in more detail, we now compute evolution curves. We project the feature value of every piece onto the timeline using the procedure described above. The complexity curves are normalized regarding the number of solos contributing to each year.³ Figure 5 shows the resulting curves as blue lines. As an additional cue, we compute for the most frequent soloists the complexity value averaged over all their solos, respectively. For each soloist, we plot the average value as horizontal bar from the first to the last solo’s recording year. Overall, we observe a slight increase of complexity over the years. The first major increase develops towards the year 1948, where soloists such as Don Byas and Charlie Parker start to contribute. Around the 1960s, we find soloists such as Chet Baker with lower complexity as well as Clifford Brown or Joe Henderson with higher complexity. During the 1970s, there is a major drop, before the complexity again increases towards the early 2000s (David Liebman or Michael Brecker).

Comparing the two curves in Figures 5b and c, we observe that their shape is similar—only the overall scale of the complexity values differs slightly. Most of the prominent changes in complexity can be observed on the basis of both representations—such as the increase around 1945, the drop in the 1970s, and even smaller changes such as the local minimum around 1950. The peak and drop after

² The window is normalized so that the total weight of a solo summed up over all 11 years is one.

³ We sum up the weighted complexity values for all pieces and divide by the number of solos per year as shown in Figure 5a.

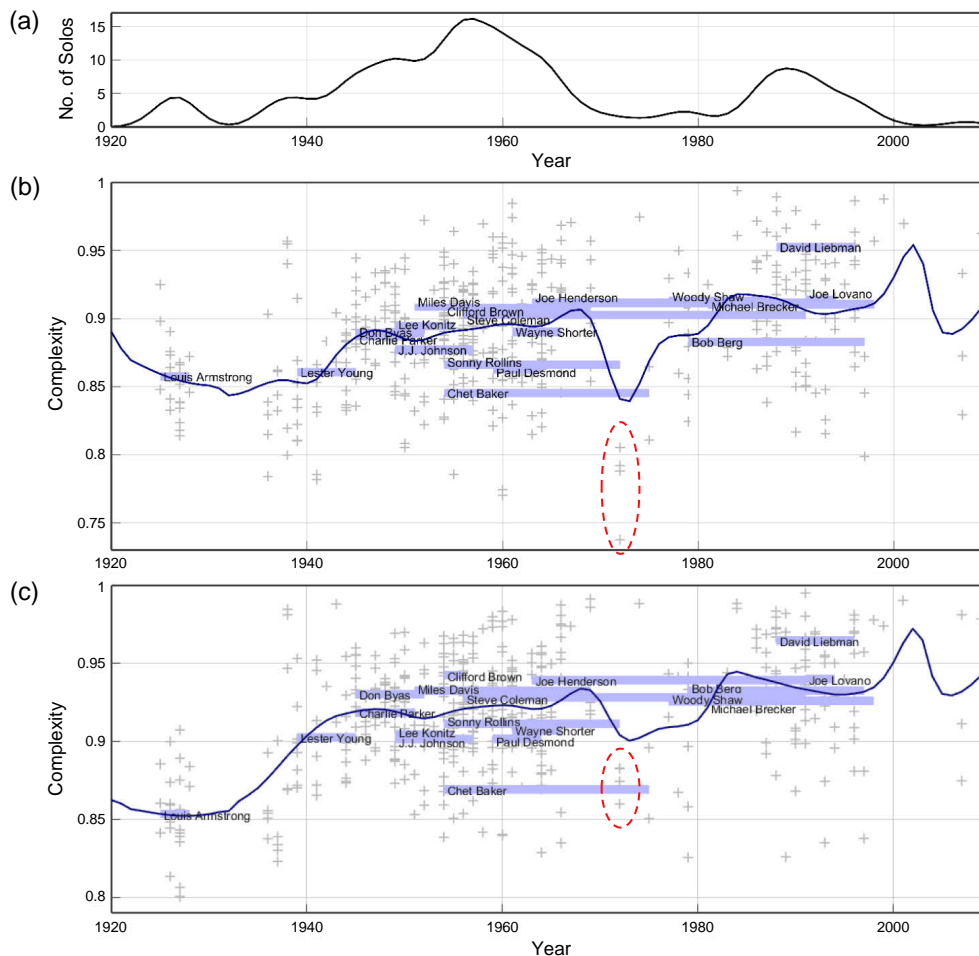


Figure 5. (a) Average number of solos per year contained in the dataset. Evolution curve and artist means based on (b) symbolic transcriptions and (c) harmonic component of audio recordings.

2000 behave very similar. However, we have to take these results with care since only a few solos contribute here. We also find differences between the two plots. For the first years, there are higher values in the symbolic-based plot (b). Here, we could identify several solos with longer silence between the phrases such as Kid Ory’s solo in “Gut Bucket Blues.” In the symbolic representation, these silent frames are all zero which results in a flat chroma vector (high complexity). This leads to a higher overall complexity of these solos.⁴ In the audio-based chromagrams, there are accompanying instruments playing between the phrases, which leads to a lower complexity here. At the year 1972, the drop in Figure 5b is more extreme than in Figure 5c. Looking at the individual solos, we can identify four points of low complexity here. These are solos by Sonny Rollins, two of them played within the piece “Playin’ in the Yard” and two within “The Everywhere Calypso” (red ellipses in Figure 5). Indeed, these solos are constructed of only a few pitch classes with clear tonal relationships. For “Playin’ in the Yard”, Rollins only uses a pentatonic scale for both solos whereas the solos in “The

⁴ Removing silent frames before computing features suppresses this effect to some degree but, at the same time, produces artificial pitch combinations within local windows (phrases squeezed together).

Everywhere Calypso” mainly consist of major scales and broken major triads (arpeggios). In the symbolic representation, these structures lead to a low complexity since there is no accompaniment. In the audio, the background instruments dampen this drop. Overall, we can observe several interesting structures that might be relevant for jazz history. These phenomena could be observed in a similar way on the basis of both symbolic and audio representations.

To test these observations in more detail, we now consider four different feature resolutions (see Section 4). Beyond the influence of the representation type, we want to test how signal processing technologies for suppressing background instruments affect the evolution curves. Figure 6 summarizes this experiment’s results. In addition to the global complexity, we use chroma window lengths of 20 s, 1 s, and 400 ms. Looking at the vertical axes, we observe different absolute ranges. For the symbolic transcription (Figure 6a), the values of Γ for the global complexity (blue curve) lie in the interval $[0.84, 0.95]$. In contrast, the audio-based complexity curve (d) lies in the range $[0.93, 0.98]$. The enhanced audio versions are located between these extremes. HPRS-enhancement (c) leads to a curve with values in $[0.85, 0.97]$. Score-informed source separation (b) produces a global complexity curve ranging

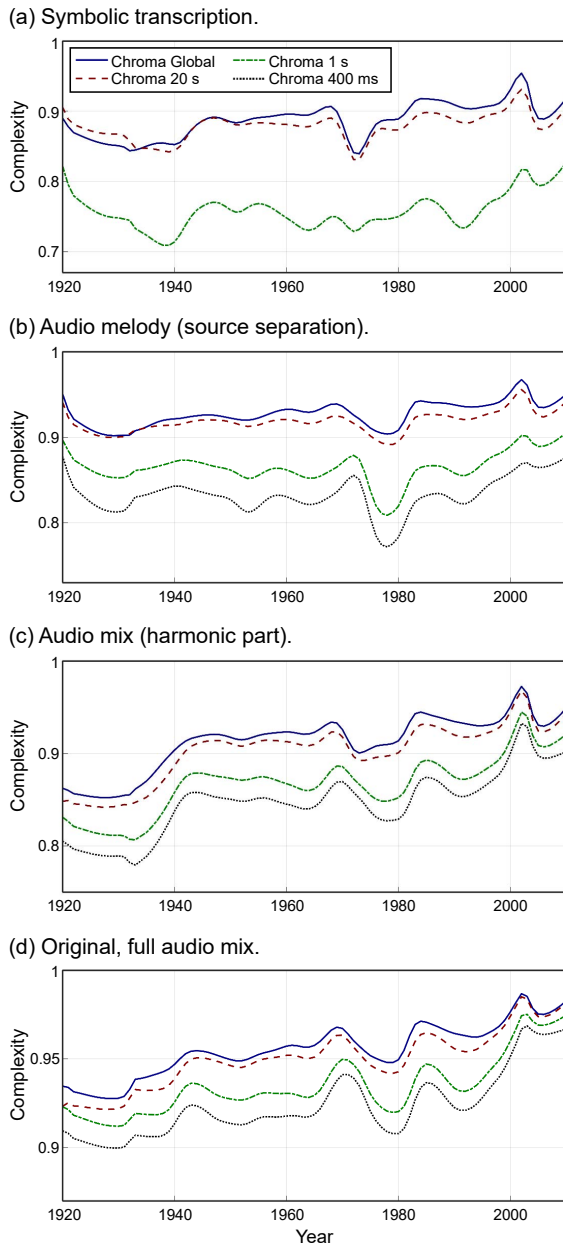


Figure 6. Evolution curve based on (a) symbolic transcription, (b) source-separated melody (score-informed), (c) harmonic part of audio (HPRS), (d) full audio mix.

in $[0.9, 0.96]$. Interestingly, these values are higher than in the HPRS-enhanced case (c). It seems that the percussive components or other artifacts remaining in the separated signal affect the complexity more than the harmonic parts of the background instruments do. For other window lengths, the behavior is similar. Only for the symbolic transcription (a), the smaller window lengths of 1 s and 400 ms (outside the plotting range) behave differently. Since the transcription of a monophonic solo exhibits only one non-zero pitch class at a time, this is no surprise—our complexity feature drops to zero then. With larger window lengths, we capture several pitch classes simultaneously leading to higher complexity.

Apart from the different ranges, we find only minor differences between the curves. As in Figure 5, the first years show higher complexity for the symbolic transcription (a) but also for the source-separated audio (b). As mentioned above, this is due to the long silence gaps between solo phrases. Considering the background instruments leads to a lower complexity and thus, stabilizes the analysis in some way. We also discover a special behavior at the year 1972. The symbolic-based curve (a) shows a sharp drop here stemming from Rollins’ solos discussed above. This drop is weakened when using source separation (b) or the full mix (d) but it can still be observed in the HPRS-enhanced analysis (c). We conclude that not the background instrument but the percussive and residual components of the melody instrument (and possible overlap signals) eliminate this drop.

Beyond these rather subtle differences, the overall behavior is similar for all curves. In all settings, we observe a major increase around 1940 followed by a slightly increasing plateau between 1945 and 1967. Then, all curves drop, again reach a peak around 1983, and finally rise towards the 2000s. Even detailed structures are preserved throughout all representations such as the small drops around 1950 and 1965, or the curvature during the 1990s. Even for years with a low number of contributing solos where we have to take the results with care, the behavior is stable across representations. These observations show that corpus-level characteristics of the WJD appear in a widely coherent way over all of our experimental settings.

6. DISCUSSION

From our experiments, we conclude that meaningful corpus analyses can be performed on the basis of different music representations. Though our evolution curves for the WJD vary in their absolute range, general trends can be observed for all representations. Some audio-related artifacts in the analysis could be suppressed with standard signal processing tools such as harmonic-percussive separation. In contrast, using a high-quality score-informed technology for melody separation did not necessarily improve the results regarding audio-specific artifacts. It seems that timbral characteristics have a greater effect on the curves than the presence of background instruments. Quite the contrary, the presence of background instruments could even stabilize the analysis since it helps to suppress extreme complexity values when the solo instrument is silent. The high similarity between symbolic- and audio-based analyses lets us conclude that in a typical jazz scenario, the solo instrument is prominent enough in the full mix for analyzing some interesting solo characteristics directly from audio. This is an encouraging finding since audio-based studies can be scaled up to a large number of solos easily—in contrast to the time-consuming procedure needed for creating the WJD melody annotations. Since the deviations between our curves occurred in regions with low solo coverage, we suppose that in a large-scale corpus study, individual outliers are suppressed even better leading to more reliable results.

Acknowledgments: This work has been supported by the German Research Foundation (MU 2686/10-1, MU 2686/11-1, AB 675/2-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS. The authors want to thank the members of the Jazzomat research project led by Martin Pfeiderer for creating the WJD.

7. REFERENCES

- [1] Jakob Abeßer, Klaus Frieler, Estefanía Cano, Martin Pfeiderer, and Wolf-Georg Zaddach. Score-informed analysis of tuning, intonation, pitch modulation, and dynamics in jazz solos. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):168–177, 2017.
- [2] Héctor G. Bellmann. *Categorization of Tonal Music Style: A Quantitative Investigation*. PhD thesis, Griffith University, Brisbane, Australia, 2012.
- [3] Juan J. Bosch, Rachel M. Bittner, Justin Salamon, and Emilia Gómez. A comparison of melody extraction methods based on source-filter modelling. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 571–577, New York City, USA, 2016.
- [4] Estefanía Cano, Gerald Schuller, and Christian Dittmar. Pitch-informed solo and accompaniment separation towards its use in music education applications. *EURASIP Journal on Advances in Signal Processing*, 2014(23), 2014.
- [5] Bruno Di Giorgi, Simon Dixon, Massimiliano Zanoni, and Augusto Sarti. A data-driven model of tonal chord sequence complexity. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(11):2237–2250, 2017.
- [6] Christian Dittmar, Martin Pfeiderer, Stefan Balke, and Meinard Müller. A swingogram representation for tracking micro-rhythmic variation in jazz performances. *Journal of New Music Research*, 47(2):97–113, 2018.
- [7] Jonathan Driedger, Meinard Müller, and Sascha Disch. Extending harmonic-percussive separation of audio signals. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 611–616, Taipei, Taiwan, 2014.
- [8] Derry FitzGerald. Harmonic/percussive separation using median filtering. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 246–253, Graz, Austria, 2010.
- [9] Klaus Frieler, Martin Pfeiderer, Wolf-Georg Zaddach, and Jakob Abeßer. Midlevel analysis of monophonic jazz solos: A new approach to the study of improvisation. *Musicae Scientiae*, 20(2):143–162, 2016.
- [10] Romain Hennequin, Bertrand David, and Roland Badeau. Score informed audio source separation using a parametric model of non-negative spectrogram. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 45–48, Prague, Czech Republic, 2011.
- [11] Antoine Liutkus, Jean-Louis Durrieu, Laurent Daudet, and Gaël Richard. An overview of informed audio source separation. In *Proceedings of the International Workshop on Image and Audio Analysis for Multimedia Interactive Services (WIAMIS)*, pages 93–96, Paris, France, 2013.
- [12] Matthias Mauch and Mark Levy. Structural change on multiple time scales as a correlate of musical complexity. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 489–494, Miami, Florida, USA, 2011.
- [13] Matthias Mauch, Robert M. MacCallum, Mark Levy, and Armand M. Leroi. The evolution of popular music: USA 1960–2010. *Royal Society Open Science*, 2(5), 2015.
- [14] Maria Panteli, Emmanouil Benetos, and Simon Dixon. A review of manual and computational approaches for the study of world music corpora. *Journal of New Music Research*, 47(2):176–189, 2018.
- [15] Martin Pfeiderer, Klaus Frieler, Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart. *Inside the Jazzomat*. Schott Campus, Mainz, Germany, 2017.
- [16] Marcelo Rodríguez-López and Anja Volk. Symbolic segmentation: A corpus-based analysis of melodic phrases. In *Sound, Music, and Motion*, pages 548–557, Cham, 2014. Springer.
- [17] Pablo H. Rodríguez Zivic, Favio Shifres, and Guillermo A. Cecchi. Perceptual basis of evolving Western musical styles. *Proceedings of the National Academy of Sciences*, 110(24):10034–10038, 2013.
- [18] Xavier Serra. Creating research corpora for the computational study of music: The case of the CompMusic project. In *Proceedings of the AES International Conference on Semantic Audio*, London, UK, 2014.
- [19] Ajay Srinivasamurthy, Andre Holzapfel, Kaus-tuv Kanti Ganguli, and Xavier Serra. Aspects of tempo and rhythmic elaboration in hindustani music: A corpus study. *Frontiers in Digital Humanities*, 4(20), 2017.
- [20] Sebastian Streich. *Music Complexity a Multi-Faceted Description of Audio Content*. PhD thesis, University Pompeu Fabra, Barcelona, Spain, 2007.
- [21] Christof Weiß, Matthias Mauch, Simon Dixon, and Meinard Müller. Investigating style evolution of Western classical music: A computational approach. *Musicae Scientiae*, 2018.

- [22] Christof Weiß and Meinard Müller. Quantifying and visualizing tonal complexity. In *Proceedings of the Conference on Interdisciplinary Musicology (CIM)*, pages 184–187, Berlin, Germany, 2014.
- [23] Christof Weiß and Meinard Müller. Tonal complexity features for style classification of classical music. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 688–692, Brisbane, Australia, 2015.
- [24] Christopher Wm. White. *Some Statistical Properties of Tonality, 1650-1900*. PhD thesis, Yale University, New Haven, Connecticut, USA, 2013.

CONTROLLED VOCABULARIES FOR MUSIC METADATA

Pasquale Lisena¹ Konstantin Todorov² Cécile Cecconi³ Françoise Leresche⁴
Isabelle Canno⁵ Frédéric Puyrenier⁴ Martine Voisin⁵ Thierry Le Meur³ Raphaël Troncy¹

¹ EURECOM, Sophia Antipolis, France ² LIRMM, University of Montpellier, CNRS, France

³ Philharmonie de Paris, France ⁴ Bibliothèque nationale de France ⁵ Radio France

lisena@eurecom.fr, todorov@lirmm.fr, ccecconi@cite-musique.fr

ABSTRACT

We present a set of music-specific controlled vocabularies, formalized using Semantic Web languages, describing topics like musical genres, keys, or medium of performance. We have collected a number of existing vocabularies in various formats, converted them to SKOS and performed the interconnection of their equivalent terms. In addition, novel vocabularies, not available online before, have been designed by an editorial team. Next to multilingual labels and definitions, we provide hierarchical relations as well as links to external resources. We also show the application of those vocabularies for the production of vector embeddings, allowing for the calculation of distances between keys or between instruments.

1. INTRODUCTION

Describing music is an activity that involves an important number of terms coming from domain-specific glossaries. In addition to the cross-domain concept of genre, we can mention musical keys, instruments or catalogues of compositions. Libraries and musical institutions have different practices for describing this kind of information. In the best case, they make use of thesauri that are often available in different incompatible formats, and that can be either internally defined or standardised by larger communities such as the International Association of Musical Libraries (IAML). In other cases, this information is codified in free text fields, delegating to the editors the responsibility of following the living practice about syntax and lexical form.

The new attitude for sharing the knowledge beyond the institutional and national borders—embodied by international consortia like IAML or in projects like European [7] and OpenGlam [4]—brings its effect also on the music domain. Accordingly, Semantic Web technologies have gained a central role in music representation, that has reached the Linked Open Data world. A second consequence is the request for a change in the previously described current practices towards the adoption of publicly available controlled vocabularies. The use of vocabularies

opens up different possibilities, like the definition of labels in different languages or of alternate lemmata in the same language (i.e. the French terms “ut majeur” and “do majeur” which both refer to the key of *C major*). Different kinds of relationships between terms can be defined and it is possible to define a hierarchy between them (for example, “violin” is a narrower concept with respect to “string”) which can produce, as benefit, a more powerful advanced search for the final user. Previous research demonstrated how an RDF (for Resource Description Framework) structure helps reasoning engines to discover links between different levels in the hierarchy of instruments [8].

Publishing Semantic Web vocabularies is not new in the field of music. The Musical Instruments Museum Online (MIMO)¹ published the biggest taxonomy of musical instrument in RDF, as result of the contribution of institutions and universities all over the world. The librarian practice draws on the UNIMARC² thesauri of musical forms (genres) and medium of performance standardised by IAML. Historically adopted by librarians worldwide, these thesauri have recently been published in the Web of Data, marking the growing interest in this technological environment. The French National Library (BnF) relies on an authority vocabulary in RDF for subject headings called RAMEAU,³ containing a list of labels for entities of encyclopedic interest which includes also music genres and instruments. A musical key vocabulary is published as side resource of the MusicOntology [14], consisting in a list of English labelled concepts, with some additional information—like the mode (*major/minor*), the tonic, etc.—, without any links describing semantic connections between them.

On the one hand, a large number of thesauri cover few well-defined categories (genres and medium of performance), making the reconciliation of data coming from different sources difficult, also because of the different formats of these thesauri. A reconciliation that would add a broader and deeper nomenclature has a benefit, increasing both the number of elements and alternate labels. On the other hand, a large set of concepts—handled so far through error-prone free-text—is asking for standardisation in specialised vocabularies.



© Lisena et al. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Lisena et al. “Controlled Vocabularies for Music Metadata”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ <http://www.mimo-db.eu/>

² They are commonly named after the UNIMARC standard for librarian records, in which they are widely used.

³ Répertoire d'autorité-matière encyclopédique et alphabétique unifié: <http://rameau.bnf.fr/>

This paper presents a set of controlled vocabularies for the description of the music information as Linked Open Data, extending and finalising related work in [10]. This research has the primary goal of the interconnection of music information datasets, building bridges between existing vocabularies and providing tools for the automatic matching. The final aim consists in the contribution to the achievement of a global music knowledge graph [1] in the Web of Data, looking at all the applications that semantically structured data have in the music information retrieval and recommendation field [2, 12, 13, 18]. In Section 2, we present the complete set of vocabularies, giving detailed information about their content. The process of realisation, collection and interlinking is described in Section 3, while we present applications, such as embeddings and literal dereferencing in Section 4. Finally, we conclude and outline some and future work in Section 5.

2. MUSIC VOCABULARIES

A controlled vocabulary is a thematic thesaurus of entities. In the Semantic Web, each term is identified with a URI. The *Simple Knowledge Organization System (SKOS)* [11] have been chosen as format because of its capability of defining preferred and alternate labels in each language, relationships between terms, comments and notes for describing the entity and help the annotation activity. In the case of the vocabulary of *Catalogues of works*, the used ontology is the RDF version of *Metadata Object Description Schema (MODS)* [17], that suits the need of defining identifiers, publication date, subject, etc.

Each vocabulary fulfils a set of requirements, including multilingualism, open and public access, presence of definitions. It must also be suitable for different contexts of use and conceptual models of musical information, which is guaranteed by the presence in the editorial team of experts from different types of cultural institutions (libraries, radio broadcasting networks, concert halls).

The vocabularies are all available in a triple store server, which provides a SPARQL endpoint⁴ for requesting the data in different formats like RDF, JSON, csv, etc. Alternatively, the vocabularies can be explored by a web browser starting from <http://data.doremus.org/vocabularies/>. The server enables the HTTP dereferencing of URIs: this means that a web browser pointing to the URI of a specific concept (e.g. <http://data.doremus.org/vocabulary/derivation/medley>) will land on a page containing its human-readable description, showing all its properties. The triple store includes also music data that use these vocabularies, so that the definition and the usage of a concept in musical works can be appreciated as part of the same knowledge base. Finally, an RDF version in Turtle format is available on GitHub.⁵

Each vocabulary is licensed for free distribution, fol-

lowing a Creative Commons Attribution 4.0 license,⁶ and it is open to the community for any kind of contribution.

We collected, implemented and published 18 controlled vocabularies belonging to 7 different families, containing more than 9500 distinct concepts and involving 26 different languages or dialects. In the following paragraphs, we describe the content of those vocabularies, subdivided in two groups.

2.1 Collection of interlinked vocabularies

This group includes vocabularies that were already available in the Web of Data, in the community or internally to a specific institution. When two or more vocabularies share the same high-level topic—e.g. the musical genre—we call that group *family*. In order to interconnect the different knowledge sources, an alignment process is needed for discovering when terms coming from vocabularies belonging to the same family refers to the same concept. This process will be detailed in Section 3.3.

Musical genres. This family includes vocabularies about the genre of a musical work. By genre, we mean the main categories by which we describe the works, like rock, lirica, funk, opera, gospel, polka, jazz, including genres of world music. The term genre is very broad and also includes musical “forms” that gained in the centuries their own genre definition like symphony, concerto, sonata.

We collected, republished as SKOS and interlinked the following vocabularies:

- **IAML**, 607 concepts, multilingual. This list, largely adopted in librarian environments, was available as a set of labels and codes, in some cases with definitions or editorial notes. We converted this big vocabulary to SKOS from different sources (librarian tabular data, online HTML version). Afterwards, a SKOS version⁷ has been published by IFLA (International Federation of Library Associations), which is however less rich than ours in terms of alternate labels. We provide owl:sameAs links from our vocabulary to the IFLA version.
- **RAMEAU**, 654 concepts, French, hierarchised. It is published as Linked Data by the French National Library (BnF). We extracted from this large nomenclature the part related to musical genres.
- **Diabolo**, 629 concepts, French, hierarchised. It is the set of labels used in the disc catalogue of Radio France (RF). It also includes some skos:related links, e.g. between *spiritual* and *gospel*.
- **Itema3**, 40 concepts, French. It is used in the technical documentation of the concert archive of RF.
- **Itema3-MusDoc**, 172 concepts, French. It is used in the musical documentation of the concert archive of RF.
- **Redomi**, 297 concepts, French, hierarchised. It is used in the musical work documentation of RF.

Medium of performance. Any instrument able to produce sounds can be considered as a medium of perfor-

⁴ <http://data.doremus.org/sparql>

⁵ <https://github.com/DOREMUS-ANR/knowledge-base/tree/master/vocabularies>

⁶ <https://creativecommons.org/licenses/by/4.0/>

⁷ <http://iflstandards.info/ns/unimarc/terms/fom/>

mance or MoP. In this family of vocabularies, we can find musical instruments coming from different cultures (western, oriental, African, Indian, etc.), the voices in different ranges (soprano, alto, etc.), aside from group of instruments (orchestras, ensembles) and voices (choirs).

We collected, republished as SKOS and interlinked the following vocabularies:

- **MIMO**, 2480 concepts, multilingual, hierarchised. The *Musical Instrument Museum Online* comes from the joint international effort of different music institutions and museum. Despite being the most complete vocabulary of instruments, it does not include voices. MIMO is publicly available as Linked Data.⁸
- **IAML**, 419 concepts, multilingual, hierarchised. Despite its smaller granularity, this vocabulary has a good coverage for voices and groups. Like for the homonym genre vocabulary, also in this case an official version from IFLA is online,⁹ less rich both with respect to the languages covered and to the number of concepts (392).
- **RAMEAU**, 876 concepts, French, hierarchised. As in the genre case, we selected the part related to MoP.
- **Diabolo**, 2117 concepts, French, hierarchised. It is the set of labels used in the disc catalogue of RF. For ethnic or traditional instrument, it includes also the reference to the relative geographic area.
- **Itéma3**, 314 concepts, French. It is used in the documentation of the concert archive of RF.
- **Redomi**, 179 concepts, French, hierarchised. It is used in the musical work documentation of RF.

2.2 New vocabularies

This section presents vocabularies for which we did not rely on any previous material, because it was not existing or not suitable for our goals. We designed these vocabularies on the basis of real data coming from institutions, enriched by an editorial process that involved also librarians. Since the work has been conducted in French, the definitions of the terms are so far available only in this language. However, every label has been translated at least in English and Italian in order to facilitate their reuse.

Musical keys. 30 concepts, English, French, Spanish, Italian. This vocabulary contains the set of keys used in western music, labelled with the tone followed from the type of scale (e.g. *C major*). The concept are linked among them by specific properties for keys relationships, like *relative*, *parallel* and *closely related* keys. It contains also sameAs links with the key vocabulary of MusicOntology.

Musical modes. 22 concepts, English, French, Italian, Latin, hierarchised. The word *mode* generally refers to a type of scale, coupled with a set of characteristic melodic behaviours. They are mostly used for describing ancient or medieval music.

Catalogues of works. 152 MODS resources. A *thematic catalogue* or *catalogue of works* is a recognised editorial

list of all known works of a composer. In practice, a classical composition can be univocally identified by the catalogue code and number. For example, *Eine kleine Nachtmusik* is identified with *K 525*, where *K* is the Köchel catalogue of Mozart's work. Each resource contains the information about the catalogue editor and publisher, the language of drafting, the date of publication. The subject artist of each catalogue is disambiguated through the DOREMUS dataset [1, 10].

Types of derivations. 16 concepts, English, French, Italian, Spanish, German, hierarchised. A work can be derived from another by transforming its material into another through orchestration, harmonisation, etc. All these types (with definitions) are collected in this vocabulary.

Functions. 106 concepts, English, French and Italian, hierarchised. A music event—a performance, a composition, a recording, etc.—involves a number of different roles or *functions* like author, performer, conductor, sound engineer, etc. Additional details can also be provided to account for the different kinds of author, like composer, lyricist or arranger. These functions are identified in this vocabulary, together with their definitions.

3. MODELING PROCESS

We detail the modeling process, which is based on an interaction between music metadata experts and automatic data conversion and fusion tools.

3.1 Editorial work

An editorial committee grouping 7 members coming from different backgrounds (library, radio, concert hall) played an important role in the vocabulary modeling. First, existing vocabularies have been inventoried and assessed as candidates for being interlinked on the basis of their completeness and adoption. Next, the committee made choices about which new vocabularies to create and what should be their scope. These choices reflect the aim of producing powerful tools to describe recordings, publications and their contexts of creation, instead of producing exhaustive vocabularies about every aspects of the music. The committee relies on the members experience in music data management practices. The experts had to confront their point of views—necessarily different because depending on the missions of their institutions—until the list of terms, their contexts of use and their definitions were coherent.

First of all, the group had to be consistent with respect to the data available in those institutions. For example, we chose not to publish a rhythmic patterns vocabulary since the data which is available was not created in a musical analysis perspective. Then, the work had to be based upon the team's area of competence. This is one of the reasons why we decided to limit the scope of the musical modes vocabulary to old and European ones only. Listing and describing scales coming from other continents would require an additional work that would largely involve musicologists.

⁸ <http://www.mimo-international.com/>

⁹ <http://iflstandards.info/ns/unimarc/terms/mop/>

We chose first to create the new vocabularies described in Section 2.2. The catalogues of works vocabulary is a special one since it contains only a list of bibliographic references and does not have the structure of a thesaurus. It was established from the titles used by the BnF. The musical keys and musical modes vocabularies were the easiest to model since they contain a small number of well-defined concepts with clear translations. The other ones were more complex to model. A first set of entities was generated on the basis of the initial datasets. Then, the experts had to confront their point of views until the list of terms, their contexts of use and their definitions were coherent. The functions vocabulary was especially complex to define since it had to reconcile very different ways of describing performing art activities: What is the relevant description for “sound engineer”? How to describe the act of improvising? etc.

3.2 Conversion to Semantic Web formats

Two different steps take part in the generation of the vocabularies as RDF graphs.

The first one is a preliminary conversion from spreadsheets or XML files to RDF, using the OpenRefine [6] tool or with specific scripts. The collections of concepts already in the Web of Data (like RAMEAU) have been instead extracted through specific SPARQL queries on an endpoint.

In the second step, additional vocabulary-specific actions are performed. In some cases, hierarchy is inferred on the basis of specific properties and rules (e.g. in the IAML MoP vocabulary, the hierarchy is taken from the letters included in the last part of the URI). All the language tags are normalised in order to follow the ISO 639 standard. Moreover, the indication of the use of Latin script is made explicit for transliterated labels in languages that use different alphabets. In this phase, some interlinking to external datasets is performed, using SPARQL queries (DOREMUS dataset, MusicOntology keys vocabulary) or REST APIs like GeoNames [19].

3.3 Vocabulary Alignments

The sets of vocabularies of musical genres and those of medium of performance, described in Section 2.1, group together a number of well-established or internally used within a given institution reference lists. There is an important overlap between the sets of entities (genres or musical instruments) described across these vocabularies in each of the two categories. For example, the music genre “folk song” is described both in the IAML vocabulary (labelled by the French “chanson populaire” and the English “folk song”) and in the Radio France-hosted Diabolo vocabulary (labelled by “folksong”). The task of vocabulary alignment consists in automatically establishing links of identity between the elements of two vocabularies from the same category. This would allow to discover automatically the equivalence between the two folk-song terms across IAML and Diabolo. Since our vocabularies are described in SKOS, the procedure comes down to discovering and declaring `skos:exactMatch` relations across the terms

of two given vocabularies. In our example, this would result in bounding the IAML and Diabolo identifiers¹⁰ of the folk-song genre in a `skos:exactMatch` relation.

We have proceeded to establish pairwise alignments between the concepts of the vocabularies in each of these two categories (genres and MoP). We have chosen IAML as a target vocabulary for the alignments of the genres-vocabularies, meaning that all remaining genre-vocabularies will be aligned to IAML. This decision is motivated by the fact that this vocabulary is large in size and also largely adopted in the librarian world. In the same line of thought, we have selected MIMO from the MoP family as a target for the alignments. This results in the performance of five pair-wise alignments for each category (all to IAML in the genre category and all to MIMO in the MoP category).

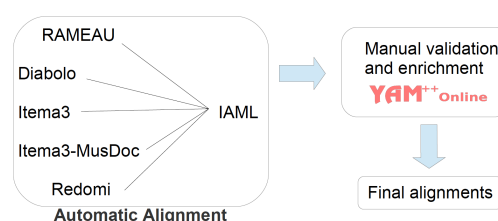


Figure 1. The overall alignment and expert-validation framework. An example with the genres family.

The overall alignment process consists of *automatic alignment* and *manual validation and enrichment* (see Figure 1). For the automatic alignment, we have taken a simplistic string-based approach that relies on a comparison between the labels of SKOS concepts by looking both at preferred and alternative labels, returning in output a confidence score. Note that in many cases, we have language tags associated to the terms. However, there is no consensus among the different vocabulary providers on the language of origin of the terms of interest – in many cases a musical instrument or a genre will be labelled as “French” in one vocabulary and “Italian” in another, although the label originates in, say, Italian in both cases, simply because the Italian word is commonly employed in French. For that reason, we have ignored the language tags when comparing the labels. This process aims to generate a large pool of mapping candidates, ensuring high recall at this step. The alignments are stored in the standard EDOAL format,¹¹ which allows to keep the confidence score of each aligned pair of terms.

In order to guarantee high quality of the produced alignments and to improve precision, the automatically generated mappings are subjected to validation by the librarian experts. In order to facilitate this laborious task, we have developed a web application that allows to assist this process. The application has been conceived as a module of *YAM++ online* [3]—a multi-task web platform for on-

¹⁰ Respectively, <http://data.doremus.org/vocabulary/iaml/genre/fso> and <http://data.doremus.org/vocabulary/diabolo/genre/folksong>

¹¹ <http://alignapi.gforge.inria.fr/edoal.html>

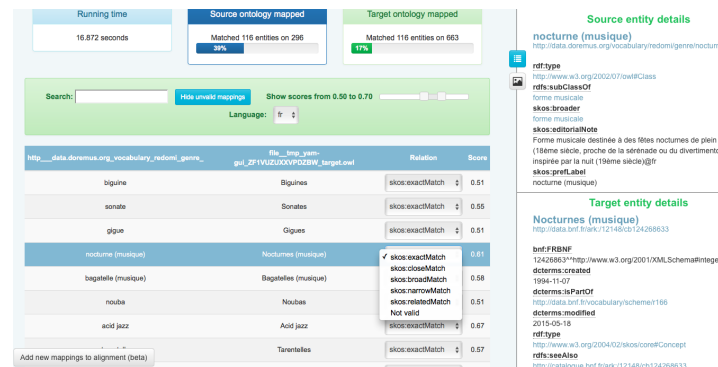


Figure 2. The mapping validator interface.

tology and thesaurus matching and validation,¹² although the **Validator** interface can be seen as a standalone tool. It takes as input a valid EDOAL alignment file, together with its two OWL ontologies or SKOS vocabularies (via an URL or a file path). A list of mappings (pairs of labels of aligned concepts) appears on the main page, together with information about the portions of the vocabularies covered by the alignment (Figure 2, above). A context description of each of the two concepts in each line is displayed (Figure 2, right), containing all alternate labels, as well as the labels (or URIs) of parents and children. The user can take benefit of the confidence score of the previously computed alignments (shown at the right end of each line) for filtering the pairs of concepts by the help of a horizontal cursor. For each concept pair, the expert is given the possibility to modify and select a relation type from a list of SKOS relations, or simply discard the mapping. Experts spotted out and corrected a number of invalid alignments (507 over 1022 for genres and 2039 over 3981 for MoP). In particular for small differences in the label (i.e. plural forms) the role of the human validation is crucial for ensuring quality in the vocabularies.

The expert is given the possibility to manually enrich the proposed alignment by the help of the alignment enrichment environment, accessible via the “Add new mappings” button (Figure 2, bottom left). A new page opens containing the full label lists of the two vocabularies. A key-word search on both lists, including preferred and alternate labels, allows to browse and select manually a pair of concepts and define their relation. The newly defined mappings are added to the initial alignment. Finally, all modifications are added to the alignment file, which can be either saved in the default EDOAL format, or exported in the form of RDF/XML triples.

4. USAGE OF VOCABULARIES

The availability of controlled vocabularies opens up new possibilities that involve the data conversion and usage. In this section, we present recent work that aims to be complementary to the vocabulary publishing, providing further tools and resources to support their effectiveness.

4.1 String2Vocabulary

A common task in what is called *knowledge graph population* (which is the generation of semantic triples starting from differently structured data sources) is the passage from plain text nodes or *literals* to a more representative object node or *entity*. Often, the target of this task consists in a set of vocabularies.

A *string2uri* algorithm – developed in the context of the Datalift platform [15] – performs an automatic mapping of string literals to URIs coming from controlled vocabularies in SKOS. The software reads a RDF graph and searches for exact matches between literal nodes and vocabulary terms.

Some experiences in knowledge base population of classical music data, have shown up some critical points. Often the title of a classical work includes or, even more, consists in the name of an instrument or a key or a genre (e.g. Ravel’s *Bolero*), that should be excluded from the replacement process and be kept as textual literals. Moreover, the complexity itself of this data – involving an important number of properties – in addition to the commonly used file formats (i.e. MARC), has led in the years to a cataloguing practise particularly prone to editorial mistakes. This is the case of musical keys declared as genre, or fields for the opus number that contain actually a catalog number and vice-versa [10].

For these reasons, we adapted the Datalift strategy in a new *String2Vocabulary* open-source library.¹³ The software uses the file name of vocabularies for grouping them in families: *mop-mimo.ttl* and *mop-iaml.ttl* are part of the family *mop*, while *key.ttl* is the sole member of the family *key*. This library accepts a configuration file that assigns a family to a RDF property. For each input graph, it searches for the properties one after the other, retrieving their values. Each value is then compared to all the terms of the vocabulary, until it finds one equal to the value. All variants for a concept label – namely `skos:prefLabel` and `skos:altLabel` – are considered in order to deal with potential differences in naming terms, and both graph values and terms receive a normalisation that has the effect of removing the punctuation, lower-casing the text and decoding it to ASCII. Then, a substitution of that node with

¹² <http://yamplusplus.lirmm.fr>

¹³ <https://github.com/DOREMUS-ANR/string2vocabulary>

the found concept URI is performed.

String2Vocabulary works both with literal values and with entities labelled through `rdfs:label`. In the latter case, the label to be matched against the vocabulary and the whole node – with all its properties – is replaced. For maximising the possibilities of selecting, if it exists, the right concept, two searches are performed in sequence. The first requires that both the given text and language match with the concept ones. If this search fails, a second one requires a match excluding the language information.

As additional feature, the configuration file allows to request the lemmatisation for certain vocabularies. Taking the MoP vocabulary as representative example, three sequential matches are tried: singularising the first word of the label (for matching cases such as “*cornets à pistons*”@fr), singularising the whole label (“*sassofoni contralti*”@it) and leaving the label as is for matching instruments that are always plural (“*cymbals*”@en).

4.2 Music Embeddings

What are the closest keys to *C major*? Is it possible to decide which instrument between the *cello* and the *oboe* is more similar to the *clarinet*? The answer to those questions would provide application in different fields, from musicology studies to the development of specialised recommendation systems. The graph structure of RDF allows to define some kind of distance between two entities, by considering the number of nodes that separate them. Hierarchies and other kind of links between vocabulary terms can be considered for computing this distance.

Node2vec [5] is a state-of-the-art algorithm for computing entity embeddings. The algorithm computes random walks in the graph following the links (edges) between nodes, computing the neighbourhood for each of them. Each edge can have a different *weight*, which affects the probability that it participates to the walk. Through this method, the graph is mapped to a vector space, in which nodes becomes points represented by numeric vectors.

A set of music embeddings for the concepts defined in controlled vocabularies are being produced and published.¹⁴ Two different graphs are considered:

- the graph of vocabularies, which defines structural and semantic connections between entities, such as hierarchies, *sameAs* links, properties in common, specific music properties (i.e. relationships between keys);
- the graph of usage, which includes all the usages of the vocabularies in the DOREMUS dataset. We considered musical works for the genre and the key, castings and performances for MoPs, composition and performance events for functions.

On these two graphs, we computed the embeddings using *node2vec*. We arbitrarily set to the graph of vocabularies a weight 6 times bigger than the graph of usage in order to counterbalance the richly larger number of triples¹⁵ and

¹⁴ <https://github.com/DOREMUS-ANR/music-embeddings>

¹⁵ More than 16 millions triples against around 100.000 ones for the vocabulary graph.

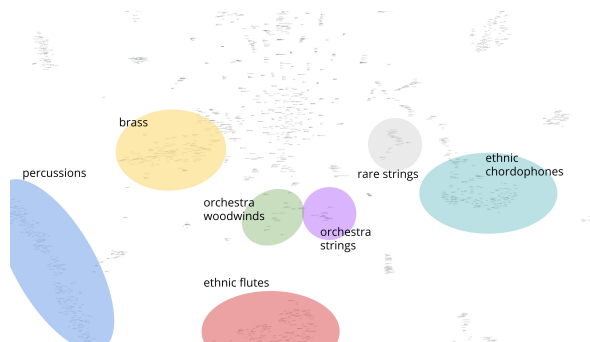


Figure 3. A 2D representation of the vector space of *medium of performance*, with some recognisable clusters.

avoid to nullify the contribution of each one. After a post-processing step that removes all the literals and the extra nodes involved, a L2 normalisation is then applied in order to have values in $\{-1;+1\}$.

In order to appreciate the effectiveness of this strategy, we used t-SNE [16] for visualising the embeddings on a 2D image. As an example, Figure 3¹⁶ shows the vector space of *medium of performance*. By observing the groups of closer entities, we can clearly identifies clusters of instruments. It is interesting to observe that even if the hierarchy of the instrument families is preserved, the usage graph strongly influenced the result, by reflecting the differences of instruments in genres and periods. This is the case of the orchestra instruments group, which puts the *violin* closer to his orchestra colleague *clarinet* than to its 15th-century relative *tromba d’amore*.

Further research is being conducted about the combination of this embedding in more complex ones (artists and works embeddings) in order to compute the similarity between musical entities [9].

5. CONCLUSION

We have presented a set of multilingual vocabularies for the description of music-specific concepts using the Semantic Web framework. Two main contributions consist in the interconnection of already in-use vocabularies of genres and medium of performance and the realisation of previously-unreleased ones. We described our working strategies as an interaction between editors and an automatic system. A dereferencing library and a set of embeddings are presented as side works, allowing to identify application benefits coming from these vocabularies.

Those vocabularies are intended to become references in the field and we strongly encourage their reuse and adoption by the community at large in all their forms. Several additional vocabularies are currently under development, covering concepts like vocal techniques, types of work, type of recording support or partitioning of musical works. Once completed, these vocabularies will be published by following the procedures described in this paper.

¹⁶ A higher-resolution image is available at <https://github.com/DOREMUS-ANR/music-embeddings/tree/master/img>

ACKNOWLEDGMENTS This work has been partially supported by the French National Research Agency (ANR) within the DOREMUS Project, under grant number ANR-14-CE24-0020.

6. REFERENCES

- [1] Manel Achichi, Pasquale Lisena, Konstantin Todorov, Raphaël Troncy, and Jean Delahousse. DOREMUS: A Graph of Linked Musical Works. In *17th International Semantic Web Conference (ISWC)*, Monterey, California, USA, 2018.
- [2] Alo Allik, Florian Thalmann, and Mark Sandler. Musiclynx: Exploring music through artist similarity graphs. In *Companion Proceedings of the The Web Conference 2018*, pages 167–170, Lyon, France, 2018.
- [3] Zohra Bellahsene, Vincent Emonet, Duyhoa Ngo, and Konstantin Todorov. YAM++ Online: A Web Platform for Ontology and Thesaurus Matching and Mapping Validation. In *Extended Semantic Web Conference, P&D*, pages 137–142. Springer, 2017.
- [4] Beat Estermann. “OpenGLAM” in Practice—How Heritage Institutions Appropriate the Notion of Openness. In *20th International Research Society for Public Management Conference (IRSPM)*, pages 13–15, Hong Kong, China, 2016.
- [5] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.
- [6] Kelli Ham. Openrefine (version 2.5). <http://openrefine.org>. free, open-source tool for cleaning and transforming data. *Journal of the Medical Library Association: JMLA*, 101(3):233, 2013.
- [7] Bernhard Haslhofer and Antoine Isaac. data.europeana.eu: The europeana linked open data pilot. *International Conference on Dublin Core and Metadata Applications*, 0:94–104, 2011.
- [8] Sefki Kolozali, Mathieu Barthet, György Fazekas, and Mark B Sandler. Knowledge Representation Issues in Musical Instrument Ontology Design. In *12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 465–470, 2011.
- [9] Pasquale Lisena and Raphaël Troncy. Combining music specific embeddings for computing artist similarity. In *18th International Society for Music Information Retrieval Conference (ISMIR), Late-Breaking Demo Track*, Suzhou, China, 2017.
- [10] Pasquale Lisena, Raphaël Troncy, Konstantin Todorov, and Manel Achichi. Modeling the complexity of music metadata in semantic graphs for exploration and discovery. In *Proceedings of the 4th International Workshop on Digital Libraries for Musicology (DLfM)*, pages 17–24, Shanghai, China, 2017.
- [11] Alistair Miles and José R Pérez-Agüera. SKOS: Simple knowledge organisation for the web. *Cataloging & Classification Quarterly*, 43(3-4):69–83, 2007.
- [12] Sergio Oramas, Oriol Nieto, Mohamed Sordo, and Xavier Serra. A deep multimodal approach for cold-start music recommendation. In *2nd Workshop on Deep Learning for Recommender Systems, at RecSys 2017*, Como, Italy, 2017.
- [13] Sergio Oramas, Vito Claudio Ostuni, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. Sound and music recommendation with knowledge graphs. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8:1–21, 2016.
- [14] Yves Raimond, Samer A. Abdallah, Mark B. Sandler, and Frederick Giasson. The music ontology. In *15th International Conference on Music Information Retrieval (ISMIR)*, pages 417–422, 2007.
- [15] François Scharffe, Ghislain Atemezang, Raphaël Troncy, Fabien Gandon, Serena Villata, Bénédicte Bucher, Fayçal Hamdi, Laurent Bihanic, Gabriel Képéklian, Franck Cotton, et al. Enabling linked-data publication with the datalift platform. In *AAAI workshop on semantic cities*, 2012.
- [16] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
- [17] Melanie Wacker, Jan Ashton, Ann Caldwell, Ray Denenberg, Angela Di Iorio, Rebecca Guenther, Myung-Ja Han, Sally McCallum, Tracy Meehleib, Stefanie Rhle, and Robin Wendler. Metadata Object Description Schema (MODS). Specification, Library of Congress’ Network Development and MARC Standards Office, 2013.
- [18] David M. Weigl and Kevin R. Page. A framework for distributed semantic annotation of musical score: “take it to the bridge!”. In *18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.
- [19] Mark Wick and Bernard Vatant. The geonames geographical database, 2012.

DALI: A LARGE DATASET OF SYNCHRONIZED AUDIO, LYRICS AND NOTES, AUTOMATICALLY CREATED USING TEACHER-STUDENT MACHINE LEARNING PARADIGM

Gabriel Meseguer-Brocal

Alice Cohen-Hadria

Geoffroy Peeters

Ircam Lab, CNRS, Sorbonne Université, Ministère de la Culture, F-75004 Paris, France

`gabriel.meseguerbrocal@ircam.fr, alice.cohenhadria@ircam.fr, geoffroy.peeters@ircam.fr`

ABSTRACT

The goal of this paper is twofold. First, we introduce DALI, a large and rich multimodal dataset containing 5358 audio tracks with their time-aligned vocal melody notes and lyrics at four levels of granularity.

The second goal is to explain our methodology where dataset creation and learning models interact using a teacher-student machine learning paradigm that benefits each other. We start with a set of manual annotations of draft time-aligned lyrics and notes made by non-expert users of Karaoke games. This set comes without audio. Therefore, we need to find the corresponding audio and adapt the annotations to it. To that end, we retrieve audio candidates from the Web. Each candidate is then turned into a singing-voice probability over time using a teacher, a deep convolutional neural network singing-voice detection system (SVD), trained on cleaned data. Comparing the time-aligned lyrics and the singing-voice probability, we detect matches and update the time-alignment lyrics accordingly. From this, we obtain new audio sets. They are then used to train new SVD students used to perform again the above comparison. The process could be repeated iteratively. We show that this allows to progressively improve the performances of our SVD and get better audio-matching and alignment.

1. INTRODUCTION

Singing voice is one of the most important elements in popular music. It combines its two main dimensions: melody and lyrics. Together, they tell stories and convey emotions improving our listening experience. Singing voice is usually the central element around which songs are composed. It adds a linguistic dimension that complements the abstraction of the musical instruments. The relationship between lyrics and music is both *global* (lyrics topics are usually highly related to music genre) and *local* (it con-

nects specific musical parts with a concrete lexical meaning, and also defines the structure of a song).

Despite its importance, singing voice has not received much attention from the MIR community. It has only been introduced a few years ago as a standalone topic [12, 17]. One of the most important factors that prevents its development is the absence of large and good quality reference datasets. This problem also exists in other MIR fields, nevertheless several solutions have been proposed [3, 9]. Currently, researchers working in singing voice use small designed dataset following different methodology [10]. Large datasets as the one used in [13] are private and not accessible to the community.

The goal of this paper is to propose such a dataset and to describe the methodology followed to construct it.

1.1 Proposal

We present the DALI dataset: a large Dataset of synchronised Audio, Lyrics and notes that aims to stand as a reference for the singing voice community. It contains 5358 songs (real music) each with – its audio in full-duration, – its time-aligned lyrics and – its time-aligned notes (of the vocal melody). Lyrics are described according to four levels of granularity: notes (and textual information underlying a given note), words, lines and paragraphs. For each song, we also provide additional multimodal information such as genre, language, musician, album covers or links to video clips. The rest of this paper focuses on our methodology for creating DALI. In Figure 1, we illustrate the input and output of our dataset creation system. See Section 4 for more details about the dataset itself.

The DALI dataset has been created automatically. Our approach consists in a constant interaction between dataset creation and learning models where they benefit from each other. We developed a system that acquires lyrics and notes aligned in time and finds the corresponding audio tracks. The time-aligned lyrics and notes come from Karaoke resources (see Section 3.1 for more details). Here, non-expert users manually describe the lyrics of a song as a sequence of annotations: time aligned notes with their associated textual information. While this information is powerful it has two major problems: **1)** there is no information about the exact audio used for the annotation process (only the song title and artist name which may lead to many different audio versions), **2)** even if the right audio is found,



© Gabriel Meseguer-Brocal, Alice Cohen-Hadria, Geoffroy Peeters. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Gabriel Meseguer-Brocal, Alice Cohen-Hadria, Geoffroy Peeters. "DALI: a large Dataset of synchronized Audio, Lyrics and notes, automatically created using teacher-student machine learning paradigm", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

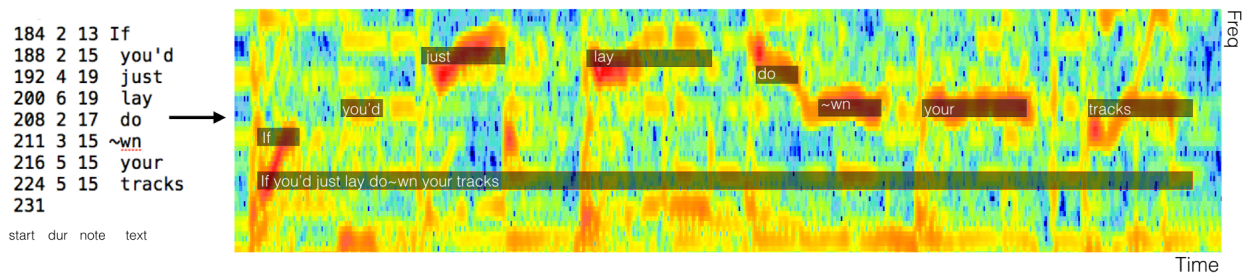


Figure 1: [Left part] The inputs of our dataset creation system are karaoke-user annotations presented as a triple of {time (start + duration), musical-notes, text}. [Right part] Our dataset creation system automatically finds the corresponding full-audio track and aligned the vocal melody and the lyrics to it. In this example, we illustrate the alignment for a small excerpts. We only represent two levels of lyrics granularity: notes and lines.

annotations may need to be adjusted to fit the audio perfectly. In Section 3.2, we define how we retrieve from the Web the possible audio candidates for each song. In Section 3.3, we describe how we select the right audio among all the possible candidates and how we automatically adapt the annotated time-alignment lyrics to this audio. In order to do this, we propose a distance that measures the correspondence between an audio track and a sequence of manual annotations. This distance is also used to perform the necessary adaptations on the annotations to be perfectly aligned with the audio. Our distance requires the audio to be described as a singing voice probability sequence. This is computed using a singing voice detection (SVD) system based on deep convolutional neural network (ConvNet). The performance of our system highly depends on the precision of the SVD. Our first version is trained on few but accurately-labeled ground truths. While this system is sufficient to select the right audio it is not to get the best alignment. To improve the SVD, in Section 3.4 we propose to use a teacher-student paradigm. Thanks to the first SVD system (the teacher) we selected a first set of audio tracks and their corresponding annotations. Using them, we train new SVD systems (the students). We show in Section 3.4.1 that new SVD systems (the students) are better than the initial one (the teacher). With this new version, we increase the quality and size of the DALI dataset. Finally, we discuss our research in Section 5.

2. RELATED WORKS

We review previous works related to our work: singing voice detection methods and the teacher-student paradigm.

Singing Voice detection. Most approaches share a common architecture. Short-time observations are used to train a classifier that discriminates observations (per frame) in vocal or non-vocal classes. The final stream of predictions is then post-processed to reduce artifacts.

Early works explore classification techniques such as Support Vector Machines (SVMs) [16, 20], Gaussian mixture model (GMM) [11] or multi-layer perceptron (MLP) [4]. Other approaches also tried to use specific vocal traits such as vibrato and tremolo [21] or to adapt speech recognition systems for the particularities of singing voice [5].

Over the past few years, most works focus on the use of deep learning techniques. For example, [23, 24] propose the use of ConvNet combined with data augmentation techniques (to increase the size of the training set) or trained on weakly labeled data (the data are only labeled at the file level, not at the segment level). [13] also proposes the use of CNN but with a Constant-Q input and a training on a very large private datasets mined from Spotify resources. Some researchers suggest the use of Recurrent Neural Networks (RNN) [15] or Long Short-Term Memory (LSTM) [14]. One advantage of these models is that they directly model the decisions sequence over time and no post-processing is needed. Other singing voice detection systems are developed to be used as a pre-processing-step: for lyrics transcription [17] or for source separation [25] trained then to obtain ideal binary masks.

Teacher-student paradigm. Teacher-student learning paradigm [2, 28] has appeared as a solution to overcome the problem of insufficient labeled training data in MIR. Since manual labeling is a time-consuming tasks, the teacher-student paradigm explores the use of unlabeled data for supervised problems. The two main agents of this paradigm are: the teacher and the student. The *teacher* is trained with labels of well known ground truths datasets (often manually annotated). It is then used to automatically label unlabeled data on a (usually) larger dataset. These new labels (the one given by the teacher) are the ones used for training the *student(s)*. Student(s) indirectly acquire(s) the desired knowledge by mimicking the “teacher behaviour”. This model has achieved great results for tasks in speech recognition [27] and multilingual models [8]. It has also been proved that student(s) can achieve superior performances than the teacher [8, 28].

3. SINGING VOICE DATASET: CREATION

3.1 Karaoke resources

Outside the MIR community there are rich sources of information that can be explored. One of these sources is Karaoke video games that fit exactly our requirements. In these games, users have to sing along with the music to win points according to their singing accuracy. To measure their accuracy, the user melody is compared with a

Table 1: Terms overview: definition of each term used in this paper.

Term	Definition
Annotation	basic alignment unit as a triple of time (start + duration wrt Fr), musical-notes (with $0 = C3$) and text.
A file with annotations	group of annotations that define the alignment of a particular song.
Offset.time as O	it indicates the start of the annotations, its modifications moves all back to the right or left.
Frame rate as Fr	it controls the annotation grid size stretching or compressing its basic unit.
Annotation voice sequence as $avs(t) \in \{0, 1\}$	singing voice (SV) sequence extracted from karaoke-users annotations.
Predictions as $\hat{p}(t) \in [0, 1]$	SV probability sequence provided by our singing voice detection.
Labels	labels sequence of well known ground truths datasets checked by the MIR community.
Teacher	first SV detection (SVD) system trained on Labels.
Student	new SVD system trained on the $avs(t)$ for the subset of track for which $NCC(\hat{o}, \hat{fr}) \geq T_{corr}$.

reference timing note (that has fine time and frequency). Hence, large datasets of time-aligned note and lyrics exist.

Such datasets can be found as open-source. Nowadays, there are several active and big karaoke open-source communities. In those, non-expert users exchange text files containing lyrics and melody annotations. However there is no further professional revision. Each file contains all the necessary information to describe a song:

- the sequence of triplets {time, musical-notes, text},
- the `offset.time` (start of the sequence) and `frame rate` (annotation time-grid),
- the `song.title` and the `artist.name`.

We refer to Table 1 for the definition of all the terms we use. These annotations can be transformed to get the time and note frequencies as seen Figure 1.

We were able to retrieve 13339 karaoke annotation files. Although this information is outstanding for our community, it presents several problems that have to be solved:

Global. When performing the annotation, users can choose the *audio file* they want. The problem is that only the `song.title` and `artist.name` are provided. This combination might refer to different audio versions (studio, radio edit, live, remix, etc.). Consequently, we do not know which audio version has been used. Annotations made for a version do not work for another. Besides, even if the correct audio is known, annotations may not perfectly fit it. As a result annotations must be adapted. This is done by modifying the provided `offset.time` and `frame rate`. These issues are not problematic for karaoke-users but critical To the automatic creation of a large dataset for MIR research.

Local. It refers to errors due to fact the that users are non-professionals. It covers local alignment problems of particular lyric blocks, text misspellings or note mistakes.

In this paper we only focus on global problems leaving the local ones for future works.

WASABI is a semantic database of song knowledge gathering metadata collected from various music databases on the Web [18]. In order to benefit from the richness of this database, we first linked each annotation file to Wasabi. To that end, we connected a specific `song.title` and `artist.name` with all possible corresponding audio versions (studio, radio, edit, live, remix, etc.). The

WASABI also provides lyrics in a text only annotations (grouped by lines and paragraphs). Using the two lyrics representations (note-based annotations and text only annotations), we created four levels of granularity: notes, words, lines and paragraphs. Finally, WASABI also provides extra multimodal information such as cover images, links to video clips, metadata, biography, expert notes, etc.

3.2 Retrieving audio candidates

Our input is an annotation file connected to the WASABI database. This database provides us with the different existing versions (studio, radio, edit, live, remix, etc.) for a `song.title` and `artist.name` combination. Knowing the possible versions, we then automatically query YouTube¹ to get a set of audio candidates. We now need to select among the set of audio candidates the one corresponding to the annotation file.

3.3 Selecting the right audio from the candidate and adapting annotation to it

Each audio candidate is compared to the reference annotation file. We do this by measuring a distance between both and keeping the one with the largest value.

Audio and annotations live in two different representation spaces that cannot be directly compared. In order to find a proper distance, we need to transform them to a common representation space. Two directions were studied:

Annotations as audio. We have explored lyrics synchronization techniques [10] but their complexity and phonetic model limitations prevent us to use them. As annotations can be transformed into musical notes, score alignment approaches [7, 26] seem a natural choice. However, due to missing information in the corresponding score (we only have the score of the vocal melody) these systems failed. We then tried to reduce the audio to the vocal melody (using Melodia [22]) and then align it to the vocal melody score but this also failed. Consequently, we did not persist in this direction.

Audio as annotations. The idea we develop in the remainder is the following. We convert the audio track to a singing-voice probability $\hat{p}(t)$ over time t . This sequence has value $\hat{p}(t) \rightarrow 1$ when voice is present at time t and $\hat{p}(t) \rightarrow 0$ otherwise. This probability is computed from the audio signal using a **Singing Voice Detection** (SVD)

¹ We use <https://github.com/rg3/youtube-dl>

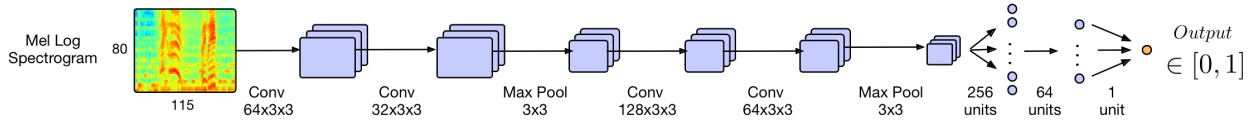


Figure 2: Architecture of our Singing Voice Detection system using a ConvNet.

system described below. We name this function **predictions**. Similarly, the sequence of annotated triplets {time, musical-notes, text} can be mapped to the same space: $avs(t) = 1$ when a vocal note exists at t and $avs(t) = 0$ otherwise. We name this function **annotation voice sequence**.

Singing Voice Detection system. Our system is based on the deep Convolutional Neural Network proposed by [24]. The audio signal is first converted to a sequence of patches of 80 Log-Mel bands over 115 time frames. Figure 2 shows the architecture of the network. The output of the system represents the singing voice probability for the center time-frame of the patch. The network is trained on binary target using cross-entropy loss-function, ADAMAX optimizer, mini-batch of 128, and 10 epochs.

Cross-correlation. To compare audio and annotation, we simply compare the functions $\hat{p}(t)$ and $avs(t)$. As explained before, the annotation files also come with a proposed `offset_time` and `frame_rate`. We denote them by O and Fr in the following. The alignment between $\hat{p}(t)$ and $avs(t)$ depends on the correctness of O and Fr values. We will search around O and Fr to find the best possible alignment. We denote by o the correction to be applied to O and by fr the best Fr . Our goals are to:

1. find the value of o and fr that provides the best alignment between $\hat{p}(t)$ and $avs(t)$,
2. based on this best alignment, deciding if $\hat{p}(t)$ and $avs(t)$ actually match each other and establishing if the match is good enough to be kept.

Since we are interested in a global matching between $\hat{p}(t) \in [0, 1]$ and $avs(t) \in \{0, 1\}$ we use the normalized cross-correlation (NCC) as distance²:

$$NCC(o, fr) = \frac{\sum_t avs_{fr}(t - o) \hat{p}(t)}{\sqrt{\sum_t avs_{fr}(t)^2} \sqrt{\sum_t \hat{p}(t)^2}}$$

The NCC provides us directly with the best \hat{o} value. This value directly provides the necessary correction to be applied to O to best align both sequences.

To find the best value of fr we compress or stretch annotation by changing the grid size. This warp is constant and respect the annotation structure. We denote it as $avs_{fr}(t)$. The optimal fr value is computed using a brute

² Matches between $\hat{p}(t)$ and $avs(t)$ can also be found using Dynamic Time Warping (DTW). However, we found its application not successful for our purpose. Indeed, DTW computes local warps that does not respect the global structure of the user annotations. In addition, its score is not normalized preventing its use for matches selection.

force approach, testing the values of fr around the original Fr in an interval controlled by α (we use $\alpha = Fr * 0.05$):

$$\hat{fr}, \hat{o} = \arg \max_{fr \in [Fr - \alpha, Fr + \alpha], o} NCC(o, fr)$$

Our final score is given by $NCC(\hat{o}, \hat{fr})$.

The audio is considered as good match the annotation if $NCC(\hat{o}, \hat{fr}) \geq T_{corr}$. The value of T_{corr} has been found empirically to be $T_{corr} = 0.8$. For a specific annotation, if several audio candidate tracks have a value $NCC \geq T_{corr}$, we only keep the one with the largest value. $T_{corr} = 0.8$ is quite restrictive but even if we may loose good pairs we ensure that those we keep are well aligned. When an audio match is found, the annotations are adapted to it using \hat{fr} and \hat{o} .

Necessity to improve the Singing Voice Detection system. The score NCC proposed above strongly depends on the quality of $\hat{p}(t)$ (the prediction provided by the **Singing Voice Detection (SVD)** system). Small differences in predictions lead to similar $NCC(\hat{o}, \hat{fr})$ values but very different alignments. While the predictions of the baseline SVD system are good enough to select the correct audio candidates (although there are still quite a few false negatives), it is not good enough to correctly estimate \hat{fr} and \hat{o} . As improving the SVD system the number of false negatives will be reduced and we will also find better alignments. We hence need to improve our SVD system.

The idea we propose below is to re-train the SVD system using the set of candidates audio that match the annotations. This is a much larger training set (around 2000) than the one used to train the baseline system (around 100). We do this using a teacher-student paradigm.

3.4 Teacher-Student

Our goal is to improve our Singing Voice Detection (SVD) system. If it becomes better, it will find better matches and align more precisely audio and annotations. Consequently, we will obtain a better DALI dataset. This larger dataset can then be used to train a new SVD system which again, can be used to find more and better matches improving and increasing the DALI dataset. This can be repeated iteratively. After our first iteration and using our best SVD system, we reach 5358 songs in the DALI dataset.

We formulate this procedure as a Teacher-Student paradigm. The processing steps of the whole Singing Voice Dataset creation is summarized in Figure 3.

Upper left box. We start from Karaoke resources that provide our set of annotation files. Each annotation file defines a sequence of triplets {time, note, next} that we con-

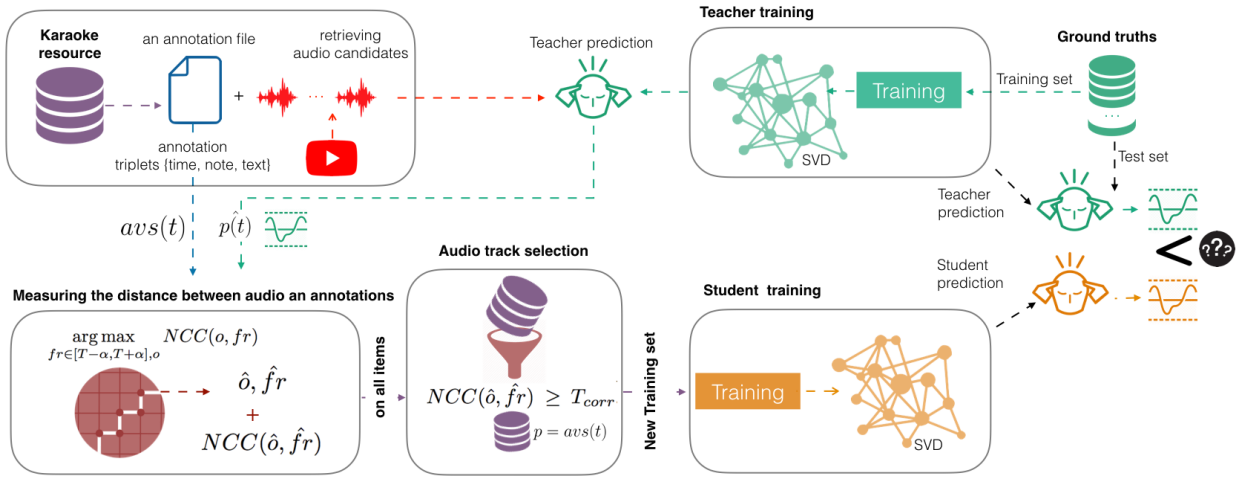


Figure 3: Singing Voice Dataset creation using a teacher-student paradigm.

vert to an annotation voice sequence $avs(t)$. For each annotation file, we retrieve a set of audio candidates.

Upper right box. We independently trained a first version of the SVD system (based on ConvNet) using the training set of a ground truth labeled dataset as provided by the Jamendo [20] or MedleyDB [6] datasets. We call this first version the **teacher**.

Upper middle part. This teacher is then applied on each audio candidate to predict $\hat{p}(t)$.

Lower left box. We measure the distance between $avs(t)$ and $\hat{p}(t)$ using our cross-correlation method. It allows us to find the best audio candidate for an annotation file and the best alignment parameters \hat{f}_r, \hat{o} .

Lower middle box. We select the audio annotation pairs for which $NCC(\hat{o}, \hat{f}_r) \geq T_{corr} = 0.8$. The set of selected audio tracks forms a new training set.

Lower right box. This new set is then used to train a new SVD systems based on the same CNN architecture. This new version is called the **student**. To this end, we need to define the target p to be minimized in the loss $\mathcal{L}(p, \hat{p})$.

There are three choices:

- we use as target p the predicted value \hat{p} given right by the **teacher** (usual teacher-student paradigm).
- we use as target p the value avs corresponding to the annotations after aligning them using \hat{f}_r and \hat{o} .
- a combination of both, keeping only these frames for which $\hat{p}(t) = avs(t)$.

Up to now and since the avs have been found more precise than the \hat{p} we only investigated option **b**).

We compare in the following part the results obtained using different teachers and students.

3.4.1 Validating the teacher-student paradigm

In this part, we demonstrate that the students trained on the new training-set actually perform better than the teacher trained on the ground-truth label dataset.

Ground-truth datasets: We use two ground-truth labels datasets: *Jamendo* [20] and *MedleyDB* [6]. We created a third dataset by merging *Jamendo* and *MedleyDB* named as *J+M*. Each dataset is split into a train and a test part using an artist filter (the same artist cannot appear in both).

Teachers: With each ground-truth datasets we trained a teacher using only the training part. Once trained, each teacher is used to select the audio matches as described in Section 3.3. As a result, we produce three new training sets. They contains 2440, 2673 and 1596 items for the teacher *J+M*, *Jamendo* and *MedleyDB* respectively. The intersection of the three sets (not presented here) indicates that 89.8 % of the tracks selected using the *MedleyDB* teacher are also present within the tracks selected using the *J+M* teacher or the *Jamendo* teacher. Also, 91.4 % of the tracks selected using the *Jamendo* teacher are within the tracks selected using the *J+M* teacher. It means that the three teachers agree most of the time on selecting the audio candidates.

Students: We train three students using the audio and the avs value of the new training sets. Even if there is a large audio files overlap within the training sets, their alignment (and therefore the avs value) is different. The reason to this is that each teacher gets a different \hat{p} which results in different \hat{f}_r, \hat{o} values.

3.4.2 Results

We evaluate the performances of the various teachers and students SVD systems using the test parts of *Jamendo* (J.test) and *MedleyDB* (M.test). We measure the quality of each SVD system using the frame accuracy i.e. average value over all the tracks of the test set.

Results are indicated in Table 2. In this table, e.g. “Student (Teacher_J.train) (2673)” refers to the student trained

on the 2673 audio candidates and the *avs* values computed with the Teacher trained on *Jamendo* train set.

Table 2: Performances of the teachers and students using the various datasets. Number of tracks in brackets.

SVD system \ Test_set	J_test (16)	M_test (36)
Teacher_J_train (61)	87%	82%
Student (Teacher_J_train) (2673)	82%	82%
Teacher_M_train (98)	76%	85%
Student (Teacher_M_train) (1596)	80%	84%
Teacher_J+M_train (159)	82%	82%
Student (teacher_J+M_train) (2440)	86%	87%

Performance of the teachers. We first test the teachers. *Teacher_J_train* obtains the best results on J_test (87%). *Teacher_M_train* obtains the best results on M_test (85%). In both cases, since training and testing are performed on two parts of the same dataset, they share similar audio characteristics. These results are artificially high. To best demonstrate the generalization of the trained SVD systems, we need to test them in a cross-dataset scenario, namely train and test in different datasets.

Indeed, in this scenario the results are quite different. Applying *Teacher_J_train* on M_test the results decreases down to 82% (a 5% drop). Similarly when applying *Teacher_M_train* on J_test the results decreases down to 76% (a 9% drop). Consequently, we can say that the teachers do not generalize very well.

Lastly, the *Teacher_J+M_train* trained on J+M_train actually performs worse on both J_test (82%) and M_test (82%) than their non-joined teacher (87% and 85%). These results are surprising and remain unexplained.

Performance of the students. We now test the students. It is important to note that students are always evaluated in a cross-dataset scenario since the DALI dataset (on which they have been trained) does not contain any track from *Jamendo* or *MedleyDB*. Hence, there is no possible overfitting for those. Our hypothesis is that students achieve better the results than the teachers because they have seen more data. Especially, we assume that their generalization to unseen data will be better.

This is true for the performances obtained with the student based on *Teacher_M_train*. When applied to J_test, it reaches 80% which is higher than the performances of the *Teacher_M_train* directly (76%).

This is also true for the performances computed with the student based on *Teacher_J+M_train*. When applied either to J_test or M_test, it reaches 86.5% (86% on *Jamendo* and 86% on *MedleyDB*) which is above the *Teacher_J+M_train* (82%). Also, 86.5% is similar or above the results obtained with *Teacher_J_train* on J_test (87%) and *Teacher_M_train* on M_test (85%). This is a very interesting result that demonstrates the generalization of the student system whichever data-set it is applied to. The **student** based on *Teacher_J+M_train* is the one used for defining the final 5358 songs of the DALI dataset.

However, the performances obtained with the student

based on *Teacher_J_train* applied to M_test (82%) do not improve over the direct use of the *Teacher_J_train* (82%).

On alignment. Not explained in this paper is the fact that the $\hat{f}r$ and \hat{o} values computed with the students are much better (almost perfect) than the ones obtained with the teacher. However, we cannot measure it precisely since DALI dataset does not have ground-truth label annotations to that end. Indeed, the goal of this paper is exactly to obtain such annotations automatically.

4. SINGING VOICE DATASET: ACCESS

The DALI dataset can be downloaded at <https://github.com/gabolsgabs/DALI>. There, we provide the detailed description of the dataset as well as all the necessary information for using it. DALI is presented under the recommendation made by [19] for the description of MIR corpora. The current version of DALI is 1.0. Future updates will be detailed in the website.

5. CONCLUSION AND FUTURE WORKS

In this paper we introduced DALI, a large and rich multimodal dataset containing 5358 audio tracks with their time-aligned vocal melody notes and lyrics at four levels of granularity.

We explained our methodology where dataset creation and learning models interact using a teacher-student paradigm benefiting one-another. From manual karaoke user annotations of time-aligned lyrics and notes, we found a set of matching audio candidates from the Web. To select and align the best candidate, we compare the annotated vocal sequence (corresponding to the lyrics) to the singing voice probability (obtained with a ConvNet). To improve the latter (and therefore obtain a better selection and alignment) we applied a teacher-student paradigm.

Through an experiment, we proved that the students outperform the teachers notably in a cross-dataset scenario, when train-set and test-set are from different datasets.

It is important to note that the results of the students are higher than the teacher ones, even if they have been training on imperfect data. In our case, we showed that, in the context of deep learning, it is better to have imperfect but large dataset rather than small and perfect ones. However, other works went in the opposite direction [1].

Future work. We have only performed the teacher-student iteration once. In next works will use the results of the first student generations to train a second student generations. This will define a new DALI dataset. We plan to quantitative measure the quality of \hat{o} , $\hat{f}r$ and to continue exploring the alignments between note annotations and the audio. Currently, we trained our student using as target $p = avs$, which do not transfer directly the knowledge of the teacher. We will explore other possibilities of knowledge transfer using other targets (points a) and c) in Section 3.4) as well as the local problems describe at Section 3.1.

Acknowledgement. This research has received funding from the French National Research Agency under the contract ANR-16-CE23-0017-01 (WASABI project).

6. REFERENCES

- [1] Xavier Amatriain. "in machine learning, is more data always better than better algorithms?". <https://bit.ly/2seQzj9>.
- [2] A. Ashok, N. Rhinehart, F. Beainy, and K. M. Kitani. N2N learning: Network to network compression via policy gradient reinforcement learning. *CoRR*, 2017.
- [3] K. Benzi, M. Defferrard, P. Vanderghyest, and X. Bresson. FMA: A dataset for music analysis. *CoRR*, abs/1612.01840, 2016.
- [4] A. Berenzweig, D. P. W. Ellis, and S. Lawrence. Using voice segments to improve artist classification of music. In *AES 22*, 2002.
- [5] A. L. Berenzweig and D. P. W. Ellis. Locating singing voice segments within music signals. In *WASPAA*, pages 119–122, 2001.
- [6] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Canam, and J. Bello. Medleydb: A multitrack dataset for annotation-intensive mir research. In *ISMIR*, 2014.
- [7] A. Cont, D. Schwarz, N. Schnell, and C. Raphael. Evaluation of Real-Time Audio-to-Score Alignment. In *ISMIR*, Vienna, Austria, 2007.
- [8] J. Cui, B. Kingsbury, B. Ramabhadran, G. Saon, T. Sercu, K. Audhkhasi, A. Sethy, M. Nussbaum-Thom, and A. Rosenberg. Knowledge distillation across ensembles of multilingual models for low-resource languages. In *ICASSP*, 2017.
- [9] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra. Freesound datasets: A platform for the creation of open audio datasets. In *ISMIR*, Suzhou, China, 2017.
- [10] H. Fujihara and M. Goto. Lyrics-to-Audio Alignment and its Application. In *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*, pages 23–36. Dagstuhl, Germany, 2012.
- [11] H. Fujihara, M. Goto, J. Ogata, and H. G. Okuno. Lyricsynchronizer: Automatic synchronization system between musical audio signals and lyrics. 5(6):1252–1261, 2011.
- [12] M. Goto. Singing information processing. In *ICSP*, pages 2431–2438, 2014.
- [13] E. J. Humphrey, N. Montecchio, R. Bittner, A. Jansson, and T. Jehan. Mining labeled data from web-scale collections for vocal activity detection in music. In *ISMIR*, 2017.
- [14] S. Leglaive, R. Hennequin, and R. Badeau. Singing voice detection with deep recurrent neural networks. In *IEEE, editor, ICASSP*, pages 121–125, Brisbane, Australia, 2015.
- [15] B. Lehner, G. Widmer, and S. Bock. A low-latency, real-time-capable singing voice detection method with lstm recurrent neural networks. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, 2015.
- [16] M. Mauch, H. Fujihara, K. Yoshii, and M. Goto. Timbre and melody features for the recognition of vocal activity and instrumental solos in polyphonic music. In *ISMIR 2011*, pages 233–238, 2011.
- [17] A. Mesaros. Singing voice identification and lyrics transcription for music information retrieval invited paper. In *7th Conference on Speech Technology and Human - Computer Dialogue (SpeD)*, pages 1–10, 2013.
- [18] G. Meseguer-Brocal, G. Peeters, G. Pellerin, M. Buffa, E. Cabrio, C. Faron Zucker, A. Giboin, I. Mirbel, R. Hennequin, M. Moussallam, F. Piccoli, and T. Fillion. WASABI: a Two Million Song Database Project with Audio and Cultural Metadata plus WebAudio enhanced Client Applications. In *Web Audio Conf.*, London, U.K., 2017. Queen Mary University of London.
- [19] G. Peeters and K. Fort. Towards a (better) Definition of Annotated MIR Corpora. In *ISMIR*, Porto, Portugal, 2012.
- [20] M. Ramona, G. Richard, and B. David. Vocal detection in music with support vector machines. In *Proc. ICASSP '08*, 2008.
- [21] L. Regnier and G. Peeters. Singing Voice Detection in Music Tracks using Direct Voice Vibrato Detection. In *ICASSP*, page 1, taipei, Taiwan, 2009.
- [22] J. Salamon and E. Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech and Language Processing*, 20:1759–1770, 2012.
- [23] J. Schlüter. Learning to pinpoint singing voice from weakly labeled examples. In *ISMIR*, New York City, USA, 2016. ISMIR.
- [24] J. Schlüter and T. Grill. Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks. In *ISMIR 2015*, Malaga, Spain, 2015.
- [25] A. J. R. Simpson, G. Roma, and M. D. Plumbley. Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network. abs/1504.04658, 2015.
- [26] F. Soulez, X. Rodet, and D. Schwarz. Improving polyphonic and poly-instrumental music to score alignment. In *ISMIR*, page 6, Baltimore, United States, 2003.
- [27] S. Watanabe, T. Hori, J. Le Roux, and J. Hershey. Student-teacher network learning with enhanced features. In *ICASSP*, pages 5275–5279, 2017.
- [28] C. Wu and A. Lerch. Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data. In *ISMIR*, 2017.

OpenMIC-2018: AN OPEN DATASET FOR MULTIPLE INSTRUMENT RECOGNITION

Eric J. Humphrey
Spotify
ejhumfrey@spotify.com

Simon Durand
Spotify
durand@spotify.com

Brian McFee
New York University
brian.mcfee@nyu.edu

ABSTRACT

Identification of instruments in polyphonic recordings is a challenging, but fundamental problem in music information retrieval. While there has been significant progress in developing predictive models for this and related classification tasks, we as a community lack a common data-set which is large, freely available, diverse, and representative of naturally occurring recordings. This limits our ability to measure the efficacy of computational models.

This article describes the construction of a new, open data-set for multi-instrument recognition. The dataset contains 20,000 examples of Creative Commons-licensed music available on the Free Music Archive. Each example is a 10-second excerpt which has been partially labeled for the presence or absence of 20 instrument classes by annotators on a crowd-sourcing platform. We describe in detail how the instrument taxonomy was constructed, how the data-set was sampled and annotated, and compare its characteristics to similar, previous data-sets. Finally, we present experimental results and baseline model performance to motivate future work.

1. INTRODUCTION

Music information retrieval (MIR) applications often depend on statistical models and machine learning algorithms to relate audio content to semantically meaningful representations. The development and evaluation of these methods, in turn, depends on access to data, typically audio recordings which have been annotated for a particular task such as chord recognition or tag prediction. Ideally, the data we use to develop and evaluate models should be large, diverse, and open access, so that we as researchers and engineers can diagnose failure modes and propose improvements. However, because the vast majority of music is subject to copyright, this has historically been difficult to achieve. This has resulted in a proliferation of *de facto* standard data-sets which are small, biased, and not freely available, which ultimately impedes scientific progress.

To address this problem, McFee et al. [15] proposed an iterative evaluation framework for developing open access data-sets for MIR, with a specific focus on instrument recognition. While this proposal was apparently met with enthusiasm from the community, little progress has been made in the intervening time toward enacting the proposal. We hypothesize that this was primarily due to two factors: a lack of a conveniently accessible audio data, and the expense of creating the initial development set. Recently, two complementary data-sets have been published, which we combine here to resolve both of these issues: the Free Music Archive data-set [8], and AudioSet [11]. The result is a diverse, open access collection of 20,000 audio clips annotated for the presence of 20 distinct instrument categories, which we denote as *OpenMIC-2018*.

1.1 Our contributions

Our primary technical contribution is a new, open dataset for training and evaluating instrument recognition algorithms. This article describes in detail how the dataset was constructed by using a combination of model transfer from previous datasets and crowd-sourced annotation. Our goals in documenting the data construction process are two-fold. First, it provides transparency around the various decisions and compromises made in this specific dataset. Second, we describe technical issues and general solutions which may be of interest to future developers of music datasets.

1.2 Related work

Instrument recognition, either monophonic or polyphonic, is a long-standing problem in MIR, and many datasets for evaluating methods have been developed over the years. Table 1 lists some of the commonly used datasets, along with various descriptive attributes. Of specific interest are the size of the collections, the number of instrument classes, the duration of each example, the diversity of the collection (*e.g.*, genre or style), whether the examples are polyphonic, the number of instrument labels per example, and whether the data is open access.

Broadly speaking, existing datasets can be broken into two categories, according to whether samples contain notes played by isolated instruments (RWC [12], Good-sounds [1], or NSynth [10]), or recordings of instrument ensembles. Datasets of isolated instrument recordings are often easier to produce and annotate at large scale because long recordings spanning multiple notes can be segmented



© Eric J. Humphrey, Simon Durand, Brian McFee. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Eric J. Humphrey, Simon Durand, Brian McFee. "OpenMIC-2018: An open dataset for multiple instrument recognition", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

Table 1. A qualitative comparison of different existing datasets for instrument identification.

Collection	# Examples	# Instruments	Duration	Diverse	Polyphonic	Multi-label	Open
RWC [12]	3,544	50	scale				
Good-sounds [1]	6,548	12	note				✓
NSynth [10]	305,979	1,006	note				✓
MedleyDB [3]	122	80	song	✓	✓	✓	
MusicNet [19]	330	11	song		✓	✓	✓
IRMAS [4]	6,705	11	3s	✓	✓		
OpenMIC-2018	20,000	20	10s	✓	✓	✓	✓

to generate examples with a shared label. However, the acoustic properties of ensemble recordings differ significantly from those of isolated recordings, so models developed on single-instrument data often do not generalize to the polyphonic case. Conversely, ensemble recordings are typically difficult to precisely annotate, which results in either high-quality collections with a small number of distinct tracks (MedleyDB [3] or MusicNet [19]), or in collections with more tracks but with only partial annotations (such as IRMAS [4] with predominant instrument tags for short excerpts). An ideal dataset would be large, diverse, strongly annotated (including both positive and negative examples), and freely available, so that at each instant in any recording, full information about all active instruments is available. While existing datasets succeed on some of these criteria, none achieves all simultaneously.

1.3 The Free Music Archive

The Free Music Archive¹ (FMA) is a web-based repository of freely available music recordings. Recently, a snapshot of FMA has been released to the research community to facilitate content-based music analysis evaluation [8]. The FMA snapshot includes 106,574 tracks by some 16,341 artists, along with pre-computed features. Each track is annotated with both coarse (16 categories) and fine (161 categories) genre tags. Tracks are provided under a small variety of licenses, with the vast majority being Creative Commons [7]. This allows practitioners to archive and redistribute data (with some minor restrictions), which is fundamental to the practice of open and reproducible scientific research.

While previous authors have noted the particular genre biases present on FMA [8], it nonetheless provides a large pool of realistic musical content which could be used in research applications. Despite the specific quirks of the FMA collection, using it as a basis for large-scale MIR evaluation has several benefits. In addition to the obvious benefits of being open access, it also facilitates data revision and inclusion of new contributions from the community at large. This in turn makes it easier for corrections to be integrated, and the collection to grow over time and not become stale.

2. CONSTRUCTING OpenMIC-2018

In developing OpenMIC-2018, we took inspiration from ImageNet [9]. ImageNet was constructed by selecting

¹ <http://freemusicarchive.org/>

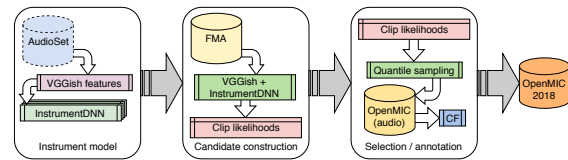


Figure 1. A multi-label instrument detector (*InstrumentDNN*, section 2.2) is trained on AudioSet data. The model is used to score each 10s clip in FMA by likelihood of each instrument (section 2.3). Clips are sorted into quantiles for each instrument, then sub-sampled and annotated by CrowdFlower workers (section 3).

and annotating natural images to represent categories (synonym sets, or *synsets*) drawn from the WordNet ontology [16], with a goal of having at least 500 positive examples for each category. Candidate images were selected by querying image search engines for each category term, and then labels were verified by crowd-sourced annotation. The label correction and verification step was critical at the time, due to the poor accuracy of image search engines when the dataset was constructed in 2009.

We follow a similar strategy here, with a few notable modifications. Rather than querying the Internet for candidate samples, we restrict attention to freely available content hosted on the Free Music Archive, and specifically those with explicit Creative Commons licensing. Additionally, instead of the WordNet ontology, we use the recently published AudioSet concept ontology [11], which itself derives from WordNet, but is adapted to acoustically meaningful concepts. Using existing AudioSet data, we construct a multi-instrument estimator and use this model to rank the unlabeled FMA data and provide candidates for annotation. The remainder of this section describes the entire process in detail, which is visualized in Figure 1.

2.1 AudioSet

AudioSet is a recently released concept ontology and human-annotated dataset derived from YouTube videos, with the goal of providing a testbed for identifying acoustic events [11]. The ontology consists of 632 classes, represented as a lattice-like graph, rather than hierarchical tree structure, *i.e.* one low-level class may have two distinct parents. The annotated dataset consists of at least 100 positive examples of 485 classes, distributed (non-uniformly) across nearly 1.8M video clips of 10 seconds (or less)

drawn from YouTube. Similar in spirit to the work presented here, AudioSet is motivated by a lack of large-scale annotated audio data for scientific research purposes.

While the AudioSet ontology includes musical instruments, the audio data does not match our requirements for an open music instrument sample. The collection is derived from YouTube videos, for which there are no guarantees on the legality of licensing, sharing, and archiving the content. Though abstract features are made available via a publicly available acoustic model, an inability to make the source content directly accessible has limited the value of other large collections, such as the Million Song Dataset [2]. Furthermore, the content is often quite different from musical performances, an important characteristic at the root of what makes this task both challenging and interesting: many of the positively labeled examples are solo performances, which makes it difficult to model and evaluate on realistic, highly correlated ensemble performances.

That said, AudioSet serves two important functions in this project. It is impractical to annotate the entire FMA collection of more than 100K recordings outright; however, it is *also* extremely unlikely that one could draw a random subsample with sufficient representation across a number of instruments. The occurrence of musical instruments is heavily biased by popularity, such as voice, guitar, or piano, and this is especially true in the Free Music Archive. Here, we leverage AudioSet to build a multi-instrument estimator that allows us to sub-sample and more efficiently use annotation resources. For better or worse, we also leverage the previous work in ontology construction, while circumventing the important, but difficult, challenge of selecting which instruments to consider: here, we are limited to only those with enough signal in AudioSet on which to build a baseline model.

We manually identify the classes that correspond to musical instruments, resulting in a set of more than 70 relevant classes. For the sake of coverage, they are merged into “instruments”, *e.g.* “Acoustic Guitar”, “Electric Guitar”, and “Tapping (guitar technique)” become *guitar*, while “Cello” and “Violin” remain distinct. Note that this class resolution is intentionally approximate, as the long-term goals of this project include iteratively refining these concepts as acoustic models improve. We then filter the 1.8M clips in AudioSet to those containing these classes. Unsurprisingly, the distribution skews toward instruments common in Western popular music, such as guitar, violin, or drums, and we cut this list at 1500 examples. Additionally, we randomly draw 8000 non-musical examples as negative instances for building the instrument model described in section 2.2. In summary, the resulting instrument subset consists of 206K clips, totalling roughly 2M seconds (570 hours) of annotated content for 23 instruments.²

2.2 Multi-instrument modeling

AudioSet offers no licensing guarantees on the source content, and there is no approved mechanism for directly accessing the audio data. To make the dataset more gener-

ally useful, the developers of AudioSet have released both a pre-trained feature embedding model [13] based on the VGG architecture for object detection in images [18],³ and its outputs over the original AudioSet audio signals.⁴ This model, referred to as “VGGish”, produces a 128-dimensional feature vector every 0.96 seconds with an equal window size, such that adjacent features capture non-overlapping context. VGGish features are ZCA-whitened and each coefficient is quantized to 8-bits to reduce the footprint of the dataset.

Using the sub-sampling process described above, we filtered the AudioSet features down to those clips relevant for the instrument ontology considered here. The data are conditionally partitioned by YouTube ID into training, validation, and test splits with a 3 : 1 : 1 ratio. We randomly generate over 200 unique, fully connected deep network architectures and hyper-parameter configurations, spanning depth (1–8 layers), width (128 to 2048 units, by powers of 2), the application of dropout and batch normalization, different optimization algorithms (stochastic gradient descent, RMSProp, and Adam [14]), as well as various parameters for each operation. All models are trained for 50 epochs of the training data, and the parameter checkpoint with the highest macro- F_1 (class-averaged) score over the validation data is taken as the best model.

Overall, we find that roughly 15% of the models behave with statistical equivalence, achieving a mean macro- F_1 score of 0.514 ($\sigma = 0.0095$) and a micro- F_1 (item-averaged) score of 0.656 ($\sigma = 0.0056$) on the test partition. The best configuration is determined to be a 7-layer network, with widths of [1024, 512, 256, 1024, 256, 1024, 23], batch-normalization on the first four layers, and pointwise dropout applied to the inputs of the last five [0.0, 0.0, 0.25, 0.125, 0.25, 0.25, 0.5]. The winning model, which we refer to as *InstrumentDNN*, is trained with the Adam optimizer in Keras for 8 epochs, with a learning rate of 0.0001 and a β_1 of 0.99. For reproducibility, the training data and trained model are made publicly available in the source repository.

2.3 FMA clip sampling

The VGGish model is applied to each track in FMA, and the resulting ZCA features are processed by *InstrumentDNN* to produce time-varying instrument likelihoods. Full tracks are then divided into candidate clips by performing maximum-likelihood aggregation over 10 second windows with a 4 second hop size. To account for framing effects, the maximum likelihood of each instrument is taken over the middle 8 seconds, centered on the frame. This produces over 7M clip candidates.

We ultimately want an approximately balanced sample that has good positive representation of each instrument class. Therefore the candidate set is sub-sampled by the following process. First, we consider the median like-

²<https://github.com/cosmir/open-mic-data>

³<https://github.com/tensorflow/models/tree/master/research/audioset>

⁴<https://research.google.com/audioset/download.html>

likelihood of each class over all candidates, and sort instruments in ascending order, as a proxy for class occurrence in the FMA. Then, proceeding from least to most likely instrument class, the clip candidate set is reordered by descending conditional class likelihood. We randomly selected N instances from the 99th percentile rank of that class, such that no two clips share a source track, *i.e.* sampled clips are recording-independent. All remaining clip candidates that also share a common recording with any sampled are discarded, and the process is repeated for the next instrument. For K instruments, the sampling process yields $N \times K$ clips from distinct tracks. Initially, we set $N = 1000$, $K = 23$, but manual inspection of the results revealed three classes that either InstrumentDNN cannot reliably detect, are poorly represented in FMA, or both: *harp*, *bagpipes*, and *harmonica*. We removed these classes, leaving $K = 20$ instruments and 20K clips.

3. CROWD-SOURCED ANNOTATION

At this stage, roughly 25M seconds of audio have been sub-sampled to 200K, a 10^5 reduction, while rebalancing for instrument occurrence. Strongly labeling a collection of this size is still cost-prohibitive, and we must be pragmatic with our annotation efforts. In tackling this challenge, one can think of annotation as a sparse, binary matrix completion problem where most of the values in the instrument occurrence matrix will be zero. Therefore annotation effort is best allocated by flattening this matrix into clip-instrument pairs, and prioritizing likely positives.

Framed this way, our most likely positives are identified by the clip selection process: each instrument has 1K potential positive examples that must be validated by human annotators. We would also like to obtain a number of strong negatives as well, and draw 500 instances per class that fall in the bottom 10^{th} likelihood percentile from the space of examples contained in OpenMIC-2018. Instrument-wise percentile thresholds are computed over the full space of clip candidates. In contrast to positive sampling, negative samples are drawn working from most to least likely instruments. This is because the most likely instrument categories will have the fewest potential strong negatives. Additionally, random sampling is constrained to draw no more than three strong negatives per clip, so as to distribute this information across the collection. Finally, to capture potential correlations and confusions, all additional likelihoods in the 99th percentile rank of their respective instrument classes are added to the pool of binary questions for human annotators. This results in 33,250 potential positive and 10,000 potential negative binary estimates for human validation, which makes up roughly 10% of all possible clip-instrument judgements.

Having identified the questions worth asking, audio annotation presents unique design challenges around *how* to best ask these questions of humans. Unlike images, audio clips cannot be scanned in parallel by humans, and must be auditioned sequentially. This encourages annotation designs that ask several binary questions about the same example. Our first attempt to annotate OpenMIC-2018 took

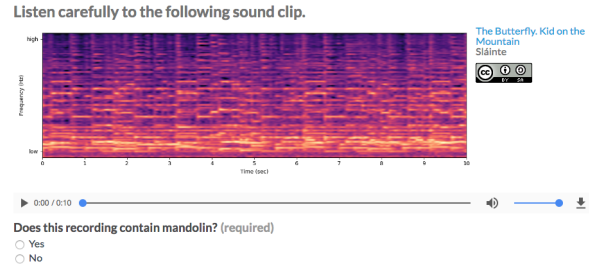


Figure 2. An example annotation task, showing the Mel-spectrogram visualization, playback, response field, and licensing meta-data.

this approach, but we found that annotators struggled with the increased burden of simultaneously judging multiple instrument tags. This resulted in poor agreement, unhappy annotators, and an increased level of effort and skill to complete. Our second attempt used 20 separate annotation tasks, one per instrument, and annotators were asked to determine the presence or absence of a specific instrument across multiple recordings.

Annotation was performed on the CrowdFlower⁵ platform (CF). In contrast to Amazon Mechanical Turk, CF provides quality controls on sets of questions, collectively called a “job”. A single contributor can provide at most 50 responses (or 10% of the job, whichever is larger), and a question is finalized when annotators reach a set agreement level and number of responses.

Additionally, CF makes it easy to include control questions for which an answer is already known. These are used to “quiz” contributors before they can perform any (paid) work on a job, and remove contributors whose accuracy drops below a threshold, *e.g.* 70%. It is important that control questions use clear, unambiguous examples. While these can be easily identified for popular classes, it is difficult in the rare classes, notably *mandolin* and *clarinet*. For these classes, control questions were generated by ranking clips according to the margin between the target instrument’s likelihood and the maximum over other instruments for that clip, which gives preference toward clips where the target instrument was both present and prominent.

Each question is a single judgement of an instrument’s presence or absence for a given audio clip. As shown in Figure 2, we use a radio button interface for the judgement, provide audio playback in the browser, and additionally display an approximately aligned Mel-spectrogram to facilitate the task, inspired by previous audio annotation research [5]. Finally, we are legally obligated to display track title, artist, and license information, which may provide coincidental information about a given track.

4. OpenMIC-2018 ANALYSIS

We collected over 230K judgements from more than 2,500 unique contributors across the 20 instrument classes. Figure 3 summarizes the resulting annotation distributions for

⁵<http://crowdfunder.com>

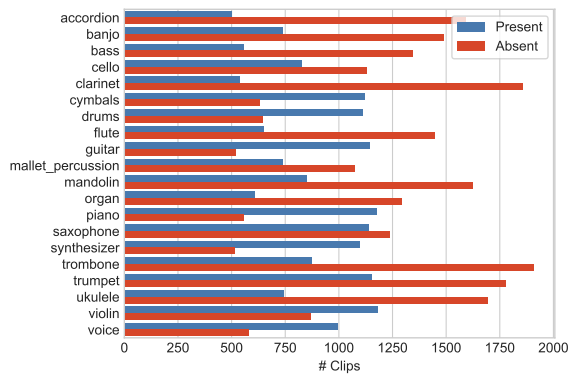


Figure 3. Statistics of crowd-sourced annotation for each instrument in OpenMIC-2018.

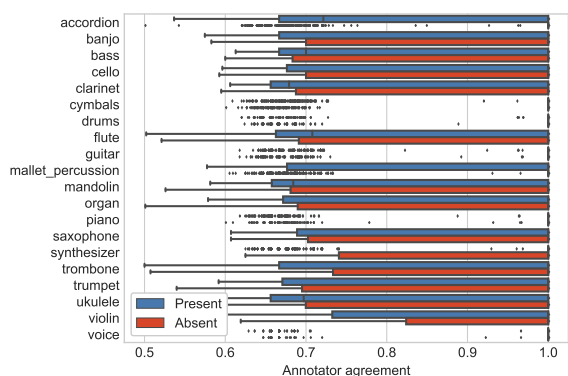


Figure 4. Annotator agreement for each instrument.

each instrument. For each instrument class, the number of confirmed positive and negative clips are plotted separately. Each class has at least 500 confirmed positives, and at least 1500 confirmed positive or negative. Although not every clip is tagged for every instrument, the abundance of strong negative labels facilitates supervised learning and strong evaluation. Figure 4 summarizes the inter-annotator agreements for each instrument’s presence or absence. Some instruments produce more agreement for absence than presence (*accordion*, *violin*), while the reverse is true for others (*synthesizer*). Overall, we observed a high amount of agreement across all instruments.

Figure 5 compares InstrumentDNN’s predicted likelihoods to the annotations for three instrument classes. InstrumentDNN produces a wide range of likelihood values on *mandolin* (fig. 5, left), indicating that the 99th percentile likelihood is well below the threshold for positive detection. This is likely due to a combination of model calibration errors and poor representation in AudioSet. However, the sampling strategy still produced a large number of validated positive examples. For more common classes, such as *cymbals* (fig. 5, center), there is a clearer distinction between the positive and negative selections. For the most common classes, such as *voice* (fig. 5, right), the vast majority of positive selections are validated by the annotators

as positive, and conversely for the negative selections.

To measure the diversity of the annotated subset, fig. 6 compares the distribution of genres over both the sample and the background population of FMA. While both distributions exhibit non-uniform genre distributions, the sample is fairly representative of FMA. The instrument-based sampling does introduce some systematic bias, increasing representation of styles with distinctive instrumentation, such as *classical* or *jazz*. This effect can be observed directly in fig. 7, which shows the number of clips in each genre that are positively labeled for each instrument. For example, the majority of *organ* and *piano* examples are tagged as *classical*, while *synthesizer* is drawn primarily from *electronic* and *experimental*.

4.1 Experiment: baseline modeling

To estimate the expected performance of standard methods on OpenMIC-2018, we conducted a set of baseline experiments. We trained independent binary classifiers for each instrument. We report the accuracy of each of those models on 100 splits of the data, randomly selecting 500 test instances, and splitting the resulting training set in 3 folds for hyper-parameter selection. As input representation, we use the mean and standard deviation of VGGish features over the clip’s duration.

We tested several baseline models, and for simplicity report only the best performing one: a random forest (RF) classifier. The hyper-parameter search is done on the number of trees ($\{10, 100, 1000\}$) and on the maximum depth of the tree ($\{2, 4, 8\}$). We also report the bias point of each instrument category, and the performance of InstrumentDNN. This last comparison point gives us a measure of how much information is gained by the crowd-sourced labels. This experiment is done with the scikit-learn [17], and the code to reproduce will be made available.

The results are shown in fig. 8. We see an overall gain in accuracy of more than 10 percent point (pp) compared to both the bias points and InstrumentDNN. The performance difference can partly be explained by the difference in training distributions between RF and InstrumentDNN, and because a strong signal can be learned from the dataset. The RF model performance is also more consistent across instruments with only a 20 pp difference between the worst and best instrument accuracy, compared to a 34 pp difference for InstrumentDNN. The gain compared to InstrumentDNN is therefore larger on the more difficult instruments, such as saxophone, mandolin and ukulele. In that case the crowd-sourced judgments might provide more value and help build a robust system.

5. CONCLUSION

OpenMIC-2018 should prove to be useful for developing and evaluating instrument detection models. We note that the dataset is not “complete” in that not every clip has been annotated for the presence or absence of every instrument. While this is true for every instrument dataset—if one considers instruments outside its vocabulary—it is usually not

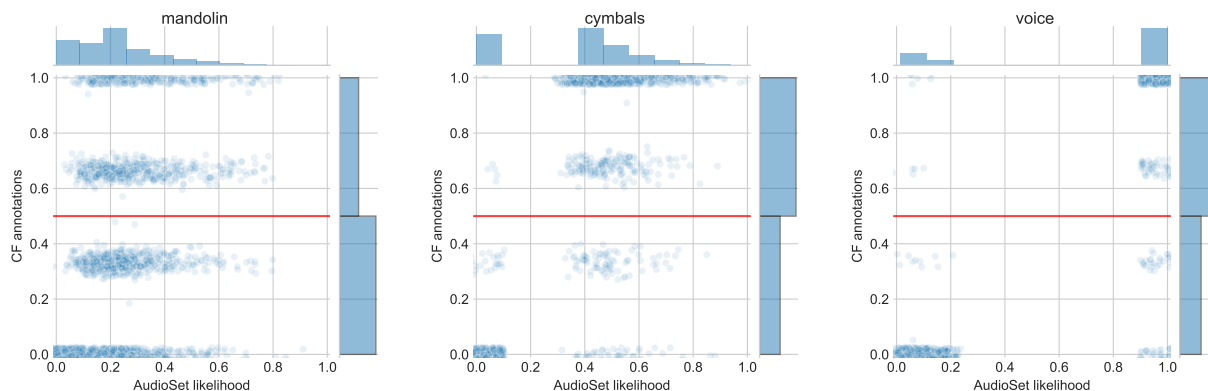


Figure 5. The distribution of the initial model likelihood compared to the crowd-sourced annotations for three instruments. Each dot represents a clip, the horizontal line indicates the majority vote threshold, and the marginal distributions of model likelihood and crowd agreement are shown as bar plots. Data have been randomly perturbed for clarity of visualization.

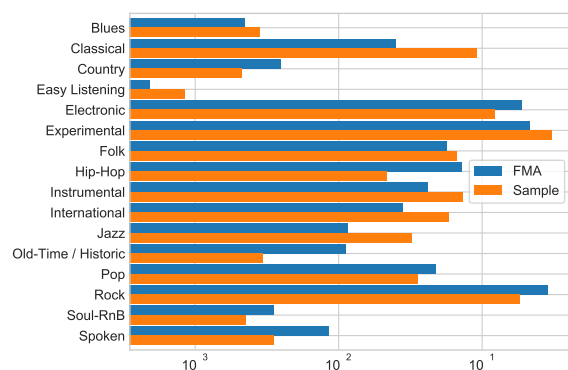


Figure 6. The distribution of (top) genres over the selected sample clips, and the background population in FMA.

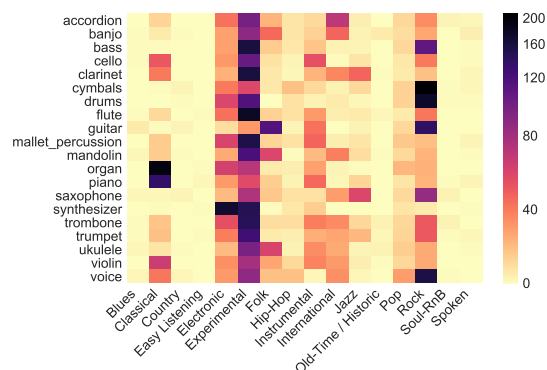


Figure 7. The distribution of (top) genres over the positive candidate sets for each instrument.

taken into consideration as part of the dataset design.

More generally, previous datasets have not typically been designed with a plan for future correction, revision, and expansion. We are explicitly planning to expand and revise the dataset over time, either by additional crowd-sourcing, semi-supervised learning [6], or incremental evaluation [15]. OpenMIC-2018 will be placed under version control, archived, and each revision will receive a unique document object identifier (DOI) via Zenodo.⁶

In addition to supporting corrections and expanded coverage, we anticipate expanding the vocabulary beyond the initial 20 classes, both in breadth of instrument classes, and in depth to provide refinements of classes, such as *alto saxophone* and *tenor saxophone* rather than *saxophone*. Similarly, future work could re-use much of the framework developed here to annotate the same collection for a variety of qualities beyond instrumentation, and facilitate the development of integrated multi-task models.

Acknowledgments. B.M. is supported by the Moore-Sloan Data Science Environment at NYU.

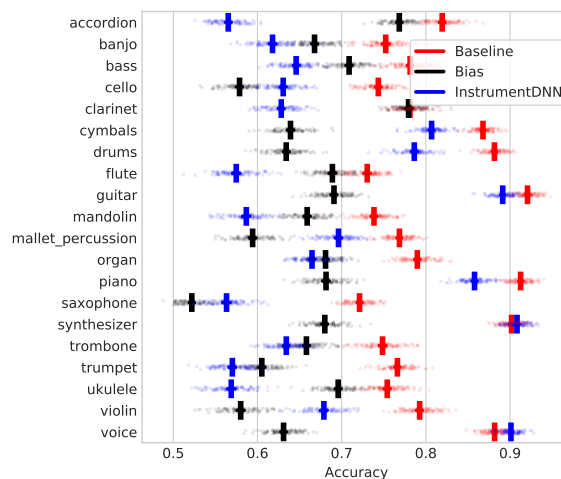


Figure 8. Accuracy of the Random Forest baseline (red), InstrumentDNN (blue), and the dataset bias (black). The RandomForest was trained on OpenMIC-2018, while InstrumentDNN was trained on AudioSet.

⁶ <http://about.zenodo.org/>

6. REFERENCES

- [1] Giuseppe Bandiera, Oriol Romani Picas, Hiroshi Tokuda, Wataru Hariya, Koji Oishi, and Xavier Serra. Good-sounds. org: A framework to explore goodness in instrumental sounds. In *Proceedings of the 17th International Society for Music Information Retrieval Conference*, pages 414–419, 2016.
- [2] Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 591–596, 2011.
- [3] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *ISMIR*, volume 14, pages 155–160, 2014.
- [4] Juan J Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera. A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. In *ISMIR*, pages 559–564, 2012.
- [5] Mark Cartwright, Ayanna Seals, Justin Salamon, Alex Williams, Stefanie Mikloska, Duncan MacConnell, E Law, J Bello, and O Nov. Seeing sound: Investigating the effects of visualizations and complexity on crowd-sourced audio annotations. *Proceedings of the ACM on Human-Computer Interaction*, 1(1), 2017.
- [6] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.
- [7] Creative Commons. About the licenses. 2015. <https://creativecommons.org/about/>.
- [8] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. FMA: A dataset for music analysis. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 316–323, 2017.
- [9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- [10] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders. 2017.
- [11] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 776–780. IEEE, 2017.
- [12] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Music genre database and musical instrument sound database. 2003.
- [13] Aren Jansen, Jort F Gemmeke, Daniel PW Ellis, Xiaofeng Liu, Wade Lawrence, and Dylan Freedman. Large-scale audio event discovery in one million youtube videos. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 786–790. IEEE, 2017.
- [14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [15] B. McFee, E.J. Humphrey, and J. Urbano. A plan for sustainable MIR evaluation. In *17th International Society for Music Information Retrieval Conference, ISMIR*, 2016.
- [16] George A Miller. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
- [17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2016.
- [19] John Thickstun, Zaid Harchaoui, and Sham Kakade. Learning features of music from scratch. In *International Conference on Learning Representations*, 2017.

FROM LABELED TO UNLABELED DATA – ON THE DATA CHALLENGE IN AUTOMATIC DRUM TRANSCRIPTION

Chih-Wei Wu, Alexander Lerch

Georgia Institute of Technology, Center for Music Technology

{ cwu307, alexander.lerch }@gatech.edu

ABSTRACT

Automatic Drum Transcription (ADT), like many other music information retrieval tasks, has made progress in the past years through the integration of machine learning and audio signal processing techniques. However, with the increasing popularity of data-hungry approaches such as deep learning, the insufficient amount of data becomes more and more a challenge that concerns the generality of the resulting models and the validity of the evaluation. To address this challenge in ADT, this paper first examines the existing labeled datasets and how representative they are of the research problem. Next, possibilities of using unlabeled data to improve general ADT systems are explored. Specifically, two paradigms that harness information from unlabeled data, namely feature learning and student-teacher learning, are applied to two major types of ADT systems. All systems are evaluated on four different drum datasets. The results highlight the necessity of more and larger annotated datasets and indicate the feasibility of exploiting unlabeled data for improving ADT systems.

1. INTRODUCTION

Automatic drum transcription (ADT), a sub-task of Automatic Music Transcription (AMT) [2] that concerns the extraction of drum events from music signals, witnesses a growth in data-driven approaches such as deep learning in recent years [24, 25, 31–33]. The majority of these ADT studies use the popular ENST-Drums dataset [11] for development by splitting the dataset into different subsets for training, validation, and testing purposes. Nevertheless, the limited amount of labeled data and its potential impact on ADT systems are rarely discussed. The heavy reliance on one dataset raises two major concerns: (i) the model could easily overfit the data, which questions its generality, and (ii) the evaluation results could be overly optimistic due to the small sample size of the split. To avoid these pitfalls, larger datasets and cross-dataset evaluation are necessary. This need has been identified by researchers and has been addressed with newly released annotated datasets such as MDB-Drums [26] and RBMA [33]. These new

data enable us to revisit ENST-Drums and re-examine the representativeness of this widely-used dataset through a unified comparison.

Motivated by the above mentioned issues concerning the data in ADT, this paper aims to address the challenge from two different angles, (i) examining the effectiveness of the existing datasets and (ii) investigating additional resources (e.g., unlabeled data) and techniques for supporting the development of general ADT systems. The contributions of this work include: first, the examination of four different datasets, highlighting the importance of data diversity. Second, the evaluation of two paradigms for integrating unlabeled data to two major types of ADT systems. Third, the demonstration of potential improvements of both types of ADT systems on different drum instruments using unlabeled data.

2. RELATED WORK

2.1 Automatic Drum Transcription

The task of automatic drum transcription can be described as converting drum related audio events into music notation. Most of the early ADT systems, as summarized by FitzGerald and Paulus [9], detect onsets of HiHat (HH), Bass Drum (BD), and Snare Drum (SD) in drum only recordings. Recently, this focus has shifted towards transcribing drums in polyphonic mixtures comprised of both percussive and melodic instruments. Following these conventions, this paper defines the ADT task as detecting HH, BD, and SD in polyphonic mixtures.

Generally speaking, the existing ADT systems can be categorized into four types according to the literature [12, 19]. These are (i) *Segment and Classify*: following the standard pattern recognition pipeline, these approaches extract audio features from detected onset locations and classifies them with pre-trained models; this is a popular approach with many proposed systems using different combinations of classifiers and features [10, 12, 27, 28], (ii) *Separate and Detect*: deriving activation functions from recordings to represent the activities of each drum, these systems subsequently perform onset detection on these activation functions to locate drum hits; approaches include matrix factorization methods such as Non-negative Matrix Factorization (NMF) [6, 23, 35] and deep-learning-based methods such as Recurrent Neural Networks (RNNs) [24, 31, 32] and Convolutional Neural Networks (CNNs) [25, 33], (iii) *Match and Adapt*: identifying drum events by comparing with a set of



© Chih-Wei Wu, Alexander Lerch. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Chih-Wei Wu, Alexander Lerch. “From labeled to unlabeled data – on the data challenge in automatic drum transcription”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

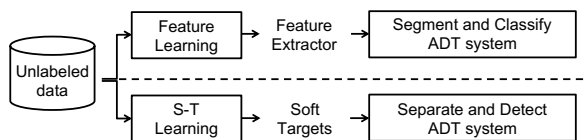


Figure 1. The overview of the evaluated paradigms for integrating unlabeled data to two major ADT approaches

pre-defined templates, these systems often iteratively update the templates [38], and (iv) *HMM-based Recognition*: modeling the temporal connections between drum events using probabilistic models such as Hidden Markov Models (HMMs), these models try to identify the underlying drum sequence by using the Viterbi algorithm [7, 20].

To date, the majority of the existing ADT systems fall into the categories of *Segment and Classify* and *Separate and Detect*. Both these types of systems, despite having fundamental differences, use data-driven methods and face the challenge described in Sect. 1. Therefore, in this paper, we considered both types of systems in our experiments.

2.2 Learning from Unlabeled Data

To address the data challenge in MIR tasks, techniques that build upon the existing labeled data have been proposed. For example, in *transfer learning* [4], a deep neural network trained on a task that has sufficient data can be used to derive features for another task with limited data. This method alleviates the data insufficiency by re-using the effective models in the similar domains. *Data augmentation*, a technique to increase diversity of training data through music-related deformations (e.g., time-stretching, pitch shifting, or distortion) and synthesis, has been successfully applied to MIR tasks [18] and in ADT specifically [32, 36]. However, these techniques still require a reasonably sized correctly annotated dataset as a starting point, which remains a challenge in certain scenarios.

Another direction for addressing the data scarcity is to use unlabeled data. Intuitively, a large collection of unlabeled data can be helpful in deriving more generalized features. This is the main concept of unsupervised *feature learning*, and it can be implemented with algorithms such as Sparse Coding [22], Deep Belief Networks [13], and Auto-encoders [17]. More recently, the *student-teacher learning* paradigm has also emerged as an interesting concept to incorporate unlabeled data. Referred by Hinton et al. as “knowledge distillation” [14], this paradigm transfers the knowledge of a teacher model to a student model using the soft-targets generated by the teacher. As opposed to learning from the hard targets (i.e., ground truth), the student learns from the “dark knowledge” residing in the soft-targets, which can be created using either labeled or unlabeled data [15]. A successful student model can reduce the complexity of the original teacher model without significant performance loss. Several studies also report superior performance of the student models [5, 34, 37]. Overall, methods that work directly with unlabeled data obviously have less dependency on existing labeled data and have

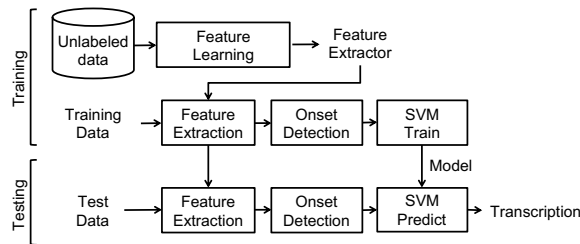


Figure 2. The flowchart of the feature learning paradigm for ADT

higher potential to be applicable to more tasks.

3. METHOD

3.1 Overview

To connect general ADT systems to the abundant resources of unlabeled data, this paper investigates the application of *feature learning* and *student-teacher learning* to *Segment and Classify*-based and *Separate and Detect*-based ADT systems, respectively. Figure 1 shows the two paradigms for integrating unlabeled data to ADT systems as investigated in this paper. The feature learning paradigm is designed for *Segment and Classify*-based ADT systems. In this paradigm, the unlabeled data is used to derive a feature extractor using an unsupervised feature learning algorithm. The resulting feature extractor is then integrated into a generic *Segment and Classify* ADT framework. The student-teacher learning paradigm is suitable for *Separate and Detect*-based ADT systems. This paradigm uses teacher models and unlabeled data to generate soft-targets; these soft-targets play the important role of connecting any *Separate and Detect*-based system with unlabeled data and enable the training of the student model. In the following sections, more details of both paradigms are presented.

3.2 Feature Learning

The flowchart in Fig. 2 shows the feature learning paradigm for ADT, including both training and testing. The training phase starts with the training of a feature extractor using the unlabeled data. Specifically, we use a Convolutional Auto-encoder (CAE) as the feature extractor. A generic *Segment and Classify*-based ADT system is then constructed with the following steps: first, the features are extracted from the audio signals using the pre-trained feature extractor. Second, the onset locations are determined by using the ground truth annotations while training. Finally, the feature vectors around the onset locations are collected and used to train three binary classifiers for HH, BD, and SD, respectively. The classifiers used in this paper are Support Vector Machines (SVMs). In the testing phase, the same pipeline is followed except for the onset detection step, which uses an onset detector instead of the ground truth locations. Finally, the presence of each drum can be predicted using the pre-trained SVMs.

The architecture of the CAE is shown in Fig. 3. The input X of the CAE is a Mel-spectrogram, and the output

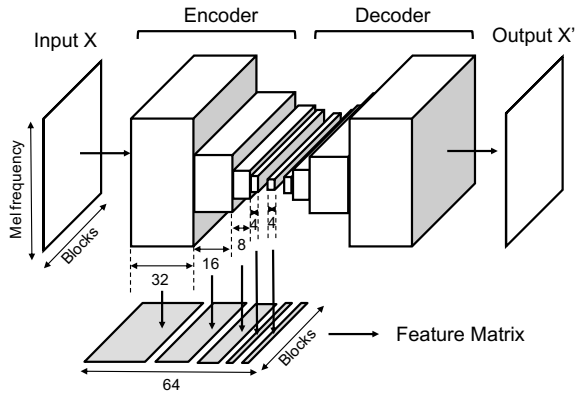


Figure 3. The architecture of the proposed CAE for unsupervised feature learning. The input X is a $128 \times N$ Mel-spectrogram.

X' is the reconstruction of X . The encoder consists of four convolutional layers with $\{32, 16, 8, 4\}$ channels of 3×3 kernels, accordingly. Each convolutional layer is used by a batch normalization layer and a max-pooling layer of $(2, 1)$. This design maintains the temporal resolution, allowing the extraction of block-wise features. The bottleneck layer is also a convolutional layer with 4 channels of 3×3 kernels. All non-linear units are Rectified Linear Units (ReLU). The structure of the decoder is symmetric to the encoder with the max-pooling layers replaced by the up-sampling layers. The CAE is trained to minimize the Mean Squared Error (MSE) between X and X' using a gradient-descent-based optimization process, and the number of training epochs is 30.

The feature extraction process, as shown in Fig. 3, is inspired by the method proposed by Choi et al. [4]: first, the intermediate activation maps from all the layers in the encoder (including the bottleneck layer) are computed. Next, average pooling is performed on these maps across the Mel-frequency axis. Finally, these outputs are stacked into a $64 \times N$ feature matrix, where N is the number of blocks. To derive the final feature vector at each block, the feature vectors from the current block and the following two blocks are spliced together to capture the temporal variations of the event. This leads to a final feature vector with a dimensionality $d = 3 \times F$, in which F is the number of features (i.e., 64).

In addition to the learned features, a set of baseline features consisting of 20 Mel Frequency Cepstral Coefficients (MFCCs) and their first and second derivatives is also included in this paradigm. As a result, the baseline feature vector has a dimensionality $d = 3 \times 60 = 180$ after the feature splicing.

3.3 Student-Teacher Learning

Figure 4 shows the flowchart of the student-teacher learning paradigm for ADT. In the training phase, the teacher models are used to analyze the unlabeled data and generate the soft-targets. These soft-targets, used as pseudo ground truth to

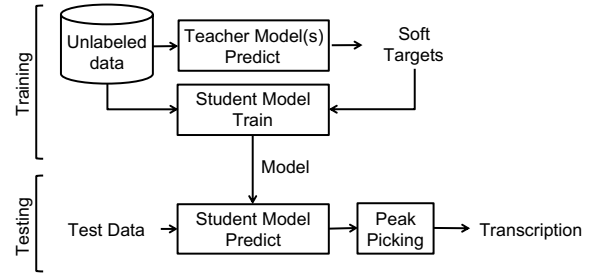


Figure 4. The flowchart of the student-teacher learning paradigm for ADT

train a student model, contain the activation functions for the different drums. When multiple teachers are present, the student model can be trained by iteratively passing the unlabeled data and its corresponding soft-targets from each teacher. The student model is trained by minimizing the MSE between the soft-targets and the model outputs. In the testing phase, the trained student model processes the test data and generates the corresponding activation functions. The estimated locations of drum hits are identified with a simple peak picking process.

The model architecture, configuration, and parametrization of this evaluated paradigm generally follows the setup described in [37]. This includes two teacher models based on Partially-Fixed NMF (PFNMF) [35] and one student model using a fully-connected, feed-forward Deep Neural Network (DNN). The soft-targets are scaled to a numerical range between 0 and 1 using min-max scaling across the training data for each instrument in order to ensure their compatibility with the outputs from the student DNN.

3.4 Implementation

The main input representations for both paradigms are derived from the magnitude spectrogram of the Short Time Fourier Transform (STFT), which is computed using a block size of 2048 and a hop size of 512 samples with Hann window. All of the audio signals are normalized to a range between 1 and -1, down-mixed to mono, and resampled to 44.1 kHz prior to the computation of STFT.

For the feature learning paradigm, both the Mel-spectrogram in dB scale with 128 bins and the MFCCs are computed using librosa,¹ a Python library for audio signal processing. The onset detection is implemented using the *CNNOnsetProcessor* from Madmom.² Additionally, the implementation of Linear SVMs from scikit-learn,³ a Python library for machine learning, is used. A grid search on the penalty parameter C within $\{0.1, 1, 10, 100, 1000\}$ is performed to optimize the performance of the SVMs.

For student-teacher learning paradigm, the teacher models are implemented using the PFNMF function from Nmf-DrumToolbox.⁴ The peak-picking parameters are set to the same as in the original paper [37].

¹ <https://librosa.github.io>, last access 2018/03/27

² <https://madmom.readthedocs.io/en/latest/>, last access 2018/03/27

³ <http://scikit-learn.org/stable/>, last access 2018/03/27

⁴ <https://github.com/cwu307/NmfDrumToolbox>, last access 2018/03/27

The neural networks in both paradigms are implemented using Keras⁵ and the Tensorflow [1] backend. The weights are randomly initialized with normal distributions, and the parameters of the ADAM optimizer are set to default. The source code used in this paper is available on Github.⁶

4. EXPERIMENT

4.1 Unlabeled Data

The unlabeled dataset in this paper is built using the source code provided in [37]; this tool allows the compilation of a list of songs from the Billboard Chart⁷ and the retrieval of these songs from Youtube. This dataset consists of six musical genres, including R&B/HipHop, Pop, Rock, Latin, Alternative, and Dance/Electronic. Each genre has 1900 songs, which leads to a collection of 11400 songs. All the songs are cross-checked for duplicates and converted to mp3 format with a sampling rate of 44.1 kHz. In our experiments, this dataset is further split into training, validation, and testing set with a percentage of 70%, 15%, and 15%, respectively. To speed up the process while maintaining the diversity, only a 30 s segment is extracted from each song for training. The segment starts in the middle of the song to avoid potential inactivity at the beginning. As a result, the entire training set has a total duration of 66.5 hrs, which is significantly larger than any existing ADT dataset. The list of songs and links are available on Github.⁸

4.2 Labeled Data

In this paper, four different labeled datasets featuring polyphonic mixtures are used: (i) the popular ENST-Drums (referred to as ENST) [11], (ii) the MIREX 2005 (referred to as m2005), (iii) the MDB-Drums (referred to as MDB) [26], and (iv) the RBMA dataset [33]. The latter three public sets have been used in the 2017 Music Information Retrieval Evaluation eXchange (MIREX)⁹ drum transcription task.

ENST minus one subset consists of 64 recordings performed by three different drummers on their own drum kits. The average duration of the recordings is 55 s. These recordings feature different musical genres and playing styles, and the multi-track files are available for remixing. In this paper, the accompaniments are mixed with their corresponding drum tracks using a scaling factor of 1/3 and 2/3, respectively. This setup is consistent with several previous studies [24, 31, 35].

m2005 was originally collected for the first MIREX drum transcription task in 2005 and recently made available for MIREX 2017 drum transcription task participants. The public set includes 23 recordings contributed from all the participants of MIREX 2005. While covering a variety of musical genres, J-pop has the highest presence in this

dataset with 10 recordings. The average duration of this dataset is 125 s.

MDB consists of 23 recordings of the MusicDelta subset from the MEDLEYDB dataset [3]. These recordings include a variety of musical genres such as Rock, Country, Disco, Reggae, and Jazz. The average duration of the recordings is 54 s. Similar to *ENST*, this dataset contains multi-track files as well as the full mixtures. In this paper, we use the full-mixtures directly without any adjustment of the mixing levels.

RBMA was released as part of the public set for the MIREX 2017 drum transcription task. This public set includes 27 recordings featuring mostly Electronic Dance Music (EDM). The average duration of the tracks is 230 s. Since this dataset focuses on electronic music, it contains electronic drum sounds that can be distinctively different from the other three datasets.

In total, there are 137 files with annotations available for evaluation. All files have a sampling rate of 44.1 kHz.

4.3 Metrics

The evaluation metrics in this paper are Precision (P), Recall (R), and F-measure (F). Only the averaged F-measure is reported due to the limited space. These metrics are implemented using *mir_eval*, a Python library of common MIR metrics [21]. To determine whether an onset is a match with the ground truth, a tolerance window of 50 ms on both sides is used. This setting is consistent with the literature [12, 24, 35], although some authors use smaller tolerance windows such as 30 ms [20] and 20 ms [32].

4.4 Experiment Setup

This paper evaluates 9 ADT systems, comprising 4 systems for the feature learning paradigm and 5 systems for the student-teacher learning paradigm. The configurations of these systems are described as follows:

For the feature learning paradigm, the 4 systems are differentiated by their features. These features are:

- (i) MFCC: this set of features has shown its effectiveness in previous ADT studies [20, 27, 29]. Therefore, it is included as a baseline.
- (ii) CONV-RANDOM: this set of features is extracted using the proposed CAE architecture with all the weights randomly initialized without further training. This is another baseline inspired by [4] to serve as a sanity check for the effectiveness of the unsupervised training process.
- (iii) CONV-AE: this is the set of features extracted from the proposed CAE after training. During the training procedure, the original input is used as the target for optimization. In other words, the CAE is trained to reconstruct the input.
- (iv) CONV-DAE: this set of features is similar to CONV-AE except for the optimization target. In this case, a processed input is used as the target. Specifically, the percussive component from the Harmonic Percussive Source Separation (HPSS) [8] algorithm is used, and the CAE is trained to approximate the percussive

⁵ <https://keras.io>, last access 2018/03/27

⁶ https://github.com/cwu307/ADT_with_unlabeledData, last access 2018/06/14

⁷ <https://www.billboard.com/charts>, last access 2018/03/27

⁸ <https://github.com/cwu307/unlabeledDrumDataset>, last access 2018/06/14

⁹ <http://www.music-ir.org/mirex/wiki/2017>, last access 2018/03/27

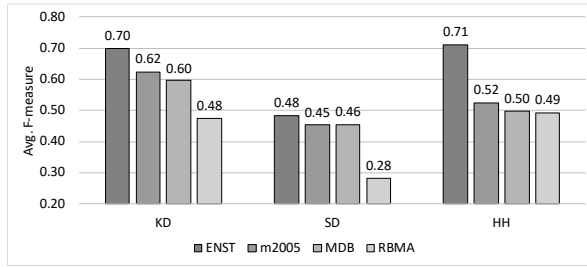


Figure 5. The evaluation results of all labeled datasets with averaged F-measure across all systems.

component. This configuration is inspired by the concept of the Denoising Autoencoder (DAE) [30] and is designed to encourage the extraction of drum-related features.

The teacher models for student-teacher learning paradigm are described in [37]. The 3 student models can be differentiated by their training data. The systems are:

- (i) PFNMF (SMT): a teacher PFNMF initialized with the drum templates extracted from the IDMT-SMT-Drum dataset [6].
- (ii) PFNMF (200D): a teacher PFNMF initialized with the drum templates extracted from the 200 Drum Machine dataset.¹⁰
- (iii) FC-200: a fully-connected student DNN trained with a subset of the unlabeled dataset, which consists of 200 randomly selected songs from each genre.
- (iv) FC-ALL: a fully-connected student DNN trained with all the songs from all genres.
- (v) FC-ALL (ALT): a fully connected student DNN trained with all the songs from only the “Alternative” genre. This particular genre is selected for its superior performance in preliminary tests.

Based on these 9 systems, the following experiments are conducted:

- E1: Experiment 1** aims to examine the variance of the labeled datasets. For each dataset, the averaged F-measures across all 9 systems are reported.
- E2: Experiment 2** aims to evaluate the usefulness of unlabeled data for *Segment and Classify*-based ADT systems using the feature learning paradigm. For each system, the averaged F-measures across all the datasets are reported.
- E3: Experiment 3** aims to evaluate the usefulness of unlabeled data for *Separate and Detect*-based ADT systems using the student-teacher learning paradigm. For each system, the averaged F-measures across all the datasets are reported.

Note that for the feature learning paradigm, a cross-dataset validation process is performed (e.g., train on three datasets and test on the remaining one) in order to train the binary classifiers (see Sect. 3.2). For student-teacher

¹⁰ <http://www.hexawe.net/mess/200.Drum.Machines/>, last access 2018/03/27

Experiments		Averaged F-measure		
Role	System	HH	BD	SD
Baseline	MFCC	0.61	0.62	0.40
Baseline	CONV-RANDOM	0.61	0.54	0.39
Evaluated	CONV-AE	0.61	0.62	0.42
Evaluated	CONV-DAE	0.61	0.61	0.42

Table 1. Evaluation results of the feature-learning-paradigm-based systems.

learning paradigm, since the student model does not need additional labeled data for training so that a cross-dataset validation is unnecessary.

4.5 Results

Figure 5 shows the evaluation result of **E1**. On average, all systems tend to perform the best on *ENST* and the worst on *RBMA*. For some instruments, this gap can be as large as 22% in F-measure. There are two possible reasons for the good performance on *ENST*. First, as many ADT systems, including *Segment and Classify*-based and *Separate and Detect*-based, have been developed and evaluated on *ENST*, there could be potential bias towards this dataset. Second, the *ENST* dataset might be relatively simple compared to the others. A closer examination of the dataset shows a lack of singing voices and the dominance of MIDI synthesized accompaniments, which could potentially over-simplify the ADT problem. The relative poor performance on the *RBMA* dataset might be related to its focus on EDM; the electronic drum sounds with strong audio effects could possibly increase the difficulty for ADT. This seems to be especially true in case of the *SD*. Overall, the results show that the evaluated systems leave much room for optimization; since many of the parameters in these systems are not extensively tuned, this result is to be expected. However, this also reflects the challenge of building an ADT system that is easily generalizable.

The results of **E2** are shown in Table 1. The following trends can be observed: first, the unlabeled data seems to be helpful in *Segment and Classify*-based ADT systems. A direct comparison between CONV-AE and MFCC shows that the features learned from unlabeled data seem to slightly improve for *SD* while achieving equal performance on *HH* and *BD*. Second, the unsupervised training process is useful for deriving better features. Compared to CONV-RANDOM, both CONV-AE and CONV-DAE show improvements on nearly all instruments, indicating the advantage of the training process. Third, the DAE-inspired training process does not lead to improvements for ADT. This is shown by the almost equivalent results from CONV-AE and CONV-DAE. Since HPSS also introduces artifacts, it might not be the most ideal method for this task; experimentation with other source separation algorithms might provide more insights.

Table 3 shows the results of **E3**. The general trends can be summarized as follows: first, the student-teacher learning seems to be useful for *Separate and Detect*-based ADT systems as all students show a noticeable improvement on *HH* over the teacher models. This observation

Compared Systems		Inst.	Paradigm	Improvement		Deterioration	
Test	Ref			# Files	F-measure Gain	# Files	F-measure Loss
CONV-AE	MFCC	SD	Feature Learning	70/137	6.5%	40/137	-4.6%
FC-200	PFNMF (SMT)	HH	S-T Learning	78/137	13.8%	44/137	-7.6%

Table 2. Significance check of the most improved pair from each paradigm.

Experiments		Averaged F-measure		
Role	System	HH	BD	SD
Teacher	PFNMF (SMT)	0.47	0.61	0.45
Teacher	PFNMF (200D)	0.47	0.67	0.40
Student	FC-200	0.56	0.57	0.44
Student	FC-ALL	0.53	0.59	0.42
Student	FC-ALL (ALT)	0.55	0.58	0.44

Table 3. Evaluation results of the student-teacher-paradigm-based systems. The performance of the teacher models are the baseline.

consolidates the preliminary finding reported in [37]. Second, more unlabeled data do not necessarily lead to better results. For example, FC-200 and FC-ALL (ALT) both outperform FC-ALL on HH and SD. Since the student model is a simple feed-forward DNN, the lack of model capacity and temporal information could limit its potential for further improvement as the data size grows. Experiments using other student models (e.g., CNNs and RNNs) are necessary for confirmation. Third, the student models seem to struggle on BD. A detailed examination on the individual results from each dataset shows that teachers and students are mostly comparable on BD except for *RBMA*. This is possibly due to the challenging nature of *RBMA* as discussed in **E1**. However, further investigation is needed before drawing conclusions.

The results of **E2** and **E3** show that feature learning and student-teacher learning paradigms are able to improve the performance on SD and HH, respectively. In light of these results, an interesting question is: “Are these improvements significant?” In an attempt to answer this question, two pairs of systems are selected for further analysis. Each pair consists of the best baseline and the best evaluated system of each paradigm. A t-test is performed on each pair by comparing their results on all 137 files. Both pairs have shown significant improvements with $p \ll 0.0014$ for both t-tests. Furthermore, the number of improved and deteriorated files is calculated. The results, shown in Table 2, show a positive trend: the number of improved files is, in both cases, greater than the number of deteriorated files. Moreover, the averaged F-measure gains are also higher than the averaged F-measure loss for both pairs.

From Table 2, it can be observed that the improvements on HH from the student-teacher learning paradigm seems to be more substantial. To further investigate the cause of this improvement, one example from the *ENST* dataset, which has the largest F-measure gain among all files, is selected. The HH activation functions generated from both teacher and student model are shown in Fig. 6. Compared to the teacher’s activation function, the student’s activation

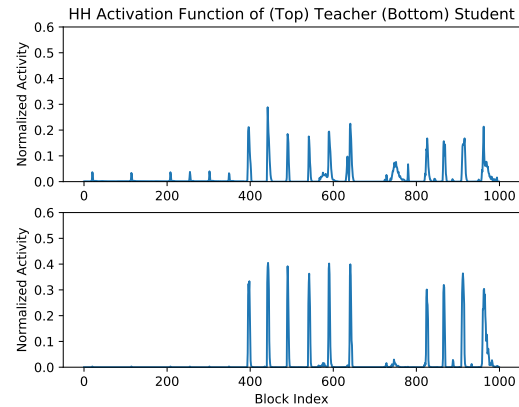


Figure 6. Example of the (top) teacher’s and (bottom) student’s HH activation function in comparison.

function is sharper and less noisy, demonstrating the benefit of this paradigm.

5. CONCLUSION

We discussed the data challenge in ADT and investigated two approaches to address this challenge by considering both labeled and unlabeled data. First, we compared system performance on multiple existing labeled datasets in an unified setting. The results indicate a potential bias of relying on one dataset and highlight the necessity of including more datasets in the future ADT evaluation. Furthermore, we evaluated the usefulness of unlabeled data for two major types of ADT systems via two different learning paradigms, feature learning and the student-teacher learning approach. For both paradigms, we got encouraging (and statistically significant) results demonstrating the potential of achieving better performance than the baseline systems on different drum instruments.

These results, while suggesting the need for additional labeled data in the field of ADT, also encourage the exploration of incorporating unlabeled data in the training. Possible future directions include (i) the evaluation of various methods for unsupervised feature learning such as Sparse Coding [22] and Deep Belief Networks [13], (ii) the evaluation of different combinations of teacher and student models, for example, the combination of different types of DNN either as teachers or students; the identification of suitable architectures for these roles could also be an interesting direction, and (iii) the application of outlier detection [16] approaches to filter out noisy unlabeled data.

6. REFERENCES

- [1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, and Google Brain. TensorFlow: A System for Large-Scale Machine Learning TensorFlow: A system for large-scale machine learning. In *Proc. of USENIX Symp. on Operating Systems Design and Implementation (OSDI)*, pages 265–284, 2016.
- [2] Emmanouil Benetos, Simon Dixon, Dimitrios Gianoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, jul 2013.
- [3] Rachel Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Bello. MedleyDB: a multitrack dataset for annotation-intensive MIR research. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2014.
- [4] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Transfer learning for music classification and regression tasks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- [5] Jia Cui, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, Tom Sercu, Kartik Audhkhasi, Abhinav Sethy, Markus Nussbaum-Thom, and Andrew Rosenberg. Knowledge Distillation Across Ensembles of Multilingual Models for Low-resource Languages. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4825–4829, 2017.
- [6] Christian Dittmar and Daniel Gärtner. Real-time transcription and separation of drum recordings based on NMF decomposition. In *Proc. of the International Conference on Digital Audio Effects (DAFx)*, pages 187–194, Erlangen, Germany, September 2014.
- [7] Georgi Dzhabazov. Towards a drum transcription system aware of bar position. In *Proc. Audio Engineering Society Conference on Semantic Audio (AES)*, London, UK, Jan 2014.
- [8] Derry Fitzgerald. Harmonic / Percussive Separation Using Median Filtering. In *Proc. of International Conference on Digital Audio Effects (DAFx)*, 2010.
- [9] Derry FitzGerald and Jouni Paulus. Unpitched percussion transcription. In *Signal Processing Methods for Music Transcription*, pages 131–162. Springer, 2006.
- [10] Nicolai Gajhede, Oliver Beck, and Hendrik Purwins. Convolutional Neural Networks with Batch Normalization for Classifying Hi-hat, Snare, and Bass Percussion Sound Samples. In *Proc. of the Audio Mostly*, pages 111–115, 2016.
- [11] Olivier Gillet and Gaël Richard. Enst-drums: an extensive audio-visual database for drum signals processing. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2006.
- [12] Olivier Gillet and Gaël Richard. Transcription and separation of drum signals from polyphonic music. *IEEE Transactions on Audio, Speech and Language Processing*, 16(3):529–540, 2008.
- [13] Philippe Hamel and Douglas Eck. Learning Features from Music Audio with Deep Belief Networks. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, pages 339–344, 2010.
- [14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv:1503.02531*, pages 1–9, 2015.
- [15] Jinyu Li, Rui Zhao, Jui Ting Huang, and Yifan Gong. Learning small-size DNN with output-distribution-based criteria. In *Proc. of the Conference of the International Speech Communication Association (INTER-SPEECH)*, pages 1910–1914, 2014.
- [16] Yen-Cheng Lu, Chih-Wei Wu, Chang-Tien Lu, and Alexander Lerch. Automatic outlier detection in music genre datasets. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 101–107, 2016.
- [17] Jonathan Masci, Ueli Meier, Dan Cirean, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Proc. of International Conference on Artificial Neural Networks (ICANN)*, 2011.
- [18] Brian Mcfee, Eric J Humphrey, and Juan P Bello. A software framework for musical data augmentation. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 248–254, 2015.
- [19] Jouni Paulus. *Signal Processing Methods for Drum Transcription and Music Structure Analysis*. PhD thesis, Tampere University of Technology, Tampere, Finland, 2009.
- [20] Jouni Paulus and Anssi Klapuri. Drum sound detection in polyphonic music with hidden markov models. *EURASIP Journal on Audio, Speech, and Music Processing*, (14), 2009.
- [21] Colin Raffel, Brian Mcfee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. mir_eval: A Transparent Implementation of Common MIR Metrics. *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 367–372, 2014.

- [22] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 759–766, 2007.
- [23] Axel Roebel, Jordi Pons, Marco Liuni, and Mathieu Lagrange. On Automatic Drum Transcription Using Non-Negative Matrix Deconvolution and Itakura Saito Divergence. In *Proc. of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*, 2015.
- [24] Carl Southall, Ryan Stables, and Jason Hockman. Automatic Drum Transcription Using Bi-Directional Recurrent Neural Networks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016.
- [25] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 606–612, 2017.
- [26] Carl Southall, Chih-Wei Wu, Alexander Lerch, and Jason Hockman. MDB DRUMS - an annotated subset of medleydb for automatic drum transcription. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)(Late-breaking Demo)*, 2017.
- [27] Vinícius M. A. Souza, Gustavo E. A. P. A. Batista, and Nilson E. Souza-Filho. Automatic classification of drum sounds with indefinite pitch. In *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, Killarney, Ireland, Jul 2015.
- [28] Dirk Van Steelant, Koen Tanghe, Sven Degroove, Bernard De Baets, Marc Leman, and Jean-Pierre Martens. Support vector machines for bass and snare drum recognition. In *Classification – the Ubiquitous Challenge*, pages 616–623. Springer, 2005.
- [29] Lucas Thompson, Matthias Mauch, and Simon Dixon. Drum Transcription via Classification of Bar-Level Rhythmic Patterns. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2014.
- [30] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proc. of the International Conference on Machine Learning (ICML)*, 2008.
- [31] Richard. Vogl, Matthias Dorfer, and Peter Knees. Recurrent Neural Networks for Drum Transcription. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 730–736, 2016.
- [32] Richard Vogl, Matthias Dorfer, and Peter Knees. Drum Transcription From Polyphonic Music With Recurrent Neural Networks. In *Proc. of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 201–205, 2017.
- [33] Richard Vogl, Matthias Dorfer, Gerhard Widmer, and Peter Knees. Drum Transcription Via Joint Beat and Drum Modeling Using Convolutional Recurrent Neural Networks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 150–157, 2017.
- [34] Shinji Watanabe, Takaaki Hori, Jonathan L. Roux, and John R. Hershey. Student-Teacher Network Learning with Enhanced Features. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5275–5279, 2017.
- [35] Chih-Wei Wu and Alexander Lerch. Drum transcription using partially fixed non-negative matrix factorization with template adaptation. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 257–263, 2015.
- [36] Chih-Wei Wu and Alexander Lerch. On drum playing technique detection in polyphonic mixtures. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, pages 218–224, 2016.
- [37] Chih-Wei Wu and Alexander Lerch. Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 613–620, 2017.
- [38] Kazuyoshi Yoshii, Masataka Goto, and Hiroshi G. Okuno. Drum sound recognition for polyphonic audio signals by adaptation and matching of spectrogram templates with harmonic structure suppression. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):333–345, 2007.

GUITARSET: A DATASET FOR GUITAR TRANSCRIPTION

Qingyang Xi¹ Rachel M. Bittner¹ Johan Pauwels²
Xuzhou Ye¹ Juan P. Bello¹

¹ Music and Audio Research Lab, New York University, USA

² Center for Digital Music, Queen Mary University of London, UK

tom.xi@nyu.edu

ABSTRACT

The guitar is a popular instrument for a variety of reasons, including its ability to produce polyphonic sound and its musical versatility. The resulting variability of sounds, however, poses significant challenges to automated methods for analyzing guitar recordings. As data driven methods become increasingly popular for difficult problems like guitar transcription, sets of labeled audio data are highly valuable resources. In this paper we present GuitarSet, a dataset that provides high quality guitar recordings alongside rich annotations and metadata. In particular, by recording guitars using a hexaphonic pickup, we are able to not only provide recordings of the individual strings but also to largely automate the expensive annotation process. The dataset contains recordings of a variety of musical excerpts played on an acoustic guitar, along with time-aligned annotations of string and fret positions, chords, beats, downbeats, and playing style. We conclude with an analysis of new challenges presented by this data, and see that it is interesting for a wide variety of tasks in addition to guitar transcription, including performance analysis, beat/downbeat tracking, and chord estimation.

1. INTRODUCTION

Well-annotated audio files are key to MIR research. They are necessary both for evaluating algorithm performance and for developing models. For time-varying musical information such as notes in a polyphonic context, the process of creating accurate annotations can be an especially difficult and slow process. For monophonic audio, there are software tools, such as Tony [12], built to facilitate the manual annotation process by first providing an estimate and allowing the user to manually correct the mistakes. However, there is no equivalent tool for polyphonic audio, and the accuracy of pitch estimation methods on polyphonic audio is significantly worse than for monophonic audio.

Recently, several methods have been developed to address the problem of creating pitch annotations. Most notably, Su and Yang’s work [22] provides an efficient way of generating note-level annotations for recordings of polyphonic music by utilizing the midi-keyboard as an annotation interface. An alternative approach was proposed that uses an analysis-synthesis framework to generate annotations by re-synthesizing estimates [18]. However, these methods are insufficient when applied to guitar recordings. In the midi-keyboard approach, it would be very difficult for a keyboard player to replicate note-by-note what a guitarist is playing. The analysis-synthesis approach requires the analysis (i.e. estimate of the correct notes) to be reasonably close to the ground truth in order to generate realistic sounding audio; unfortunately, the existing transcription algorithms perform woefully badly on polyphonic solo guitar recordings. Unsurprisingly, there is no sizable database of guitar recordings with note-level annotations and realistic guitar playing.

In this paper, we present GuitarSet: a sizable dataset of richly annotated, realistic guitar recordings. We describe our data collection and annotation process in detail and introduce our solution for efficiently creating note-level annotations. Our solution relies on the use of an acoustic guitar with a *hexaphonic pickup*, which outputs one channel of audio signal per guitar string; as well as custom annotation tools. This effectively turns polyphonic transcription into monophonic transcription. We conclude with an analysis of new challenges presented by this data, and see that it is interesting for a wide variety of tasks in addition to guitar transcription, including performance analysis, beat/downbeat tracking, and chord estimation. The dataset (audio and annotations) and the code used to generate the annotations are made freely available online.¹

2. RELATED WORK

A handful of datasets exist for polyphonic instrument transcription. The MAPS dataset [7] contains a large collection of transcribed piano notes, chords, and pieces (using a Disklavier), recorded in different acoustic conditions. Similarly, the UMA-Piano [2] dataset contains all possible combinations of notes at varying dynamics. These datasets have been critical to the development of automated piano transcription methods; Sigtia’s deep-learning powered



© Qingyang Xi, Rachel Bittner, Johan Pauwels, Xuzhou Ye, Juan Bello. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Qingyang Xi, Rachel Bittner, Johan Pauwels, Xuzhou Ye, Juan Bello. “GuitarSet: A Dataset for Guitar Transcription”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ <https://github.com/marl/GuitarSet>

piano transcription algorithm [20] and Ewert’s algorithm based on non-negative matrix deconvolution [8] are just two of many data driven algorithms that rely on the MAPS dataset. More recently, efforts devoted to historic preservation of player piano rolls also provide new ways of extending transcription datasets for piano music [19].

For guitar, the Guitar Playing Techniques dataset [23] contains 6580 clips of single notes along with playing technique annotations. The IDMT-SMT-Audio-Effects dataset [21] contains ≈ 20 hours of single guitar notes and chords with varying audio effects. Finally, the IDMT-SMT-Guitar dataset [11] contains several types of guitar data, including single notes, playing techniques, note clusters, and note and chord-level annotations for short excerpts. While each of these datasets are useful, none of them provide note-level annotations of realistic polyphonic guitar pieces, which is a limiting factor in exploring many interesting new research directions.

The absence of a sizable dataset for realistic polyphonic guitar playing is largely due to the difficulty of annotating complex guitar recordings directly. In order to help facilitate analysis of guitar recordings, hexaphonic guitar pickups have become a useful research tool. The idea of using hexaphonic pickups to generate transcriptions was first proposed by O’Grady and Rickard in 2009 [16]. In their method, signals from individual strings are analyzed using supervised non-negative matrix factorization. Hexaphonic pickups have also been used for analysis and resynthesis of monophonic single-note guitar recordings [15], as well as for visualizing guitar performances [1].

We posit that, despite piano and guitar having comparable popularity, research has focused much more heavily on analysis of piano recordings simply because of the availability of data. Online communities that provide guitar tablature such as Ultimate Guitar² are very popular, and accurate methods for guitar tablature transcription would have the potential to attract a vibrant community. By creating GuitarSet, and therefore demonstrating an efficient process of creating detailed note level annotations for guitar, we hope to provide the community with better resources for studying guitar transcription and more.

The collection and analysis methods for GuitarSet was designed with the principles described by Su and Yang [22] in mind: (1) **Generality**: We chose well-known progressions in popular styles as the basis of GuitarSet’s material, and collect realistic, complex and polyphonic musical phrases. (2) **Efficiency**: The method of creating annotations for GuitarSet is mostly automated, with human experts focusing on correcting onsets, which requires context and expertise. GuitarSet can be easily extended for this reason. (3) **Cost**: The key equipment, the hexaphonic pickup, is very affordable. and (4) **Quality**: In order to preserve nuances in the performance, including intra-note pitch deviations and inter-string onset-time patterns, we craft special tools and provide multiple annotation formats to ensure high quality annotations.

3. DATA COLLECTION PROCESS

Hexaphonic pickups are magnetic pickups that have individual outputs for each magnet. We ordered a clip-on hexaphonic pickup from ubertar.com, which has 6 individual single coil magnets, and is manually attached to an acoustic guitar. For better pickup signal-to-noise ratio (SNR), nickel wound steel strings are used for the acoustic guitar.

The audio was recorded in a small, soundproof recording studio with minimal reverberation. In addition to the six channels from the hexaphonic pickup, we also record the guitar using a Neumann U87 condenser microphone, placed ≈ 30 cm in front of the 18th fret of the guitar. This results in seven channels of audio overall.

Six experienced guitarists were recruited to record for this database. All six players have more than 10 years of guitar playing experience, and were recruited by the authors. The guitarists were asked to play 30 twelve to sixteen bar excerpts from lead-sheets in a variety of keys, tempos, and musical genres, described in Section 4. During recording, guitarists were provided with a backing track that consisted of a click track, drum set, and bass line, heard through monitoring headphones. For each excerpt, players were asked to comp (play chords), and then to solo over their own comping. The guitarists were allowed to replay excerpts until they were aesthetically satisfied with their performance.

4. DATASET OVERVIEW

We use the JAMS file format [10] to store the rich collection of annotations for this dataset. For each recording, the JAMS file contains annotations for tempo, key, and style (metadata); beats and downbeats (inferred from the click track); instructed chords (from the lead-sheets); performed chords (via automatic estimation); note-level transcription, including string and fret position (via automatic estimation), onsets (via annotation), offsets (via automatic estimation) and pitch contour for each note (via validated automatic estimation). Descriptions of each of these annotation types are detailed in Section 5. Figure 1 gives a visualization of some of the annotations provided for an excerpt of the dataset.

In total, each player provided 30.47 minutes of musical material, resulting in just over 3 hours of content in total. Each player was asked to play 30 excerpts, organized as follows: 3 different chord progressions are paired with each of the 5 different genres, all recorded at two different tempi: slow and fast. The three progressions were the 12 bar blues, Autumn Leaves, and Pachelbel’s Canon. The five different genres were Rock, Jazz, Funk, Bossa Nova (BN), and Singer-Songwriter (SS). In order to broaden the chord gamut in GuitarSet, key signatures were independently assigned to each of the 30 excerpts.

5. ANNOTATION METHODS

The hexaphonic recordings are analyzed to generate annotations for each string individually, and a complete tran-

² <http://www.ultimate-guitar.com/>

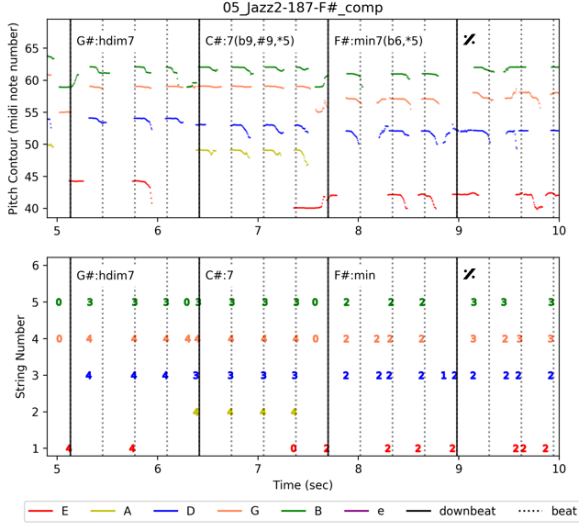


Figure 1. A 5 second excerpt of Jazz comping. Downbeats and beats are indicated with solid and dashed vertical lines respectively. (Top) Played chords and pitch contours, colored by string. (Bottom) Instructed chords (lead sheet) and string/fret positions.

scription of the excerpt is generated by aggregation. For each string, onset/offset time pairs along with continuous pitch tracks are annotated semi-automatically, with manual validation. The validated transcriptions are then used to automatically create derivative annotations, including chords, string and fret number, and more.

We first pre-process the hexaphonic recordings using the KAMIR bleed removal algorithm [17] to reduce noise picked up by the single coil magnets from adjacent strings. We then generate a rough note-level transcription by running pYIN-note [13] over the recording of each string; this rough transcription is used as the starting point for manual validation.

5.1 Note-Level Annotations

We focus our manual annotation efforts on creating note-level annotations, such as the one shown in Figure 1 (Top). Accurately creating or correcting annotations of individual string recordings requires contextual information and musical expertise. For example, an intentionally muted string in a full chord still produces a clear pitch and onset when examining the single muted string in isolation. However, when mixed together with the other more resonant strings, the muted note is completely masked. Because the muted note is neither intended by performer nor heard by listeners, we chose not to annotate it.

In order to address this issue efficiently and maximize automation, we simplify the problem by taking a component approach, and determine the onsets, offsets and pitch tracks sequentially. We first focus on generating high quality onset annotations. By manually validating the onsets, muted notes that shouldn't be included in the annotation

and other non-note events are left out of the annotation. Offsets are then automatically estimated, and the resulting note regions are used to facilitate highly accurate pitch track estimations.

5.1.1 Onsets

Given automatically estimated onsets, removing false positive onsets can efficiently be done manually, but accurately adding missed onsets efficiently requires machine assistance. In order to allow annotators to easily add missed onsets, we automatically adjust human-estimated onset times by searching for the most likely spectral flux peak in a local neighborhood. Concretely, for a human estimated onset time \tilde{a} , the true onset time a is determined by finding the position for which the windowed onset strength function $G_a(t)$ is maximized.

Let $E(t)$ be the root-mean-squared (RMS) energy calculated at time t , and $N_a(t)$ be the spectral flux novelty function at time t [3]:

$$N_a(t) = \sum_{k=1}^{n/2} H(|X(t, k)| - |X(t - l, k)|) \quad (1)$$

where $H(x) = \frac{x+|x|}{2}$ is the half-wave rectification function and $l = 5.8\text{ms}$ is a constant lag in time, n is the number of analysis bins, and k is the bin index.

The windowed onset strength function $G_a(t)$ is constructed as follows,

$$G_a(t) = E(t) * N_a(t) * \mathcal{N}(\tilde{a}, \sigma^2) \quad (2)$$

and the onset time is computed as

$$a = \arg \max_t (G_a(t)), \quad (3)$$

where $t \in [\max(a_{prev} + \tau_a, a - 3\sigma), a + 3\sigma]$, $\tau_a = 50\text{ms}$ and $\sigma = 30\text{ms}$. The lower limit on t ensures there are at least τ_a seconds between consecutive onsets. The Gaussian component in $G_a(t)$ ensures the locality of the onset search, favoring proximity with the human estimate. Figure 2 shows an instance of such an adjustment.

5.1.2 Offsets

For all onsets a , the corresponding offset b is estimated automatically, using the following criteria. First the offset novelty $N_b(t)$ is modified slightly from Equation 1:

$$N_b(t) = - \sum_{k=1}^{n/2} H'(|X(t, k)| - |X(t - l, k)|) \quad (4)$$

where $H'(x) = \frac{x-|x|}{2}$ is the negative half wave rectification function and $l = 5.8\text{ms}$.

Using the generated offset novelty function, an offset strength function $G_b(t)$ is generated.

$$G_b(t) = \frac{N_b(t) * (\log E(t - l) - \log E(t))}{E(t)} \quad (5)$$

where $t \in [a + \tau_b, a_{next}]$ and $\tau_b = 30\text{ms}$. $l = 5.8\text{ms}$ is the hop length in time of the analysis window.

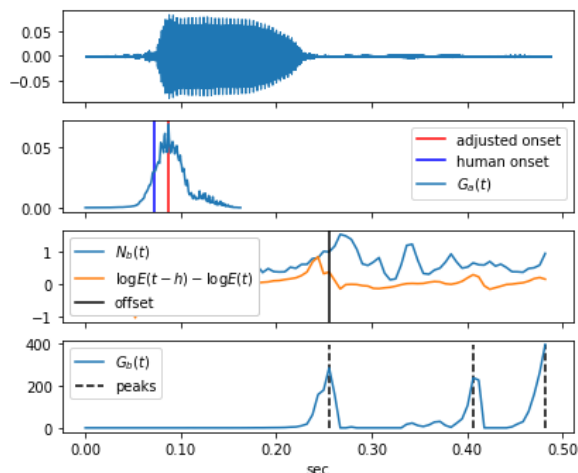


Figure 2. (Top) Waveform of a single note. (Upper Middle) Human estimated onsets, adjusted by examining the onset strength function. (Lower Middle) Offset novelty functions and the detected offset in black. (Bottom) $G_b(t)$ and detected peaks.

The intuition behind the offset strength function $G_b(t)$ is straightforward: log-RMS difference and spectral flux both give peaks for potential offsets, the RMS in the denominator penalizes peaks of $G_b(t)$ that still have significant energy. The peaks of the offset strength function $G_b(t)$ are then thresholded to generate the offset candidates as shown in Figure 2. Offset candidates within 30 ms of the onset a are discarded, and the first offset candidate in time is then chosen from the remaining offsets as b .

5.1.3 Pitch

After onsets and offsets are determined, the pitch tracks of voiced regions are then estimated using pYin. The resulting estimation is then cleaned by the first author in Tony, mostly correcting octave mistakes.

While we annotated the continuous pitch trajectories of each note, the overall center pitches still needs to be inferred. We choose a simple heuristic that averages the pitch track frequencies. For a note with onset at time a and associated pitch track $f(t)$, $t \in [a, b]$; the center pitch of the note p is estimated by taking the average pitch track over a subset of the note region $t \in [a', b']$, where a' and b' are 25% and 50% of the note duration respectively:

$$p = \frac{1}{b' - a'} \sum_{t=a'}^{b'} f(t) \quad (6)$$

We only consider the subset $t \in [a', b']$ to ensure a perceptually relevant average pitch, since the pitch near the onset and offset of a guitar note can sometimes be unstable (e.g. see Figure 1).

5.2 Derivative Annotations

Given note-level annotations, the lead sheets and the click track, we automatically generate a series of derivative annotations.

5.2.1 String and Fret Position

Since the tuning of the guitar is known at the time of data collection, fret positions can be determined simply by finding the difference in semitones between the annotated pitch and the pitch of the open string. A visualization of these annotations is shown in Figure 1 (Bottom).

5.2.2 Chords

Two different types of chord annotations accompany each of the 180 excerpts. The first type of chord annotation is the chord written in the lead sheet that is provided to the guitar players at the time of data collection. However, in order to better fit the given genre, the players often modified the given chords, hereafter called *instructed chords*. Therefore the *performed chords* are not necessarily the same as the instructed chords. Because the backing track contains a bass line that is aligned to the root and the timing of the instructed chords, the instructed and performed chords vary mostly in chord type, not root. The instructed chords have only four types (major, minor, dominant seventh, half-diminished seventh); specific voicings, extensions and alterations could be freely determined by the players without suggestion bias.

We infer the performed chords by combining information from the lead sheet and the annotated notes. In order to make the comparison between the chords as instructed by the lead sheet and the actual performed chords straightforward, the chord segmentation is determined from the lead sheets. A drawback of this approach is that anticipated or lagging chords changes lead to a slight mismatch between the audio signal and the annotations, which may disturb data-driven methods using this data as a training set. However, we argue that such quantization leads to annotations that are more fit for displaying as sheet music and more consistent than human segmentation, which is subjective in this regard³. Furthermore, these cases are expected to be rare because of the aforementioned backing track.

For each chord segment, we first determine if a string is active by verifying whether the total duration of all notes played on that string exceeds 5% of the segment duration. This activity thresholding ensures that notes in adjacent chord segments do not accidentally cause otherwise silent strings to appear active simply because of an offset in chord changes between the lead sheet and recording. Next, the predominant note is determined for all active strings per segment. This is done by taking the MIDI note value with the longest total duration per-string (summed over all note repetitions in the chord segment), resulting in a set of up to six notes per chord segment from which we subsequently derive a chord label.

³ Informal experiments with symbolic chord recognition software resulted in a far worse segmentation.

The root of the chord is also taken from the lead sheet and the inversion naturally arises from the lowest note in the set. Finally, the chord type is determined from the chroma of the set of notes per string through a decision tree that is part of the open-source MusOO library⁴. See Figure 1 for examples of instructed and played chords.

It is notable that our approach of determining the played chord has several biases; namely, the boundary of each chord and the root of each chord is predetermined. More in depth investigation is needed to determine chord boundaries and roots purely automatically, but this is left for future work.

5.2.3 Beats and Downbeats

Since the data is recorded against a click track, the tempo, beats, downbeats and meter of all the excerpts are known. These annotations are generated for each excerpt automatically given this known metadata.

5.3 Inferred Stroke Information

Another pattern that can be recognized from the annotated data is the inter-string onsets. With the help of the onset adjustment step, the annotation captures minute timing differences across onsets on different strings; and by looking at these onset patterns, one can gain a much better understanding of the picking activity that would be otherwise complex to analyze. Figure 3 shows the onsets per string for a short excerpt. Four different strokes can be clearly identified within this 650 ms excerpt. By examining the relative order of strings in each of the strokes, we can clearly observe that the first and last stroke are down-strokes, and the second and third are up-strokes. Evident from Figure 3, the inter-string onsets are only milliseconds apart during fast strokes, and would be very difficult for humans to manually annotate precisely. This nuanced detail would have been lost if the onset adjustment step were not applied.

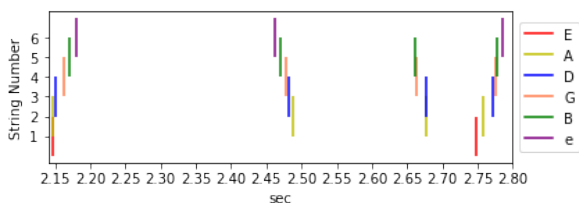


Figure 3. Onsets for each string are shown in different colors.

6. BASELINE EXPERIMENTS

In order to better understand the new challenges posed by this dataset, we evaluate the performance of strong baseline algorithms against our ground truth notes, chords, and

beats/downbeats. These experiments are performed without the algorithms seeing any of GuitarSet’s data. Detailed results can be found in the GuitarSet repository.⁵ All box plots used in this section have box edges showing the first and third quartile, and the whiskers showing 1.5 interquartile range (IQR) away from the box edges.

6.1 Notes

We evaluate the performance of the Deep Saliency multiple- f_0 estimation algorithm [4] on GuitarSet’s polyphonic rhythmic recordings. Figure 4 shows the results across different splits of the data.

Overall, the model has an accuracy of $\approx 46\%$, and the most common type of error is missed, rather than incorrect, notes. Looking at Figure 4 (Top Left), the results are split by genre, and we see that Jazz is overall the most difficult genre to transcribe (likely due to the more complex chord combinations), while Funk has the highest recall and lowest precision (due to short notes and more unvoiced regions). In Figure 4 (Bottom Left), we see that the audio from the pickup is easier to transcribe than the audio from the microphone, likely because the pickup signal is cleaner.

From Figure 4 (Top Right), we see that the performance varies by player, both in terms of average accuracy and in terms of the variance across all the player’s recordings. This suggests that each player’s technique or playing style is different enough that algorithm performance differs significantly. Finally, in Figure 4 (Bottom Right), we see the clear trend that the faster the tempo, the more difficult the excerpt is to transcribe.

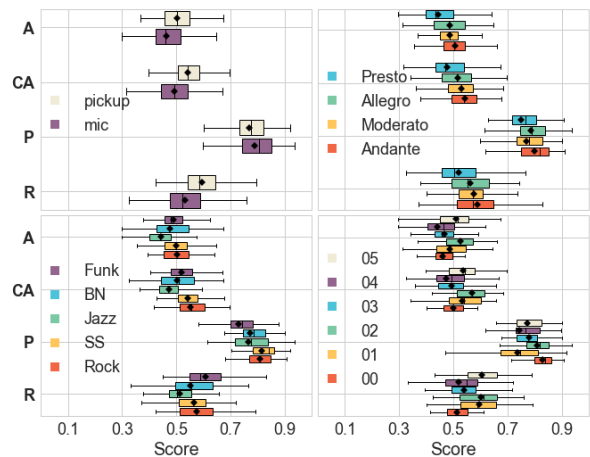


Figure 4. Baseline algorithm multiple- f_0 scores on different splits of GuitarSet. The metrics are A (Accuracy), CA (Chroma Accuracy), P (Precision), and R (Recall). (Top Left) Scores split by recording mode. (Top Right) Scores split by excerpt tempo. (Bottom Left) Scores split by genre. (Bottom Right) Scores split by player.

If only the microphone split of the data is considered, we see that the Deep Saliency model performs worse on

⁴ <https://github.com/jpauwels/libMusOO>

⁵ <https://github.com/marl/GuitarSet>

GuitarSet than it does on Bach10 and MedleyDB [4]. With the accuracy at only $\approx 43\%$, there are still significant possible performance gains to be had.

6.2 Chords

Next, we evaluate the performance of a state-of-the-art chord recognition baseline [14] against the GuitarSet chord labels. The results, stratified by genre, are shown in Figure 5. First, we see that again, some genre's chord labels are easier to estimate than others; in particular, the Rock and Singer Songwriter genres are much easier due to the generally simpler chord types used in those genres compared with the others. Next, we see that there is a large variance in the scores and that there are many outliers. Upon investigating the reason for these outliers, we discovered that some popular guitar textures are not represented in the estimation algorithm's output space. Power chords and octaves, for example, are common guitar textures that are not within the range of typical chord estimation output. While the lead sheet that guides the data collection contains 42 unique chords, the actual detailed chord annotations had a total of 478 unique chord labels (counting all inversions and variations as unique), most of which were small variations of the 42 due to players adding or removing notes.

As shown in Table 1, the overall performance of the baseline chord recognition algorithm on GuitarSet is comparable with the dataset evaluated by Humphrey and Bello [9]. However, as mentioned above, some strata of the dataset are considerably more difficult than the rest.

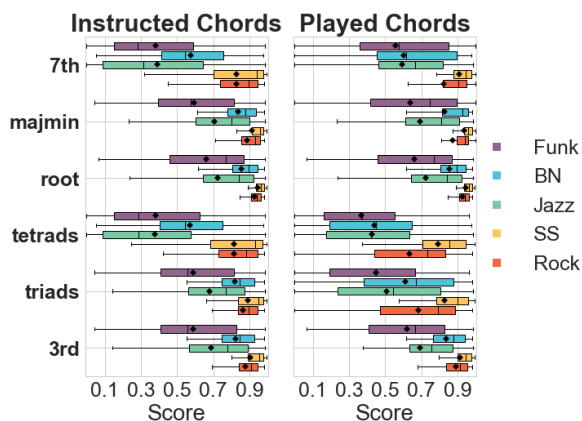


Figure 5. Chord recognition baseline algorithm results on GuitarSet, stratified by genre.

6.3 Beats and Downbeats

The performance of a state-of-the-art beat and downbeat detection algorithm [6] is evaluated on GuitarSet, and the results, stratified by player, are shown in Figure 6. More so than for the previous two tasks, there is a substantial difference between the beat tracker's performance for different players. This suggests that the guitarists have different

Dataset	Root	3rds	Triads	7ths	Tetrads
GuitarSet					
— Instructed	0.903	0.862	0.838	0.669	0.619
— Played	0.903	0.866	0.708	0.810	0.544
H. & B. [9]	0.861	0.836	0.812	0.729	0.671

Table 1. Median weighted recall scores for the baseline algorithm [14] performed on different datasets

characteristics in how they play that affect beat detection, such as their choice of strumming patterns or the strength of their attacks. For example, player 00 has a fast strumming style, and plays chords with embedded melodies, which proves difficult for the algorithm.

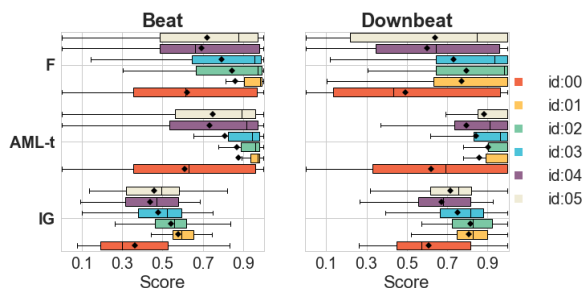


Figure 6. Evaluation of baseline beat/downbeat detection algorithm on GuitarSet, split by player. The metrics are F (F-measure), AML-t (Any Metric Level-Total), and IG (Information gain).

While the median beat and downbeat tracking F-measure is in the 90% range for several players (which is typical for state-of-the-art beat tracking [5]), several substrates of GuitarSet are challenging for beat and downbeat estimation. This is especially true because the tempo and meter do not change over time for each excerpt, yet the data is still challenging for a state-of-the-art beat and downbeat estimation algorithm.

7. CONCLUSIONS

In this paper, we presented a large and carefully annotated dataset of guitar recordings which is available as an open source resource to the research community. We gave a detailed overview of the data collection process and a description of the data itself. Finally, we described our novel process for efficiently and accurately creating note, chord, and beat annotations, and reported the performance of state-of-the-art algorithms on these annotations.

We hope GuitarSet will be useful beyond providing training and evaluation data for transcription models by providing a gateway to investigate interesting problems such as stroke analysis or harmony segmentation. We are pleased to release GuitarSet to the research community and hope that it will foster new, guitar-focused research.

8. ACKNOWLEDGEMENTS

Johan Pauwels has been partly funded by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/L019981/1 and by the European Unions Horizon 2020 research and innovation programme under grant agreement N° 688382.

9. REFERENCES

- [1] Iñigo Angulo, Sergio Giraldo, and Rafael Ramirez. Hexaphonic guitar transcription and visualisation. In *Proceedings of the Second International Conference on Technologies for Music Notation and Representation (TENOR)*, 2016.
- [2] Ana M Barbancho, Isabel Barbancho, Lorenzo J Tardón, and Emilio Molina. *Database of Piano Chords: An Engineering View of Harmony*. Springer, 2013.
- [3] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on speech and audio processing*, 2005.
- [4] R.M. Bittner, B. McFee, J. Salamon, P. Li, and J.P. Bello. Deep salience representations for f_0 estimation in polyphonic music. In *18th International Society for Music Information Retrieval Conference, ISMIR*, 2017.
- [5] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *ISMIR*, 2016.
- [6] S. Durand, J. P. Bello, B. David, and G. Richard. Robust downbeat tracking using an ensemble of convolutional networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 25(1):76–89, 2017.
- [7] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.
- [8] Sebastian Ewert and Mark B Sandler. An augmented lagrangian method for piano transcription using equal loudness thresholding and lstm-based decoding. *arXiv preprint arXiv:1707.00160*, 2017.
- [9] Eric J Humphrey and Juan Pablo Bello. Four timely insights on automatic chord estimation. In *ISMIR*, pages 673–679, 2015.
- [10] Eric J. Humphrey, Justin Salamon, Oriol Nieto, Jon Forsyth, Rachel M. Bittner, and Juan P. Bello. JAMS: A JSON annotated music specification for reproducible MIR research. In *International Society of Music Information Retrieval (ISMIR)*, October 2014.
- [11] Christian Kehling, Jakob Abeßer, Christian Dittmar, and Gerald Schuller. Automatic tablature transcription of electric guitar recordings by estimation of score-and instrument-related parameters. In *DAFx*, pages 219–226, 2014.
- [12] Matthias Mauch, Chris Cannam, Rachel Bittner, George Fazekas, Justin Salamon, Jiajie Dai, Juan Bello, and Simon Dixon. Computer-aided melody note transcription using the tony software: Accuracy and efficiency. In *2015 International Conference on Technologies for Music Notation and Representation*, May 2015.
- [13] Matthias Mauch and Simon Dixon. pyin: A fundamental frequency estimator using probabilistic threshold distributions. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 659–663. IEEE, 2014.
- [14] B. McFee and J.P. Bello. Structured training for large-vocabulary chord recognition. In *18th International Society for Music Information Retrieval Conference, ISMIR*, 2017.
- [15] Raymond Vincent Migneco. *Analysis and synthesis of expressive guitar performance*. PhD dissertation, Drexel University, 2012.
- [16] Paul D. O’Grady and Scott T. Rickard. Automatic hexaphonic and guitar transcription and using and non-negative constraints. In *IET Irish Signals and Systems Conference (ISSC 2009)*. IET, 2009.
- [17] Thomas Prätzlich, Rachel M. Bittner, Antoine Liutkus, and Meinard Muller. Kernel additive modeling for interference reduction in multi-channel music recording. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, May 2015.
- [18] Justin Salamon, Rachel M Bittner, Jordi Bonada, Juan José Bosch Vicente, Emilia Gómez Gutiérrez, and Juan P Bello. An analysis/synthesis framework for automatic f_0 annotation of multitrack datasets. In *18th International Society of Music Information Retrieval (ISMIR) Conference*, October 2017.
- [19] Zhengshan Shi, Kumaran Arul, and Julius O Smith. Modeling and digitizing reproducing piano rolls. In *18th International Society for Music Information Retrieval Conference, ISMIR*, 2017.
- [20] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(5):927–939, 2016.
- [21] Michael Stein, Jakob Abeßer, Christian Dittmar, and Gerald Schuller. Automatic detection of audio effects in guitar and bass recordings. In *Audio Engineering Society Convention 128*. Audio Engineering Society, 2010.

- [22] Li Su and Yi-Hsuan Yang. Escaping from the abyss of manual annotation: New methodology of building polyphonic datasets for automatic music transcription. In *International Symposium on Computer Music Multidisciplinary Research*, pages 309–321. Springer, 2015.
- [23] Li Su, Li-Fan Yu, and Yi-Hsuan Yang. Sparse cepstral, phase codes for guitar playing technique classification. In *ISMIR*, pages 9–14, 2014.

MUSICAL-LINGUISTIC ANNOTATIONS OF *IL LAURO SECCO*

Emilia Parada-Cabaleiro¹ Maximilian Schmitt¹ Anton Batliner¹ Björn W. Schuller^{1,2}

¹ZD.B Chair of Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, Germany

²GLAM – Group on Language, Audio & Music, Imperial College London, UK

emilia.parada-cabaleiro@informatik.uni-augsburg.de

ABSTRACT

The *Italian madrigal*, a polyphonic secular a cappella composition of the 16th century, is characterised by a strong musical-linguistic relationship, which has made it an icon of the ‘Renaissance humanism’. In madrigals, lyrical meaning is mimicked by the music, through the utilisation of a composition technique known as *madrigalism*. The synergy between Renaissance music and poetry makes madrigals of great value to musicologists, linguists, and historians—thus, it is a promising repertoire for computational musicology. However, the application of computational techniques for automatic detection of madrigalisms within scores of such repertoire is limited by the lack of annotations to refer to. In this regard, we present 30 madrigals of the anthology *Il Lauro Secco* encoded in two symbolic formats, MEI and **kern, with hand-encoded annotations of madrigalisms. This work aims to encourage the development of algorithms for madrigalism detection, a composition procedure typical of early music, but still underrepresented in music information retrieval research.

1. INTRODUCTION

The *Italian madrigal* of the 16th century is a secular polyphonic vocal composition characterised by the use of *madrigalisms*, a composition technique that mimics the linguistic content of the lyrics (e.g., emotional concepts such as happiness or sorrow) through the music [14]. This synergy between poetry and music shows the important role that the arts played in the development of the ‘Renaissance humanism’ [29]. Given the intellectual and cultural repercussion of this philosophical movement in Western Europe [13], madrigals evoke high interest for musicological, linguistic, and historical research. Yet, for the comprehension of madrigals, advanced knowledge of the Italian language and poetry, as well as music analysis expertise and knowledge of *mensural notation* [1] are essential. Since music historians, literary scholars, and librarians not always have all these abilities, the development of automatic systems for musical-linguistic synergy detection

within madrigals would assist them in analytical, pedagogical, and cataloguing tasks.

The application of machine learning techniques to early music is restricted by early music being mainly conserved in scanned copies of the original, i.e., no symbolic (machine-readable) information is available. To address this limitation, Optical Music Recognition (OMR) has shown promising results in the automatic generation of symbolic representations of such repertoire [6]. Nevertheless, in the framework of automatic analysis within symbolically encoded scores, for the development of successful systems able to automatically interpret composition procedures, appropriate annotations of such techniques are essential. Despite the large amount of scores from early music repertoire freely available on-line, symbolically encoded or not, labeled early music is still missing. Our work represents an initial contribution to address this lacuna, by presenting the symbolically encoded transcription and annotated representation of 30 madrigals of the *Il Lauro Secco* anthology [21]. A total of 120 scores are presented, 60 in MEI and 60 in **kern—30 of each annotated¹.

With the presented work, we aim at encouraging the development of algorithms for pattern recognition that would pursue identification of musical-linguistic synergies, as e.g., madrigalisms. This will advance automatic analysis techniques, whose practical applications could help researchers from diverse fields (e.g., musicology, linguistics, and history) by assisting them in the evaluation of artistic Renaissance manifestations. The manuscript is laid out as follows: an overview of related work (Section 2); an evaluation of musical-linguistic connections in the Italian madrigal and in the presented repertoire (Sections 3 and 4); a description of the annotation methodology (Section 5); an outline of the annotated repertoire (Section 6); finally, conclusions and future work (Section 7).

2. RELATED WORK

Given the musical, literary, and historical value of the Italian repertoire of the late 16th and early 17th centuries, some initiatives, such as *Tasso in Music Project* [25]² or *The Marenzio Online Digital Edition – MODE*³, spend great effort in making available online symbolic representations of such repertoire. Even though analytical tools are



© Emilia Parada-Cabaleiro, Maximilian Schmitt, Anton Batliner, Björn W. Schuller. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Emilia Parada-Cabaleiro, Maximilian Schmitt, Anton Batliner, Björn W. Schuller. “Musical-linguistic annotations of *Il Lauro Secco*”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ <https://github.com/SEILSdataset/SEILSdataset>

² <http://www.tassomusic.org/>

³ <http://www.marenzio.org/about-mode.html>

The figure shows a musical score for five voices: Canto (C), Alto (A), Quinto (Q), Tenor (T), and Basso (B). The lyrics are: "Nel fo - co d'un bel lau - ro Nel fo - co". The score is divided into two systems. In the first system, the Canto and Quinto parts have a red box highlighting a melisma on the word "foco". The Alto, Tenor, and Basso parts have a blue box highlighting a contrary motion. In the second system, the Alto part has a green box highlighting a contrary motion. The Quinto part has a red box highlighting a melisma on the word "foco".

Figure 1: Example of Contrapunctal madrigalism (CON) in Giovanelli's madrigal. The word *foco* (fire) is mimicked by a contrapunctal texture where the five voices are involved: C (Canto) and Q (Quinto) perform the motive 1 (highlighted in red), in which the word *foco* is displayed by a melisma; A (Alto), T (Tenor), and B (Basso) perform the motive 2 (in blue for T and B), being considered the contrary motion for A (in green).

provided by these initiatives, such as text extraction, word counting, or graphic representation of pitch and rhythm, the symbolically encoded scores, presented in a variety of formats, such as MEI [27]⁴ or **kern [15], do not contain annotations of the musical content. This may limit, e. g., the evaluation of the performance of analytical toolkits, such as Humdrum Toolkit [15]⁵ and music21 [7]⁶, since no ground truth is provided.

Ground truth is essential in the development of algorithms for music information retrieval. Due to this, datasets with annotated information have been developed in order to support a variety of machine learning tasks, as e. g., OMR [22], or harmonic analysis [8]. With the rise of the world-wide web, crowd-sourcing has become a very effective strategy to collect annotations [9]. Indeed, within the framework of digital score libraries, this has been considered for web-based annotation tools [26] as well as to collaboratively perform hand-written transcription [4]. Nevertheless, the annotation of musical content could require a musicological expertise, as e. g., harmonic analysis [8], or the identification of melodic similarities [28] which would make a collaborative annotation system impracticable, thus leading to consider only a limited number of annotators.

3. RHETORIC & MUSIC IN THE ITALIAN MADRIGAL

Rhetoric is the discipline that, through an efficient codification of the discourse (either spoken or written), achieves to convince the audience. Having a consolidated tradition from the times of the ancient Greece [2], in the 16th century, this discipline has been directly applied to music, laying the foundation of *Musica Poetica* [5]. This stylistic movement is founded in a close collaboration between poetry and music, by highlighting the emotional content of the text through the use of musical-rhetoric figures, which will evolve in the 17th century into the *Affektenlehre*, i. e.,

the 'Doctrine of the affections' [17]. As these musical-rhetoric principles are characteristic of the Italian madrigal from the 16th century, such 'word painting' strategies are also known as *madrigalisms* [24]. In madrigalisms, the use of 'chromatism' is progressively introduced, a practice typical of Monteverdi, who at the beginning of the 17th century coined that known as *Seconda pratica* [3]: a new conception of composition in which the music should be governed by the words, thus justifying dissonances and melodic movements that were considered unacceptable till that time, according to Zarlino's harmonic rules [30].

Yet, the madrigal of the 16th century is characterised by madrigalisms which relate to the alternation of musical textures, and not to chromatism, as typical for the madrigal of the 17th century. The madrigal of the 16th century, since based on strong musical-linguistic synergies, differs clearly from other contemporary musical genres such as *frottola*, in which such 'word painting' strategies are not present [14]. Indeed, the artistic value of this madrigal relates also to the high qualification of poets, composers, and interpreters involved in such artistic representation, though to be interpreted in high status social reunions, i. e., in the court [20]. In this regard, the music of the madrigal, in contrast to the *frottola*, shows a more free representation of the text, highlighting its content (usually related to pastoral, sentimental, and erotic themes) through virtuous musical writing [12]. Thus, the essential point of the Italian madrigal of the 16th century is that the composer puts the music into the same artistic level as the poetry [14].

The *Il Lauro Secco* anthology, published for the first time by Angelo Gardano in 1582 at Ferrara (Italy) [18], is a good example of such a repertoire, since both music and lyrics were created by some of the most reputable composers and poets of the time [20]. Furthermore, it was intended to be interpreted in the court of Ferrara, by the *Concerto delle donne* [10], a vocal ensemble of professional singers, which rapidly became an example for other contemporary courts, transforming Italy, for the first time, into the center of music in Europe [14]. Moreover, *Il Lauro Secco* was conceived as a unitary anthology with a common theme where music and poetry of all the madrigals were expressively created for the anthology itself, whose purpose was to be a wedding present for Laura Peverara [11, 19], one of the singers of the *Concerto delle donne*.

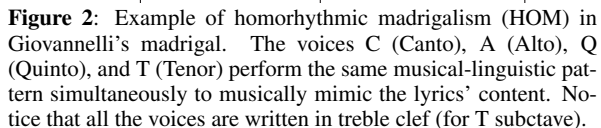
4. MUSICAL-LINGUISTIC SYNERGIES IN *IL LAURO SECCO*

In the madrigals of *Il Lauro Secco* ('The Dry Laurel'), the meaning of the lyrics is expressed mainly through textual 'musical metaphors' and diatonic writing. Thus, the 'word painting' procedures are musically driven by the alternation of diverse musical textures, which we will identify as contrapunctal, homorhythmic, and antiphonal; the melodic development flows through step-wise motion, i. e., the melody is performed in conjunction, so each note is followed by the immediate upper or lower note. For this, rhythmic-melodic 'motifs' are chosen to represent each verse of the lyrics, and are placed into specific musical tex-

⁴ <http://music-encoding.org/>

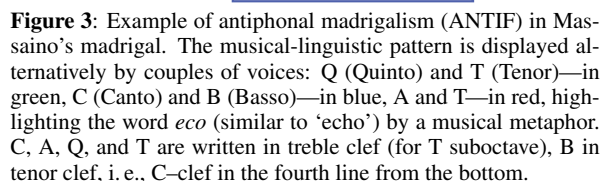
⁵ <http://www.humdrum.org/>

⁶ <http://web.mit.edu/music21/>



4.1 Madrigalisms based on Contrapuntal Texture

In this madrigalism, the word *foco* (fire) is mimicked by music as a dynamic and confused state, as it relates to fire as a physical phenomenon (and its typical instability) as well as a metaphor of *love*. The dynamism and confusion inherent of this concept is enhanced through a contrapuntal texture (most typical composition technique to create movement) as well as through the use of two contrasting motifs. The first of these is characterised by fast rhythm (made up of eighth-notes) and rising ‘melismatic prosody’ (a single syllable of text is sung through several different notes), whereas the second is characterised by a slower rhythm and descending ‘syllabic prosody’ (each syllable



4.2 Madrigalisms based on Homorhythmic Texture

4.3 Madrigalisms based on Antiphonal Texture

In Figure 3, antiphonal texture is used to highlight the similarity between the word *ecco* (interjection used to

Figure 4: Engraved version of the annotation in MEI for the first madrigalism (CON) of Giovanelli's madrigal. Two motifs (CON1 and CON2), one displayed in contrary motion (inv), and a melisma are indicated (cf. Figure 1).

claim attention), and *eco* (acoustic phenomenon for which a sound, through the reflections, is repeatedly perceived, i. e., 'echo'). Here, the 'word painting' procedure is based on the acoustic metaphor generated by the phonetic similarity between the two words. This is a typical example of ANTIF, where each repetition of the musical-linguistic pattern (which consist in two repetitions of the word *ecco* musicalised by a syllabic motif based on a descending third) starts just before the previous has finished and is performed alternatively by different couples of voices.

5. ANNOTATION METHODOLOGY

5.1 Encoding formats

We present 30 madrigals of the *Il Lauro Secco* anthology transcribed in modern notation and encoded in MEI and ****kern** format. For both formats, the annotated and not annotated symbolic scores (cf. subsections 5.2 and 5.3) are included—120 symbolic representations in total, 60 for each format (30 annotated). Both representations have been generated from the Music XML representation of the repertoire given in [21]. The MEI representation has been generated through the on-line Music XML converter *Verovio* [23]⁷, whereas ****kern** files have been produced by using the *xm12hum* compiled program of *Humdrum-extras* toolkit [16]⁸. Conversion errors were manually corrected; given the difficulty to find several annotators with the adequate expertise, the 30 madrigals were annotated by only one expert (one of the authors). Aware of the limitations due to taking into account one single annotator, we will focus on the development of an annotation methodology which adequately describes the considered composition strategies; yet, the presented annotations might be subject to some bias. Notice that both, the original Music XML file and the newly presented symbolic transcriptions in MEI and ****kern**, take

into account the accidentals of the original source, something relevant to consider since in early music, even though some accidentals are not written, they might be considered when performing the repertoire. In this regard, when playing the MEI and ****kern** files, some dissonances should not be considered as 'real' indications of the composer, but just as the result of performing a 'diplomatic', faithful transcription of the source. A transcription which contains cautionary accidentals is included in *finale* and *pdf* formats in [21].

5.2 Annotation in MEI

For the annotation of the madrigalisms in MEI, the function `<harm>` has been considered, which visually engraves the annotations above each staff. For each voice, each single musical-linguistic pattern within a madrigalism has been marked by a starting and ending point, indicated by '*', followed by the name of the madrigalism, i. e., CON, HOM, and ANTIF (cf. Figure 4). Additional composition strategies have also been indicated:

Melisma (mel): When several notes are performed for a syllable of the text (cf. Figure 4 upper staff). Notice that typical embellishments, i. e., ornaments added to a note to 'briefly' decorate it are not considered a *melisma*.

Inversion (inv): When the melodic line of a musical-linguistic pattern is displayed in contrary motion w. r. t. the 'reference', i. e., the first presentation of such musical-linguistic pattern (cf. Figure 4, second staff from the top).

Acephalous (acef): When a musical-linguistic pattern starts without the initial part present in the reference. See, e. g., CON in Marenzio's madrigal at measure 27.

Multiple voices: Double and triple voices, i. e., voices that perform simultaneously the same musical-linguistic pattern, are intrinsic of HOMs and ANTIFs. However, this procedure may also be considered in CONs—when a musical-linguistic pattern is performed simultaneously by more than one voice; yet, it is possible to perceive the contrapunctal texture. Such voices have been indicated as 'CONdouble' or 'CONtriple' (see, e. g., the CON of Gabrieli's madrigal at measure 12). Notice that 'anticipations' and 'retardations' (i. e., when one of the voices, performed simultaneously, starts before or finishes after the others), since typical of madrigalisms, have not been taken into account for the annotation.

Repetition (rep): When a musical-linguistic pattern is repeated in the same voice within a madrigalism, this has been indicated as **rep**. When a whole madrigalism is repeated, this has been indicated as **CONrep**, **HOMrep**, and **ANTIFrep**. Notice that the end of madrigalisms is usually denoted by rests, and their repetition uses to be performed by a different combination of voices. See, e. g., the HOM of Fronti's madrigal at measure 19 (four voices) and its repetition at measure 23 (five voices).

Variation (var): When a musical-linguistic pattern is perceived as similar to the reference, due to rhythmic-melodic aspects still present but with modifications that goes beyond minimal melodic alterations, which would be typical

⁷ <http://www.verovio.org/musicxml.html>

⁸ extras.humdrum.org/man/xm12hum/

**kern	**text	**kern	**text	**kern	**text	**kern	**text	**kern	**text	**cdata—harm
*staff5	*staff5	*staff4	*staff4	*staff3	*staff3	*staff2	*staff2	*staff1	*staff1	*
=1—	=1—	=1—	=1—	=1—	=1—	=1—	=1—	=1—	=1—	=1—
*clefF4	*	*clefGv2	*	*clefGv2	*	*clefG2	*	*clefG2	*	*
*k[]	*	*k[]	*	*k[]	*	*k[]	*	*k[]	*	*
*M4/4	*	*M4/4	*	*M4/4	*	*M4/4	*	*M4/4	*	*
1r	.	2g\	Nel	1r	.	1r	.	4r	.	<CON_5v+2mot
.	4g/	.	.
.	.	4.f\	fo—	8a\L	fo—	.
.	8b\J	.	.
.	8cc\L	.	.
.	.	8f\	co	8dd\J	.	.
=2	=2	=2	=2	=2	=2	=2	=2	=2	=2	=2
1r	.	4e\	d'un	1r	.	2g/	Nel	8ee\L	.	.
.	8ff\J	.	.
.	.	4e\	bel	8gg\L	.	.
.	8ee\J	.	.
.	.	2d\	lau—	.	.	4.a/	fo—	2ff#\	.	.
.	8a/	co	.	.	.

Figure 5: **kern annotation for the CON at the beginning of of Giovanelli's madrigal, in which five voices (5v) and two motifs (2mot) are involved. Notice that the annotation would be visually displayed above the first staff, in the same position as *CON1 (cf. Figure 4).

in order to prevent dissonant collisions. See, e.g., Mas-saino's madrigal at measure 50.

Imitation (imit): When a voice within a madrigalism 'freely' imitates a musical-linguistic pattern, usually by repeating single elements taken from it, such as a rhythm and/or melodic extracts, and by repeating words or by anticipating the next verse. See, e.g., the first madrigalism (ANTIF) of Perue's madrigal at measure 2–5.

Libero (lib): In CON and ANTIF, when all the voices perform the same verse of the lyrics in 'free musical imitation' among them, i.e., since no specific rhythmic-melodic pattern is associated to the textual verse, no musical-linguistic pattern can be identified as reference. In HOM, this indicates that a madrigalism starts and finishes in homorhythm but in its central area, the voices present rhythmic variations that disrupt their perfect vertical alignment; see, e.g., Fronti's madrigal at measure 71.

Different motifs: When a verse of the text is musicalised by different musical motifs within the same madrigalism; this has been identified with a different number, e.g., CON1 and CON2 (cf. Figure 4).

Diminution (dim): When a musical-linguistic pattern is performed in rhythmic diminution, i.e., the rhythm displayed is divided by half w.r.t. the reference. See, e.g., the last madrigalism of Giovanelli's madrial at measure 62.

5.3 Annotation in **kern

For the annotation of madrigalisms in **kern, the **harm spine has been considered, which visually displays the harmonic annotations below the staff, where the lyrics are located in the presented repertoire. In order to avoid collision with the lyrics, and since our intention is not to annotate harmonic content, we have engraved the annotations above the first staff from the top, by using the command 'cdata', i.e., **cdata-harm (cf. Figure 5). For each madrigalism, the starting and ending point has been identified as '<' and '>', respectively. When a madrigalism starts before

the previous has finished, i.e., there is a overlap between both, '<<>>' has been considered. In addition to these, other elements have been indicated:

(i) The number of voices, i.e., for CON and HOM the voices participating (from 1v to 5v); for ANTIF the alternating entries (e.g., four entries—4v). When in CON 'multiple voices' are involved (cf. Section 5.2), these were also indicated (e.g., one doubled voice—1doub).

(ii) The combination among textures: HOM+imit and ANTIF+imit—when the majority of the voices are homorhythmic or antiphonal and one performs imitatively (see, e.g., the first madrigalism of Perue's madrigal); HOM+CON and ANTIF+CON—when the majority of the voices are homorhythmic or antiphonal and one performs the same musical-linguistic pattern in counterpoint.

(iii) The number of motifs considered, when 'different motifs' (cf. Section 5.2) have been used to musicalise a verse of the lyrics (e.g., two motives—2mot).

(iv) The repetitions of a madrigalism are indicated as <CONrep, <HOMrep, and <ANTIFrep (cf. Section 5.2).

6. ANNOTATIONS ASSESSMENT

6.1 Musical Evaluation

In the presented repertoire, we identified a total of 437 madrigalisms across the 30 madrigals (mean of 14.5, and standard deviation (std) 3.7): 199 CON (mean of 6.6, std 2.9); 139 HOM (mean of 4.6, std 3); 59 ANTIF (mean of 1.9, std 1.9); 40 combination between the previous—comb (mean of 1.3, std 1). In Table 1, the distribution of madrigalisms across the 30 madrigals displays the typical alternation between contrapunctal and homorhythmic textures, which is shown by almost all the madrigals presenting both CON and HOM. Even those in which HOM has not been considered, i.e., Correggio's and Strigio's madrigals, present a high number of multiple voices, which decreases the sensation of movement typical of CON; this is

	Alberti	Bardi	Belli	Bertani	Correggio	Da Locca	Eremita	Fiorino	Fronti	Gabrielli	Giovannelli	Ingegneri	Isnardi	Luzzaschi	Macque	Manara	Marenzio	Massaino	Milleville	Mosto	Perue	Pigna	Porta	Spontone	Stabile	Striggio	Vecchi	Virchi	Wert	Zailo
CON	11	8	5	8	8	9	2	3	4	5	8	8	9	14	3	3	8	10	7	10	4	6	3	7	4	11	5	6	5	5
HOM	2	4	8	1	—	7	7	5	15	5	5	2	4	3	3	3	7	5	8	7	9	1	3	6	3	—	3	5	5	3
ANTIF	1	—	1	—	—	4	3	—	—	3	—	1	3	2	1	6	7	4	1	2	—	2	2	2	7	—	2	1	2	2
comb	2	2	—	1	1	1	3	3	—	—	2	2	1	—	3	4	2	—	1	1	1	1	2	2	—	2	1	—	2	—
TOTAL	16	14	14	10	9	21	15	11	19	13	15	13	17	19	10	16	24	19	17	20	14	10	10	17	14	13	11	12	14	10
voice _{mul}	3	6	4	3	5	3	—	1	1	2	—	2	6	5	—	1	6	8	3	6	—	5	3	4	1	5	—	2	1	2
<<>>	2	—	—	1	2	3	—	1	1	1	—	4	1	2	1	2	—	2	1	3	3	1	—	—	1	5	—	—	1	—
rep	—	—	3	—	2	4	3	2	5	1	4	1	1	6	1	2	2	—	6	—	—	—	3	2	1	—	—	3	—	—
mot _{dif}	1	—	—	1	—	—	—	—	—	3	1	2	2	3	1	—	—	1	—	2	—	—	—	—	—	—	2	3	1	—
mel	2	—	5	5	1	—	4	2	—	—	5	—	10	22	13	—	30	2	—	—	—	14	3	3	1	13	3	4	—	—
# ms	77	61	66	75	71	84	69	63	87	65	82	81	99	130	71	71	100	96	81	92	50	75	63	69	72	96	70	79	96	66

Table 1: Occurrence for each madrigal of CON, HOM, ANTIF, and textural combinations (comb). Total number of madrigalisms, overlap between these (<<>>), their repetitions, and length of the madrigals in measures (# ms). The use within madrigalisms of multiple voices (voice_{mul}), different motifs (mot_{dif}), and melisma (mel), is also given.

also observed in madrigals with more CON than HOM (see e. g., Luzzaschi’s madrigal amongst others).

The madrigals with more HOM than CON are rare, and present the opposite tendency, i. e., a low number of multiple voices, as e. g., those from Fronti and Perue. The use of ANTIF, even less typical than the other madrigalisms, is characteristic in the musical writing of Marenzio, Stabile, and Manara. As it would be expected, the use of repetitions is mostly related to longer madrigals, with the exception of the one by Belli that—only 66 measures long—presents three repetitions of a madrigalism. However, this is related to the fact that Belli’s madrigal presents a majority of HOM, which commonly are shorter than CON. This is clear in Perue’s madrigal, i. e., the shortest (50 measures), presenting 14 madrigalisms (9 of them HOM), whose compactness is increased by the use of 3 overlaps between madrigalisms. For general statistics of the dataset, such as total number of notes or accidentals, see [21].

6.2 Linguistic Evaluation: Melismas

One of the most interesting musical-linguistic synergies within madrigals is the use of *melisma* (cf. Section 5.2). By annotating the presented repertoire, we have identified 142 melismas, which usually are displayed within CON. Indeed, apart from Macque’s madrigal, which presents 13 melismas and only 3 CON, all the other madrigals with a high number of melismas are also characterised by presenting a high number of CON. Yet, we should also consider that in Macque’s madrigal, there are 3 combined madrigalisms, which implicitly present contrapuntal texture. Furthermore, the relationship between counterpoint and melismatic writing should not be taken as a rule but only as a tendency, as shown by Mosto’s madrigal, with 10 CON and no melismas. The purpose of a melisma is to highlight a word, thus this rhetoric ‘artifact’ relates most of the times to linguistic concepts that have an important meaning within a madrigal.

The evaluation of the melisma in the presented repertoire makes the unity of the *Il Lauro Secco* anthology evident, whose madrigals have been composed expressively for the creation of the anthology itself. The majority of the

linguistic concepts highlighted through melisma are therefore mostly the same across the whole anthology, and can be clustered into three categories: (i) Nature, i. e., words such as *leaf* or *green*, making often a meaning game with the name of the addressee of the anthology—‘Laura’ and ‘lauro’ (*laurel* in Italian); (ii) Emotion, i. e., words such as *love*, *happiness*, or *rage*; (iii) Elements of nature, i. e., words such as *fire* or *wind*. Out of the 142 melisma, 46 relate to nature and are displayed across 11 madrigals, the most recurrent words being *lauro* (laurel), *verde* (green), *foglie* (leaves), and *rami* (branch), as well as synonyms of those; 42 relate to emotions, displayed across 8 madrigals through recurrent words such as *lieto* (happy), *amore* (love), and *ira* (ire), and synonyms of those; 34 relate to elements, displayed across 9 madrigals through recurrent words such as *venti* (winds), *acqua* (water), and *fuoco* (fire), as well as synonyms and other related words.

7. CONCLUSIONS AND FUTURE WORK

Our study presents symbolically codified scores and annotations, in **kern and MEI format, of 30 madrigals of the anthology *Il Lauro Secco*. The evaluation of the annotations confirms the unity of the presented repertoire, by displaying similarities across the different madrigals, related in a particular way to musical-linguistic synergies, such as the use of *melisma* to highlight specific concepts. The relationships between poetry and music inherent in the presented repertoire, and consistently presented across pieces by different composers, make it promising for the application of machine learning techniques aimed at the detection of similarities among composers. Our future goals include to continue the annotation of the anthology by other experts, in order to offer an appropriate ‘gold standard’ to refer to. We also plan to further evaluate the presented repertoire through available toolkits for automatic music analysis, as e. g., music21. In addition, we will also work on symbolic annotations of similar repertoires, in order to promote the advancement of algorithms for automatic analysis of scores in early music, especially considering the automatic recognition of music-linguistic synergies.

8. ACKNOWLEDGEMENT



This work was supported by the European Union's Seventh Framework and Horizon 2020 Program under grant agreement No. 338164 (ERC StG iHEARu).

9. REFERENCES

- [1] Willi Apel. *The notation of polyphonic music, 900-1600*. The Mediaeval Academy of America, Cambridge, MA, USA, 1961.
- [2] Aristotle and George Alexander Kennedy. *On rhetoric: A theory of civic discourse*. Oxford University Press, New York, NY, USA, 2006.
- [3] Denis Arnold and Nigel Fortune. *The new Monteverdi companion*. Faber & Faber, London, UK, 1985.
- [4] Manuel Burghardt and Sebastian Spanner. Allegro: User-centered design of a tool for the crowdsourced transcription of handwritten music scores. In *Proc. of DATECH*, pages 15–20, Göttingen, Germany, 2017. ACM.
- [5] Joachim Burmeister. *Musical poetics*. Yale University Press, New Haven, CT, USA, 1993.
- [6] Jorge Calvo-Zaragoza, Gabriel Vigliensoni, and Ichiro Fujinaga. One-step detection of background, staff lines, and symbols in medieval music manuscripts with convolutional neural networks. In *Proc. of ISMIR*, pages 724–730, Suzhou, P. R. China, 2017. ISMIR.
- [7] Michael S. Cuthbert and Christopher Ariza. Music21: A toolkit for computer-aided musicology and symbolic music data. In *Proc. of ISMIR*, pages 637–642, Utrecht, Netherlands, 2010. ISMIR.
- [8] Johanna Devaney, Claire Arthur, Nathaniel Condit-Schultz, and Kirsten Nisula. Theme and variation encodings with roman numerals (TAVERN): A new data set for symbolic music analysis. In *Proc. of ISMIR*, pages 728–734, Málaga, Spain, 2015. ISMIR.
- [9] Anhai Doan, Raghu Ramakrishnan, and Alon Y Halevy. Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96, 2011.
- [10] Elio Durante and Anna Martellotti. *Madrigali segreti per le dame di Ferrara: Il manoscritto musicale F. 1358 della biblioteca estense di modena*. Studio per edizioni scelte (SPES), Florence, Italy, 2000.
- [11] Elio Durante and Anna Martellotti. *"Giovinetta peregrina": La vera storia di Laura Peperara e Torquato Tasso*. Leo S. Olschki, Firenze, Italia, 2010.
- [12] Alfred Einstein. *The Italian madrigal*. Princeton University Press, Princeton, NJ, USA, 1971.
- [13] Anthony Goodman and Angus MacKay. *The impact of humanism on Western Europe*. Taylor & Francis, London, UK, 2013.
- [14] Donald J. Grout and Claude V. Palisca. *A history of Western music*. Norton, New York, NY, USA, 2001.
- [15] David Huron. Music information processing using the Humdrum Toolkit: Concepts, examples, and lessons. *Computer Music Journal*, 26(2):11–26, 2002.
- [16] David Brian Huron. *The humdrum toolkit: Reference manual*. Center for Computer Assisted Research in the Humanities, Menlo Park, CA, USA, 1994.
- [17] Athanasius Kircher. *Musurgia universalis sive ars magna consoni et dissoni in X. libros digesta*, volume 2. Georg Olms, Hildesheim, Germany, 1999.
- [18] François Lesure. *Recueils imprimés XVIe-XVIIe siècles*. Henle, Munich, Germany, 1960.
- [19] Anthony Newcomb. The three anthologies for Laura Peperara, 1580–1583. *Rivista Italiana di Musicologia*, 10:329–345, 1975.
- [20] Anthony Newcomb. *The madrigal at Ferrara: 1579-1597*, volume 1. Princeton University Press, Princeton, NJ, USA, 1980.
- [21] Emilia Parada-Cabaleiro, Anton Batliner, Alice E. Baird, and Björn Schuller. The SEILS dataset: Symbolically encoded scores in modern-early notation for computational musicology. In *Proc. of ISMIR*, pages 575–581, Suzhou, P. R. China, 2017. ISMIR.
- [22] Pavel Pecina and Jan Hajič. In search of a dataset for handwritten optical music recognition: Introducing MUSCIMA++. *arXiv preprint arXiv:1703.04824*, 1:1–16, 2017.
- [23] Laurent Pugin, Rodolfo Zitellini, and Perry Roland. Verovio: A library for engraving MEI music notation into SVG. In *Proc. ISMIR*, pages 107–112, Taipei, Taiwan, 2014. ISMIR.
- [24] Don Michael Randel. *The Harvard dictionary of music*. Harvard University Press, Cambridge, MA, USA, 4 edition, 2003.
- [25] Emiliano Ricciardi. The Tasso in music project. *Early Music*, 43(4):667–671, 2015.
- [26] Philippe Rigaux, Lylia Abrouk, Hervé Audéon, Nadine Cullot, Cécile Davy-Rigaux, Zoé Faget, Elisabeth Gaignet, David Gross-Amblard, Alice Tacaille, and Virginie Thion-Goasdoué. The design and implementation of Neuma, a collaborative digital scores library. *International Journal on Digital Libraries*, 12(2-3):73–88, 2012.
- [27] Perry Roland, Andrew Hankinson, and Laurent Pugin. Early music and the music encoding initiative. *Early Music*, 42(4):605–611, 2014.
- [28] Anja Volk, Peter Van Kranenburg, Jörg Garbers, Frans Wiering, Remco C. Veltkamp, and Louis P. Grijp. A manual annotation method for melodic similarity and the study of melody feature sets. In *Proc. of ISMIR*, pages 101–106, Philadelphia, PA, USA, 2008. ISMIR.
- [29] James Anderson Winn. *Unsuspected eloquence: A history of the relations between poetry and music*. Yale University Press, New Haven, CT, USA, 1981.
- [30] Gioseffo Zarlino. *Le istituzioni harmoniche*. Gregg Press, Ridgewood, NJ, USA, 1966.

VOCALSET: A SINGING VOICE DATASET

Julia Wilkins^{1,2}

Prem Seetharaman¹

Alison Wahl^{2,3}

Bryan Pardo¹

¹ Computer Science, Northwestern University, Evanston, IL

² School of Music, Northwestern University, Evanston, IL

³ School of Music, Ithaca College, Ithaca, NY

juliawilkins2018@u.northwestern.edu

ABSTRACT

We present VocalSet, a singing voice dataset of a capella singing. Existing singing voice datasets either do not capture a large range of vocal techniques, have very few singers, or are single-pitch and devoid of musical context. VocalSet captures not only a range of vowels, but also a diverse set of voices on many different vocal techniques, sung in contexts of scales, arpeggios, long tones, and excerpts. VocalSet has recordings of 10.1 hours of 20 professional singers (11 male, 9 female) performing 17 different different vocal techniques. This data will facilitate the development of new machine learning models for singer identification, vocal technique identification, singing generation and other related applications. To illustrate this, we establish baseline results on vocal technique classification and singer identification by training convolutional network classifiers on VocalSet to perform these tasks.

1. INTRODUCTION

VocalSet is a singing voice dataset containing 10.1 hours of recordings of professional singers demonstrating both standard and extended vocal techniques in a variety of musical contexts. Existing singing voice datasets aim to capture a focused subset of singing voice characteristics, and generally consist of fewer than five singers. VocalSet contains recordings from 20 different singers (11 male, 9 female) performing a variety of vocal techniques on 5 vowels. The breakdown of singer demographics is shown in Figure 1 and Figure 3, and the ontology of the dataset is shown in Figure 4. VocalSet improves the state of existing singing voice datasets and singing voice research by capturing not only a range of vowels, but also a diverse set of voices on many different vocal techniques, sung in contexts of scales, arpeggios, long tones, and excerpts.

Recent generative audio models based on machine learning [11, 25] have mostly focused on speech applications, using multi-speaker speech datasets [6, 13]. Generation of musical instruments has also recently been ex-

Gender and Voice Type Distribution

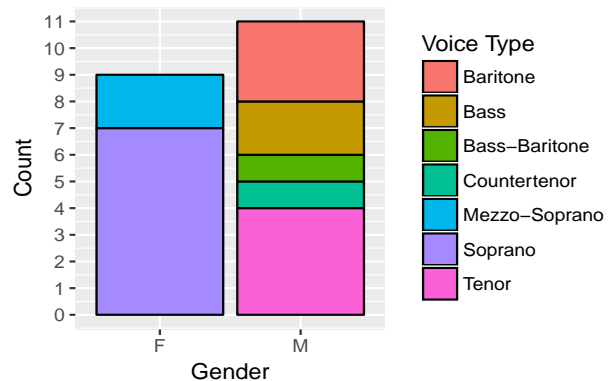


Figure 1. Distribution of singer gender and voice type. VocalSet data comes from 20 professional male and female singers ranging in voice type.

plored [2, 5]. VocalSet can be used in a similar way, but for singing voice generation. Our dataset can also be used to train systems for vocal technique transfer (e.g. transforming a sung tone without vibrato into one with vibrato) and singer style transfer (e.g. transforming a female singing voice to a male singing voice). Deep learning models for multi-speaker source separation have shown great success for speech [7, 21]. They work less well on singing voice. This is likely because they were never trained on a variety of singers and singing techniques. This dataset could be used to train machine learning models to separate mixtures of multiple singing voices. The dataset also contains recordings of the same musical material with different modulation patterns (vibrato, straight, trill, etc), making it useful for training models or testing algorithms that perform unison source separation using modulation pattern as a cue [17, 22]. Other obvious uses for such data are training models to identify singing technique, style [9, 19], or a unique singer’s voice [1, 10, 12, 14].

The structure of this article is as follows: we first compare VocalSet to existing singing voice datasets and cover existing work in singing voice analysis and applications. We then describe the collection and recording process for VocalSet and detail the structure of the dataset. Finally, we illustrate the utility of VocalSet by implementing baseline classification systems for identifying vocal technique and



© Julia Wilkins, Prem Seetharaman, Alison Wahl, Bryan Pardo. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Julia Wilkins, Prem Seetharaman, Alison Wahl, Bryan Pardo. “VocalSet: A Singing Voice Dataset”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

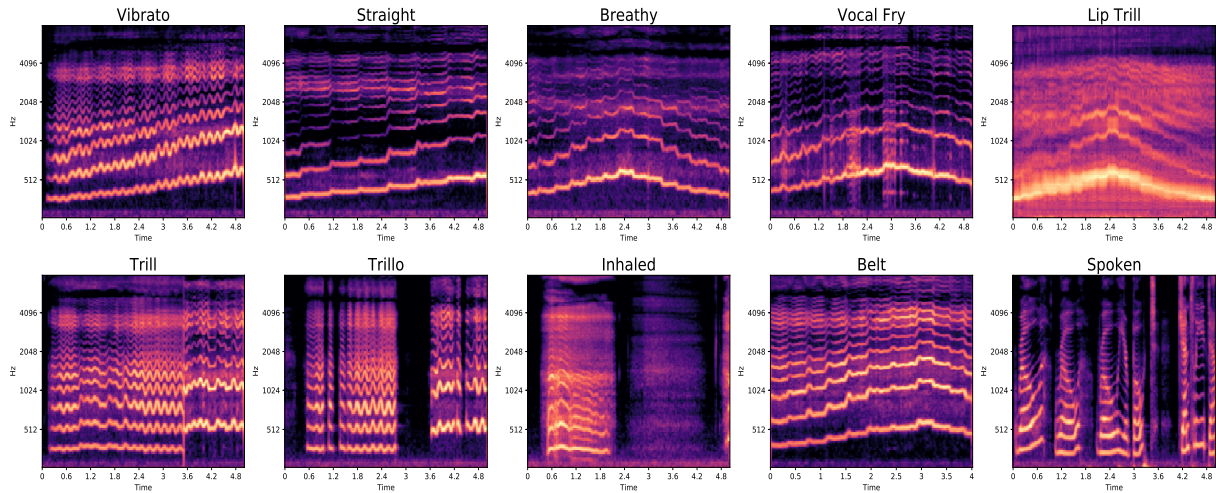


Figure 2. Mel spectrograms of 5-second samples of the 10 techniques used in our vocal technique classification model. All samples are from Female 2, singing scales, except “Trill”, “Trillo”, and “Inhaled” which are found only in the Long Tones section of the dataset, and “Spoken” which is only in the Excerpts section.

singer identification, trained on VocalSet.

2. RELATED WORK

A few singing voice datasets already exist. The Phonation Modes Dataset [18] captures a range of vocal sounds, but limits the included techniques to ‘breathy’, ‘pressed’, ‘flow’, and ‘neutral’. The dataset consists of a large number of sustained, sung vowels on a wide range of pitches from four singers. While this dataset does contain a substantial range of pitches, the pitches are isolated, lacking any musical context (e.g. a scale, or an arpeggio). This makes it difficult to model changes between pitches. VocalSet consists of recordings within musical contexts, allowing for this modeling. The techniques listed above that are observed in the Phonation Modes Dataset are based on the different formations of the throat when singing and not necessarily on musical applications of these techniques. Our dataset focuses on a broader range of techniques in singing, such as vibrato, trill, vocal fry, and inhaled singing. See Table 2 for the full set of techniques in our dataset.

The Vocobox dataset ¹ focuses on single vowel and consonant vocal samples. While they feature a broad range of pitches, they only capture data from one singer. Our data contains 20 singers, with a wide range of voice types and singing styles over a larger range of pitches.

The Singing Voice Dataset [3] contains over 70 vocal recordings of 28 professional, semi-professional, and amateur singers performing predominantly Chinese Opera. This dataset does capture a large range of voices, like VocalSet. However, it does not focus on the distinction between vocal techniques but rather on providing extended excerpts of one genre of music. VocalSet provides a wide

range of vocal techniques that one would not necessarily classify within a single genre so that models trained on VocalSet could generalize well to many different singing voice tasks.

We illustrate the utility of VocalSet by implementing baseline systems trained on VocalSet for identifying vocal technique and singer identification. Prior work on vocal technique identification includes work that explored the salient features of singing voice recordings in order to better understand what distinguishes one person’s singing voice from another as well as differences in sung vowels [4, 12], and work using source separation and F0 estimation to allow a user to edit the vocal technique used in a recorded sample [8].

Automated singer identification has been a topic of interest since at least 2001 [1, 10, 12, 14]. Typical approaches use shallow classifiers and hand-crafted features (e.g. mel cepstral coefficients) [16, 24]. Kako et al. [9] identifies four singing styles music style using the phase plane. Their work was not applied to specific vocal technique classification, likely due to the lack of a suitable dataset. We hypothesize that deep models have not been proposed in this area due to the scarcity of high-quality training data with multiple singers. The VocalSet data addresses these issues. We illustrate this point by training deep models for singer identification and vocal technique classification using this data.

For singing voice generation, [20] synthesized singing voice by replicating distinct and natural acoustic features of sung voice. In this work, we focus on classification tasks rather than generation tasks. However, VocalSet could be applied to generation tasks as well, and we hope our making this dataset available will facilitate that research.

¹ <https://github.com/vocobox/human-voice-dataset>

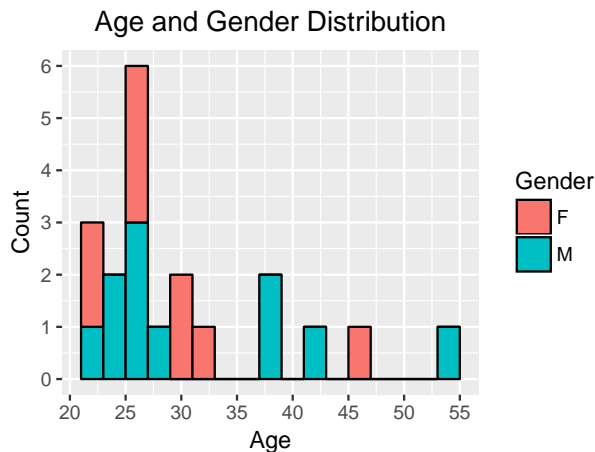


Figure 3. Distribution of singer age and gender. Singer age $\mu = 30.9, \sigma = 8.7$. We observe that the majority of singers lie in the range of 20 to 32, with a few older outlying singers.

3. VOCALSET

3.1 Singer Recruitment

9 female and 11 male professional singers were recruited to participate in the data collection. A professional singer was considered to be someone who has had vocal training leading to a bachelors or graduate degree in vocal performance and also earns a portion of their salary from vocal performance. The singers are of a wide age range and performance specializations. Voice types present in the dataset include soprano, mezzo, countertenor, tenor, baritone, and bass. See Figure 1 for a detailed breakdown of singer gender and voice type and Figure 3 for the distribution of singer age vs. gender. We chose to include a relatively even balance of genders and voice types in the dataset in order to capture a wide variety of timbre and spectral range.

3.2 Recording setup

Participants were recorded in a studio-quality recording booth with an Audio-Technica AT2020 condenser microphone, with a cardioid pickup pattern. Singers were placed close to the microphone in a standing position. Reference pitches were given to singers to ensure pitch accuracy. A metronome was played for the singers immediately prior to recording for techniques that required a specific tempo. Techniques marked 'fast' in Table 2 were targeted at 330 BPM, while techniques marked 'slow' were targeted at 60 BPM. Otherwise, the tempo is varied.

3.3 Dataset Organization

The dataset consists of 3,560 WAV files, totalling 10.1 hours of recorded, edited audio. The audio files vary in length, from less than 1 second (quick arpeggios) to 1 minute. Participants were asked to sing short vocalises of arpeggios, scales, long tones, and excerpts during the

data collection. The arpeggios and scales were sung using 10 different techniques. The long tones were sung on 7 techniques, some of which also appear in arpeggios and scales (see Figure 4). Each singer was also asked to sing *Row, Row, Row Your Boat*, *Caro Mio Ben*, and *Dona Nobis Pacem* each in vibrato and straight tone, as well as an excerpt of their choice. The techniques included range from standard techniques such as 'fast, articulated forte' to difficult extended techniques such as 'inhaled singing'. For arpeggios, scales, and long tones, every vocalise was sung on vowels 'a', 'e', 'i', 'o', and 'u'. A portion of the arpeggios and scales are in both C major and F major (underlined in 4, while the harsher extended techniques and long tones are exclusively in C major. For example, singers were instructed to 'belt' a C major arpeggio on each vowel, totalling to 5 audio clips (one per vowel). This is shown in Figure 4. Table 2 shows the data broken down quantitatively by technique.

The data is sorted in nested folders specifying the singer, type of sample, and vocal technique used. This folder hierarchy is displayed in Figure 4.

Each sample is uniquely labelled based on this nested folder structure that it lies within. For example, Female 2 singing a slow, forte arpeggio in the key of F and on the vowel 'e' is labelled as 'f2.arpeggios.f.slow.forte.e.wav'.

The dataset is publicly available ² and samples from the dataset used in training the classification models are also available on a demo website ³.

4. EXPERIMENTS

As an illustrative example of the utility of this data, we perform two classification tasks using a deep learning model on the VocalSet data. In the first task, we classify vocal techniques from raw time series audio using convolutional neural networks. In the second task, we identify singers from raw audio using a similar architecture. The network architectures are shown in Table 1. Note, architectures are identical except for the final output layer.

4.1 Training data and data preprocessing

We removed silence from the beginning, middle, and end of the recordings and then partitioned them into 3 second, non-overlapping chunks at a sample rate of 44.1k. The chunks were then normalized using their mean and standard deviation so that the network didn't use amplitude as a feature for classification. Additionally, by limiting the chunk to 3 seconds of audio, our models can't use musical context as a cue for learning the vocal technique. These vocal techniques can be deployed in a variety of contexts, so being context-invariant is important for generalization.

For each task, we partitioned the dataset into a training and a test set. For the vocal technique classification, we place all samples from 15 singers in the training set and all samples from the remaining 5 singers in the test set. For the singer identification, we needed to ensure that all

² <https://doi.org/10.5281/zenodo.1203819>

³ <https://interactiveaudiolab.github.io/demos/vocalset>

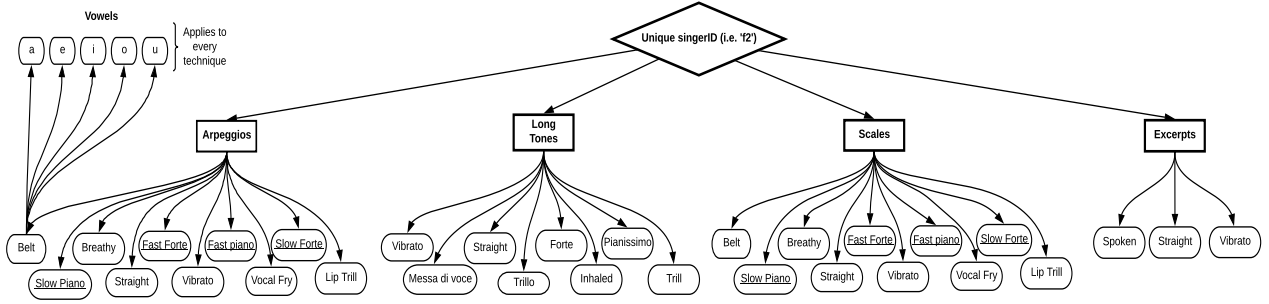


Figure 4. Breakdown of the techniques used in the VocalSet dataset. Each singer performs in four different contexts: arpeggios, long tones, scales, and excerpts. The techniques used in each context are shown. Each technique is sung on 5 vowels, and underlined techniques indicate that the technique was sung in F major and C major.

Layer Name	Input	Conv1	BatchNorm1	MaxPool1	Conv2	BatchNorm2	MaxPool2	Conv3	BatchNorm3	MaxPool3	Dense1	Dense2
# of Units/Filters	3*44100	16	16	-	8	8	-	32	32	-	32	10/20
Filter Size, Stride	-	(1, 128), (1, 1)	-	(1, 64), (1, 8)	(1, 64), (1, 1)	-	(1, 64), (1, 8)	(1, 256), (1, 1)	-	(1, 64), (1, 8)	-	-
Activation function	-	ReLU	-	-	ReLU	-	-	ReLU	-	-	ReLU	softmax

Table 1. Network architecture. The input to the network is 3 seconds of time series audio samples from VocalSet. The output is a 10-way classification for vocal technique classification and a 20-way classification for Singer ID. The architecture for both classifiers is identical except for the output size of the final dense layer. For the dense layers, L2 regularization was set to .001.

singers were present in both the training and the test sets in order to both train and test the model using the full range of singer ID possibilities. We randomly sampled the entire dataset to create training and test sets with a ratio of 0.8 (train): 0.2 (test), while ensuring all singers were both in training and testing data. The recordings were disjoint between the training and test sets, meaning that parts of the same recording were not put in both training and testing data.

Our vocal technique classifier model was trained and tested on the following ten vocal techniques: vibrato, straight tone, belt, breathy, lip trill, spoken, inhaled singing, trill, trillo, and vocal fry (bold in Table 2). Mel spectrograms of each technique are shown in 2, illustrating some of the differences between these vocal techniques.

The remaining categories, such as *fast/articulated forte* and *messa di voce* were not included in training for vocal technique classification. These techniques are heavily dependent on the amplitude of the recorded sample, and the inevitable human variation in the interpretation of dynamic instructions makes these samples highly variable in amplitude. Additionally, singers were not directed to sing a particular *technique* when making amplitude-oriented technique. As a result, singers often paired these amplitude-based techniques with other techniques at the same time, making the categories non-exclusive (e.g. singing fast/articulated forte with a lot of vibrato, or possibly with straight tone). Additionally, messa di voce was excluded because this technique requires singers to slowly crescendo and then decrescendo which, in full, was generally much longer than 3 seconds (the length of training samples).

We train our models with a convolution neural network using RMSProp [23], a learning rate of 1e-4, ReLU activation functions, an L2 regularization of 1e-3, and a dropout

of 0.4 for the second to last dense layer. We use cross entropy as the loss function and a batch size of 64. We train both the singer identification and vocal technique classification models for 200,000 iterations each, where the only difference between the two model architectures is the output size of the final dense layer (10 for vocal technique, 20 for singer ID). Both models were implemented in PyTorch. [15].

4.1.1 Data augmentation

We can also augment our data using standard data augmentation techniques for audio such as pitch shifting. We do this to our training set for vocal technique classification, but not for singer identification. Every excerpt is pitch shifted up and down 0.5 and 0.25 half steps. We report the effect of data augmentation on our models in Table 3. As shown in the table, we did observe some but not a significant accuracy boost when using the augmented model.

4.2 Vocal technique classification

4.2.1 Results

Evaluation metrics for our best 10-way vocal technique classification model are shown in Table 3. We were able to achieve these results using the model architecture in Table 1. This model performs well on unseen test data as we can see from table metrics. When examining sources of confusion for the model, we observed that the model most frequently incorrectly labels test samples as “straight” and “vibrato”. We attribute this in part to the class imbalance in the training data in which there are many more “vibrato” and “straight” samples than other techniques. Additionally, for techniques such as “belt”, many singers exhibited a great deal of vibrato when producing those samples which could place such techniques under the umbrella of

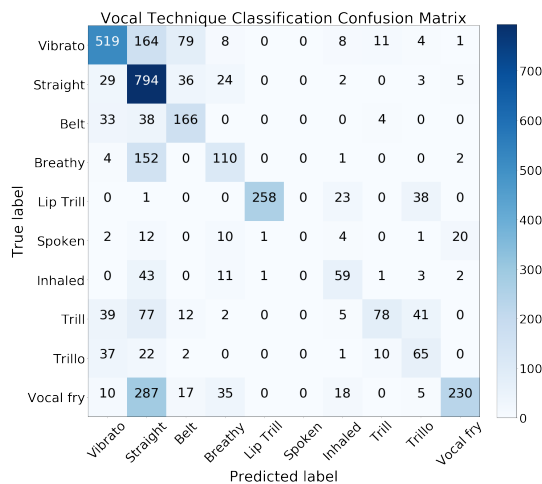


Figure 5. Confusion matrix for the technique classification model showing the quantity of predicted labels vs. true labels for each vocal technique. This model was trained on 10 vocal techniques. A class imbalance can be observed, as the number of vibrato and straight samples is much larger than the remaining techniques. The model performs relatively well for a majority of the techniques, however we see that nearly half of the vocal technique test samples were incorrectly classified as straight tone.

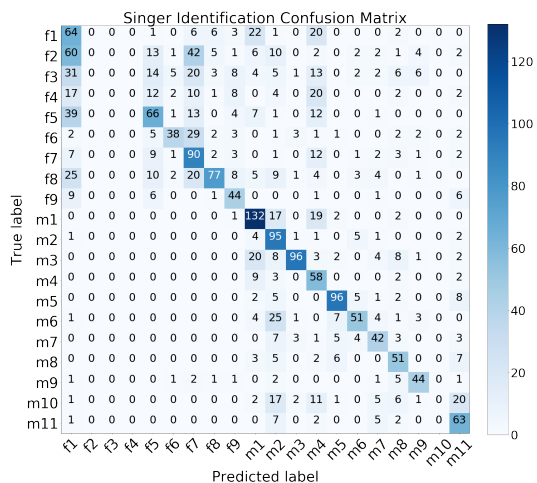


Figure 6. Confusion matrix for the singer identification model displaying the predicted singer identification vs. the true singer identification. We can observe that female voices are much more commonly classified incorrectly versus male voices, likely due to the broader range of male voices present in the training data.

Vocal Techniques	Examples (#)	Time (min.)
Fast/articulated forte	394	22.57
Fast/articulated piano	386	23.03
Slow/legato forte	395	65.28
Slow/legato piano	397	69.75
Lip trill	202	24.40
Vibrato	255	57.79
Breathy	200	28.00
Belt	205	26.24
Vocal fry	198	34.10
Full voice forte	100	16.29
Full voice pianissimo	100	16.58
Trill (upper semitone)	95	18.45
Trillo (goat tone)	100	14.54
Messa di voce	99	23.47
Straight tone	361	71.65
Inhaled singing	100	9.95
Spoken excerpt	20	4.06
Straight tone excerpt	60	24.19
Molto vibrato excerpt	59	24.55
Excerpt of choice	20	20.50

Table 2. The content of VocalSet, totalling to 10.1 hours of audio. Each vocal technique is performed by all 20 singers (11 male, 9 female). Some vocal techniques are performed in more musical contexts (e.g. scales) than others. Bold techniques were used for our classification task.

“vibrato”. We also observed a little bit of expected confusion between “trill” and “vibrato”, as these techniques may have some overlap depending on the singer performing the technique. As seen in Figure 2, the spectrogram representation of these two techniques looks very similar. To address the issue of class imbalance, we tried using data augmentation with pitch shifting to both balance the classes and create more data, but as previously stated and shown in Table 3, there was little improvement over the original model when using training data augmentation.

4.3 Singer identification (ID)

4.3.1 Results

Evaluation metrics for our best 20-way singer identification model are shown in Table 3. The model architecture is identical to that of the vocal technique classification model (see 1), with the exception of the number of output nodes in the final dense layer (20 in the singer identification model vs. 10 in the technique model). The singer identification model did not perform as well as the vocal technique classification model. As shown in Table 3, classifying male voices correctly was much easier for the model than classifying female voices. This is expected due to the high similarity between the female voices in the training data. Figure 1 shows that the female data only contains 2 voice types, while the male data contains 5 voice types.

Because voice type is largely dependent on the vocal range of the singer, having 5 different voice types within the male singers makes it much easier to distinguish be-

Classification Task	Prior	Precision	Recall	Top-2 Accuracy	Top-3 Accuracy	Male Accuracy	Female Accuracy
Vocal Technique	0.242	0.676	0.619	0.801	0.867	-	-
Vocal Technique (trained on augmented data)	0.242	0.677	0.628	0.815	0.891	-	-
Singer ID	-	0.473	0.516	0.638	0.700	0.684	0.351

Table 3. Evaluation metrics for our vocal technique and Singer ID classification models performing on unseen test data. “Prior” indicates the accuracy if we were to simply choose the most popular class (“straight”) to predict test data. We observe a very slight increase in accuracy in the augmented vocal technique model. Our singer ID model has lower performance, likely due to the similarity between different, primarily female, singers.

tween male singers than female singers. The accuracy (recall) for classifying unseen male singers was nearly twice as good as that of unseen female singers.

5. FUTURE WORK

In the future, we plan to experiment with more network architectures and training techniques (e.g. Siamese training) to improve the performance of our classifiers. We also expect researchers to use the VocalSet dataset to train a vocal style transformation model that can transform a voice recording into one using one of the techniques that we have recorded in VocalSet. For example, an untrained singer could sing a simple melody on a straight tone, and our system could remodel their voice using the vibrato or articulation of a professional singer. We envision this as a tool for both musicians and non-musicians alike, and hope to create a web application or even a physical sound installation that users could transform their voices in. We would also like to use VocalSet to train autoregressive models (e.g. Wavenet [25]) that can generate singing voice of specific techniques.

6. CONCLUSION

VocalSet is a large dataset of high-quality audio recordings of 20 professional singers demonstrating a variety of vocal techniques on different vowels. Existing singing voice datasets either do not capture a large range of vocal techniques, have very few singers, or are single-pitch and lacking musical context. VocalSet was collected to fill this gap. We have shown illustrative examples of how VocalSet can be used to develop systems for diverse tasks. The VocalSet data will facilitate the development of a number of applications, including vocal technique identification, vocal style transformation, pitch detection, and vowel identification. VocalSet is available for download at <https://doi.org/10.5281/zenodo.1203819>.

7. ACKNOWLEDGMENTS

This work was supported by NSF Award #1420971 and by a Northwestern University Center for Interdisciplinary Research in the Arts grant.

8. REFERENCES

- [1] Mark A Bartsch and Gregory H Wakefield. Singing voice identification using spectral envelope estimation. *IEEE Transactions on speech and audio processing*, 12(2):100–109, 2004.
- [2] Merlijn Blaauw and Jordi Bonada. A neural parametric singing synthesizer modeling timbre and expression from natural songs. *Applied Sciences*, 7(12):1313, 2017.
- [3] Dawn A. Black, Ma Li, and Mi Tian. Automatic identification of emotional cues in chinese opera singing. 2014.
- [4] Thomas F. Cleveland. Acoustic properties of voice timbre types and their influence on voice classification. *The Journal of the Acoustical Society of America*, 61(6):1622–1629, 1977.
- [5] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders. *arXiv preprint arXiv:1704.01279*, 2017.
- [6] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continuous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.
- [7] John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 31–35. IEEE, 2016.
- [8] Yukara Ikemiya, Katsutoshi Itoyama, and Kazuyoshi Yoshii. Singing voice separation and vocal f0 estimation based on mutual combination of robust principal component analysis and subharmonic summation. 24(11), Nov. 2016.
- [9] Tatsuya Kako, Yasunori Ohishi, Hirokazu Kameoka, Kunio Kashino, and Kazuya Takeda. Automatic identification for singing style based on sung melodic contour characterized in phase plane. In *ISMIR*, pages 393–398. Citeseer, 2009.
- [10] Youngmoo E Kim and Brian Whitman. Singer identification in popular music recordings using voice coding features. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, volume 13, page 17, 2002.

- [11] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.
- [12] Maureen et al. Mellody. Modal distribution analysis, synthesis, and perception of a soprano’s sung vowels. pages 469–482, 2001.
- [13] Gautham J Mysore. Can we automatically transform speech recorded on common consumer devices in real-world environments into professional production quality speech? a dataset, insights, and challenges. *IEEE Signal Processing Letters*, 22(8):1006–1010, 2015.
- [14] Tin Lay Nwe and Haizhou Li. Exploring vibrato-motivated acoustic features for singer identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(2):519–530, 2007.
- [15] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.
- [16] Hemant A Patil, Purushotam G Radadia, and TK Basu. Combining evidences from mel cepstral features and cepstral mean subtracted features for singer identification. In *Asian Language Processing (IALP), 2012 International Conference on*, pages 145–148. IEEE, 2012.
- [17] Fatemeh Pishdadian, Bryan Pardo, and Antoine Liutkus. A multi-resolution approach to common fate-based audio separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 566–570. IEEE, 2017.
- [18] Polina Prooutskova, Christopher Rhodes, and Tim Crawford. Breathy, resonant, pressed - automatic detection of phonation mode from audio recordings of singing. 2013.
- [19] Keijiro Saino, Makoto Tachibana, and Hideki Kenmochi. A singing style modeling system for singing voice synthesizers. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.
- [20] T. Saitou, M. Goto, M. Unoki, and M. Akagi. Speech-to-singing synthesis: Converting speaking voices to singing voices by controlling acoustic features unique to singing voices. pages 215–218, Oct 2007.
- [21] Paris Smaragdis, Gautham Mysore, and Nasser Mohammadhiha. Dynamic non-negative models for audio source separation. In *Audio Source Separation*, pages 49–71. Springer, 2018.
- [22] Fabian-Robert Stöter, Antoine Liutkus, Roland Badeau, Bernd Edler, and Paul Magron. Common fate model for unison source separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 126–130. IEEE, 2016.
- [23] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.
- [24] Tsung-Han Tsai, Yu-Siang Huang, Pei-Yun Liu, and De-Ming Chen. Content-based singer classification on compressed domain audio data. *Multimedia Tools and Applications*, 74(4):1489–1509, 2015.
- [25] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

THE NES MUSIC DATABASE: A MULTI-INSTRUMENTAL DATASET WITH EXPRESSIVE PERFORMANCE ATTRIBUTES

Chris Donahue

UC San Diego

cdonahue@ucsd.edu

Huanru Henry Mao

UC San Diego

hhmao@ucsd.edu

Julian McAuley

UC San Diego

jmcauley@ucsd.edu

ABSTRACT

Existing research on music generation focuses on composition, but often ignores the expressive performance characteristics required for plausible renditions of resultant pieces. In this paper, we introduce the Nintendo Entertainment System Music Database (NES-MDB), a large corpus allowing for separate examination of the tasks of composition and performance. NES-MDB contains thousands of multi-instrumental songs composed for playback by the compositionally-constrained NES audio synthesizer. For each song, the dataset contains a musical score for four instrument voices as well as expressive attributes for the dynamics and timbre of each voice. Unlike datasets comprised of General MIDI files, NES-MDB includes all of the information needed to render *exact* acoustic performances of the original compositions. Alongside the dataset, we provide a tool that renders generated compositions as NES-style audio by emulating the device’s audio processor. Additionally, we establish baselines for the tasks of composition, which consists of learning the semantics of composing for the NES synthesizer, and performance, which involves finding a mapping between a composition and realistic expressive attributes.

1. INTRODUCTION

The problem of automating music composition is a challenging pursuit with the potential for substantial cultural impact. While early systems were hand-crafted by musicians to encode musical rules and structure [25], recent attempts view composition as a statistical modeling problem using machine learning [3]. A major challenge to casting this problem in terms of modern machine learning methods is building representative datasets for training. So far, most datasets only contain information necessary to model the semantics of music composition, and lack details about how to translate these pieces into nuanced performances. As a result, demonstrations of machine learning systems trained on these datasets sound rigid and deadpan. The datasets that do contain expressive performance character-

istics predominantly focus on solo piano [10, 27, 32] rather than multi-instrumental music.

A promising source of multi-instrumental music that contains both compositional and expressive characteristics is music from early videogames. There are nearly 1400¹ unique games licensed for the Nintendo Entertainment System (NES), all of which include a musical soundtrack. The technical constraints of the system’s audio processing unit (APU) impose a maximum of four simultaneous monophonic instruments. The machine code for the games preserves the *exact* expressive characteristics needed to perform each piece of music as intended by the composer. All of the music was composed in a limited time period and, as a result, is more stylistically cohesive than other large datasets of multi-instrumental music. Moreover, NES music is celebrated by enthusiasts who continue to listen to and compose music for the system [6], appreciating the creativity that arises from resource limitations.

In this work, we introduce NES-MDB, and formalize two primary tasks for which the dataset serves as a large test bed. The first task consists of learning the semantics of composition on a *separated score*, where individual instrument voices are explicitly represented. This is in contrast to the common *blended score* approach for modeling polyphonic music, which examines reductions of full scores. The second task consists of mapping compositions onto sets of expressive performance characteristics. Combining strategies for separated composition and expressive performance yields an effective pipeline for generating NES music *de novo*. We establish baseline results and reproducible evaluation methodology for both tasks. A further contribution of this work is a library that converts between NES machine code (allowing for realistic playback) and representations suitable for machine learning.²

2. BACKGROUND AND TASK DESCRIPTIONS

Statistical modeling of music seeks to learn the distribution $P(\text{music})$ from human compositions $c \sim P(\text{music})$ in a dataset \mathcal{M} . If this distribution could be estimated accurately, a new piece could be composed simply by sampling. Since the space of potential compositions is exponentially large, to make sampling tractable, one usually assumes a factorized distribution. For *monophonic* sequences, which consist of no more than one note at a time, the probability



© Chris Donahue, Huanru Henry Mao, Julian McAuley. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Chris Donahue, Huanru Henry Mao, Julian McAuley. “The NES Music Database: A multi-instrumental dataset with expressive performance attributes”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ Including games released only on the Japanese version of the console

² <https://github.com/chrisdonahue/nesmdb>

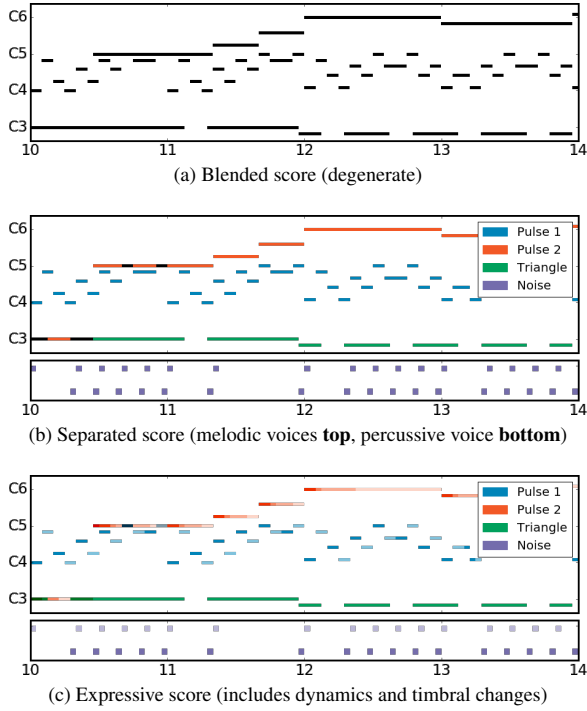


Figure 1: Three representations (rendered as piano rolls) for a segment of *Ending Theme* from *Abadox* (1989) by composer Kiyohiro Sada. The blended score (Fig. 1a), used in prior polyphonic composition research, is degenerate when multiple voices play the same note.

of a sequence c (length T) might be factorized as

$$P(c) = P(n_1) \cdot P(n_2 | n_1) \cdot \dots \cdot P(n_T | n_{t < T}). \quad (1)$$

2.1 Blended composition

While Eq. 1 may be appropriate for modeling compositions for monophonic instruments, in this work we are interested in the problem of multi-instrumental *polyphonic* composition, where multiple monophonic instrument *voices* may be sounding simultaneously. Much of the prior research on this topic [2, 5, 17] represents music in a blended score representation. A blended score B is a sparse binary matrix of size $N \times T$, where N is the number of possible note values, and $B[n, t] = 1$ if any voice is playing note n at timestep t or 0 otherwise (Fig. 1a). Often, N is constrained to the 88 keys on a piano keyboard, and T is determined by some subdivision of the meter, such as sixteenth notes. When polyphonic composition c is represented by B , statistical models often factorize the distribution as a sequence of *chords*, the columns B_t :

$$P(c) = P(B_1) \cdot P(B_2 | B_1) \cdot \dots \cdot P(B_T | B_{t < T}). \quad (2)$$

This representation simplifies the probabilistic framework of the task, but it is problematic for music with multiple instruments (such as the music in NES-MDB). Resultant systems must provide an additional mechanism for assigning notes of a blended score to instrument voices, or otherwise render the music on polyphonic instruments such as the piano.

2.2 Separated composition

Given the shortcomings of the blended score, we might prefer models which operate on a separated score representation (Fig. 1b). A separated score S is a matrix of size $V \times T$, where V is the number of instrument voices, and $S[v, t] = n$, the note n played by voice v at timestep t . In other words, the format encodes a monophonic sequence for each instrument voice. Statistical approaches to this representation can explicitly model the relationships between various instrument voices by

$$P(c) = \prod_{t=1}^T \prod_{v=1}^V P(S_{v,t} | S_{v,\hat{t} \neq t}, S_{\hat{v} \neq v, \forall \hat{t}}). \quad (3)$$

This formulation explicitly models the dependencies between $S_{v,t}$, voice v at time t , and every other note in the score. For this reason, Eq. 3 more closely resembles the process by which human composers write multi-instrumental music, incorporating temporal and contrapuntal information. Another benefit is that resultant models can be used to harmonize with existing musical material, adding voices conditioned on existing ones. However, any non-trivial amount of temporal context introduces high-dimensional interdependencies, meaning that such a formulation would be challenging to sample from. As a consequence, solutions are often restricted to only take past temporal context into account, allowing for simple and efficient ancestral sampling (though Gibbs sampling can also be used to sample from Eq. 3 [13, 16]).

Most existing datasets of multi-instrumental music have uninhibited polyphony, causing a separated score representation to be inappropriate. However, the hardware constraints of the NES APU impose a strict limit on the number of voices, making the format ideal for NES-MDB.

2.3 Expressive performance

Given a piece of a music, a skilled performer will embellish the piece with *expressive characteristics*, altering the timing and dynamics to deliver a compelling rendition. While a few instruments have been augmented to capture this type of information symbolically (e.g. a Disklavier), it is rarely available for examination in datasets of multi-instrumental music. Because NES music is comprised of instructions that recreate an exact rendition of each piece, expressive characteristics controlling the velocity and timbre of each voice are available in NES-MDB (details in Section 3.1). Thus, each piece can be represented as an *expressive score* (Fig. 1c), the union of its separated score and expressive characteristics.

We consider the task of mapping a composition c onto expressive characteristics e . Hence, we would like to model $P(e | c)$, and the probability of a piece of music $P(m)$ can be expressed as $P(e | c) \cdot P(c)$, where $P(c)$ is from Eq. 3. This allows for a convenient pipeline for music generation where a piece of music is first composed with binary amplitudes and then mapped to realistic dynamics, as if interpreted by a performer.

# Games	397
# Composers	296
# Songs	5,278
# Songs w/ length > 10s	3,513
# Notes	2,325,636
Dataset length	46.1 hours
$P(\text{Pulse 1 On})$	0.861
$P(\text{Pulse 2 On})$	0.838
$P(\text{Triangle On})$	0.701
$P(\text{Noise On})$	0.390
Average polyphony	2.789

Table 1: Basic dataset information for NES-MDB.

2.4 Task summary

In summary, we propose three tasks for which NES-MDB serves as a large test bed. A pairing of two models that address the second and third tasks can be used to generate novel NES music.

1. The *blended composition* task (Eq. 2) models the semantics of blended scores (Fig. 1a). This task is more useful for benchmarking new algorithms than for NES composition.
2. The *separated composition* task consists of modeling the semantics of separated scores (Fig. 1b) using the factorization from Eq. 3.
3. The *expressive performance* task seeks to map separated scores to expressive characteristics needed to generate an expressive score (Fig. 1c).

3. DATASET DESCRIPTION

The NES APU consists of five monophonic instruments: two pulse wave generators (P1/P2), a triangle wave generator (TR), a noise generator (NO), and a sampler which allows for playback of audio waveforms stored in memory. Because the sampler may be used to play melodic or percussive sounds, its usage is compositionally ambiguous and we exclude it from our dataset.

In raw form, music for NES games exists as machine code living in the read-only memory of cartridges, entangled with the rest of the game logic. An effective method for extracting a musical transcript is to emulate the game and log the timing and values of writes to the APU registers. The video game music (VGM) format³ was designed for precisely this purpose, and consists of an ordered list of writes to APU registers with 44.1 kHz timing resolution. An online repository⁴ contains over 400 NES games logged in this format. After removing duplicates, we split these games into distinct training, validation and test subsets with an 8:1:1 ratio, ensuring that no composer appears in two of the subsets. Basic statistics of the dataset appear in Table 1.

³ http://vgmrips.net/wiki/VGM_Specification

⁴ <http://vgmrips.net/packs/chip/nnes-apu>

3.1 Extracting expressive scores

Given the VGM files, we emulate the functionality of the APU to yield an expressive score (Fig. 1c) at a temporal discretization of 44.1 kHz. This rate is unnecessarily high for symbolic music, so we subsequently downsample the scores.⁵ Because the music has no explicit tempo markings, we accommodate a variety of implicit tempos by choosing a permissive downsampling rate of 24 Hz. By removing dynamics, timbre, and voicing at each timestep, we derive separated score (Fig. 1b) and blended score (Fig. 1a) versions of the dataset.

Instrument	Note	Velocity	Timbre
Pulse 1 (P1)	{0, 32, ..., 108}	[0, 15]	[0, 3]
Pulse 2 (P2)	{0, 32, ..., 108}	[0, 15]	[0, 3]
Triangle (TR)	{0, 21, ..., 108}		
Noise (NO)	{0, 1, ..., 16}	[0, 15]	[0, 1]

Table 2: Dimensionality for each timestep of the expressive score representation (Fig. 1c) in NES-MDB.

In Table 2, we show the dimensionality of the instrument states at each timestep of an expressive score in NES-MDB. We constrain the frequency ranges of the *melodic* voices (pulse and triangle generators) to the MIDI notes on an 88-key piano keyboard (21 through 108 inclusive, though the pulse generators cannot produce pitches below MIDI note 32). The *percussive* noise voice has 16 possible “notes” (these do not correspond to MIDI note numbers) where higher values have more high-frequency noise. For all instruments, a note value of 0 indicates that the instrument is not sounding (and the corresponding velocity will be 0). When sounding, the pulse and noise generators have 15 non-linear velocity values, while the triangle generator has no velocity control beyond on or off.

Additionally, the pulse wave generators have 4 possible duty cycles (affecting timbre), and the noise generator has a rarely-used mode where it instead produces metallic tones. Unlike for velocity, a timbre value of 0 corresponds to an actual timbre setting and does not indicate that an instrument is muted. In total, the pulse, triangle and noise generators have state spaces of sizes 4621, 89, and 481 respectively—around 40 bits of information per timestep for the full ensemble.

4. EXPERIMENTS AND DISCUSSION

Below, we describe our evaluation criteria for experiments in separated composition and expressive performance. We present these results only as statistical baselines for comparison; results do not necessarily reflect a model’s ability to generate compelling musical examples.

Negative log-likelihood and Accuracy Negative log-likelihood (NLL) is the (log of the) likelihood that a model assigns to unseen real data (as per Eq. 3). A low NLL averaged across unseen data may indicate that a model captures

⁵ We also release NES-MDB in MIDI format with no downsampling

semantics of the data distribution. Accuracy is defined as the proportion of timesteps where a model’s prediction is equal to the actual composition. We report both measures for each voice, as well as aggregations across all voices by summing (for NLL) and averaging (for accuracy).

Points of Interest (POI). Unlike other datasets of symbolic music, NES-MDB is temporally-discretized at a high, fixed rate (24 Hz), rather than at a variable rate depending on the tempo of the music. As a consequence, any given voice has around an 83% chance of playing the same note as that voice at the previous timestep. Accordingly, our primary evaluation criteria focuses on musically-salient *points of interest* (POIs), timesteps at which a voice deviates from the previous timestep (the beginning or end of a note). This evaluation criterion is mostly invariant to the rate of temporal discretization.

4.1 Separated composition experiments

For separated composition, we evaluate the performance of several baselines and compare them to a cutting edge method. Our simplest baselines are unigram and additive-smoothed bigram distributions for each instrument. The predictions of such models are trivial; the unigram model always predicts “no note” and the bigram model always predicts “last note”. The respective accuracy of these models, 37% and 83%, reflect the proportion of the timesteps that are silent (unigram) or identical to the last timestep (bigram). However, if we evaluate these models only at POIs, their performance is substantially worse (4% and 0%).

We also measure performance of recurrent neural networks (RNNs) at modeling the voices independently. We train a separate RNN (either a basic RNN cell or an LSTM cell [15]) on each voice to form our RNN Soloists and LSTM Soloists baselines. We compare these to LSTM Quartet, a model consisting of a single LSTM that processes all four voices and outputs an independent softmax over each note category, giving the model full context of the composition in progress. All RNNs have 2 layers and 256 units, except for soloists which have 64 units each, and we train them with 512 steps of unrolling for backpropagation through time. We train all models to minimize NLL using the Adam optimizer [19] and employ early stopping based on the NLL of the validation set.

While the DeepBach model [13] was designed for modeling the chorales of J.S. Bach, the four-voice structure of those chorales is shared by NES-MDB, making the model appropriate for evaluation in our setting. DeepBach embeds each timestep of the four-voice score and then processes these embeddings with a bidirectional LSTM to aggregate past and future musical context. For each voice, the activations of the bidirectional LSTM are concatenated with an embedding of all of the other voices, providing the model with a mechanism to alter its predictions for any voice in context of the others at that timestep. Finally, these merged representations are concatenated to an independent softmax for each of the four voices. Results for DeepBach and our baselines appear in Table 3.

As expected, the performance of all models at POIs is

worse than the global performance. DeepBach achieves substantially better performance at POIs than the other models, likely due to its bidirectional processing which allows the model to “peek” at future notes. The LSTM Quartet model is attractive because, unlike DeepBach, it permits efficient ancestral sampling. However, we observe qualitatively that samples from this model are musically unsatisfying. While the performance of the soloists is worse than the models which examine all voices, the superior performance of the LSTM Soloists to the RNN Soloists suggests that LSTMs may be beneficial in this context.

We also experimented with artificially emphasizing POIs during training, however we found that resultant models produced unrealistically sporadic music. Based on this observation, we recommend that researchers who study NES-MDB always train models with unbiased emphasis, in order to effectively capture the semantics of the particular temporal discretization.

4.2 Expressive performance experiments

The expressive performance task consists of learning a mapping from a separated score to suitable expressive characteristics. Each timestep of a separated score in NES-MDB has note information (random variable N) for the four instrument voices. An expressive score additionally has velocity (V) and timbre (T) information for P1, P2, and NO but not TR. We can express the distribution of performance characteristics given the composition as $P(V, T | N)$. Some of our proposed solutions factorize this further into a conditional autoregressive formulation $\prod_{t=1}^T P(V_t, T_t | N, V_{t<t}, T_{t<t})$, where the model has explicit knowledge of its decisions for velocity and timbre at earlier timesteps.

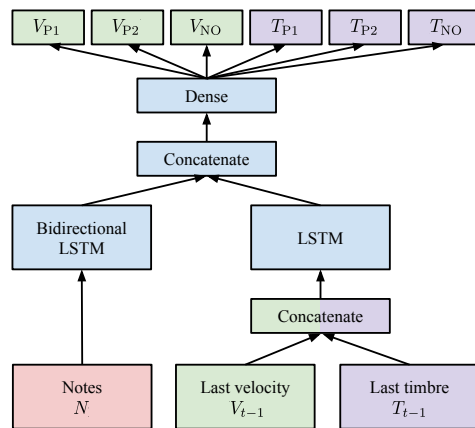


Figure 2: LSTM Note+Auto expressive performance model that observes both the score and its prior output.

Unlike for separated composition, there are no well-established baselines for multi-instrumental expressive performance, and thus we design several approaches. For the autoregressive formulation, our most-sophisticated model (Fig. 2) uses a bidirectional LSTM to process the separated score, and a forward-directional LSTM for the autoregressive expressive characteristics. The represen-

Model	Negative log-likelihood						Accuracy					
	Single voice				Aggregate		Single voice				Aggregate	
	P1	P2	TR	NO	POI	All	P1	P2	TR	NO	POI	All
Random	4.36	4.36	4.49	2.83	16.04	16.04	.013	.013	.011	.059	.024	.024
Unigram	4.00	3.77	3.01	2.50	13.27	11.53	.020	.022	.057	.061	.040	.369
Bigram	4.91	4.93	4.15	3.52	17.50	3.63	.000	.000	.000	.000	.000	.831
RNN Soloists	4.92	4.90	3.59	2.23	15.64	3.11	.000	.000	.004	.183	.047	.830
LSTM Soloists	4.60	4.30	3.01	1.91	13.82	2.70	.014	.008	.125	.246	.098	.838
LSTM Quartet	3.87	3.71	2.45	1.62	11.65	2.21	.028	.031	.294	.449	.201	.854
DeepBach [13]	0.82	1.01	0.63	0.83	3.28	0.75	.781	.729	.784	.748	.761	.943

Table 3: Results for separated composition experiments. For each instrument, negative log-likelihood and accuracy are calculated at points of interest (POIs). We also calculate aggregate statistics at POIs and globally (All). While DeepBach [13] achieves the best statistical performance, it uses future context and hence is more expensive to sample from.

Model	Negative log-likelihood						Accuracy							
	Single voice					Aggregate		Single voice					Aggregate	
	V_{P1}	V_{P2}	V_{NO}	T_{P1}	T_{P2}	POI	All	V_{P1}	V_{P2}	V_{NO}	T_{P1}	T_{P2}	POI	All
Random	2.77	2.77	2.77	1.39	1.39	11.09	11.09	.062	.062	.062	.250	.250	.138	.138
Unigram	2.87	2.89	3.04	1.35	1.33	11.47	9.65	.020	.022	.061	.006	.004	.023	.309
Bigram	2.82	2.85	2.78	4.27	4.27	17.00	4.57	.000	.000	.000	.000	.000	.000	.741
MultiReg Note	2.74	2.72	2.23	1.27	1.18	10.13	8.49	.106	.122	.292	.406	.507	.287	.359
MultiReg Note+Auto	2.58	2.56	2.04	2.90	2.48	12.56	4.32	.073	.100	.345	.071	.096	.137	.752
LSTM Note	2.68	2.63	2.09	1.32	1.21	9.94	8.28	.115	.134	.305	.456	.532	.308	.365
LSTM Note+Auto	1.93	1.89	1.99	2.23	1.89	9.93	3.42	.305	.321	.386	.241	.432	.337	.774

Table 4: Results for expressive performance experiments evaluated at points of interest (POI). Results are broken down by expression category (e.g. V_{NO} is noise velocity, T_{P1} is pulse 1 timbre) and aggregated at POIs and globally (All).

tations from the composition and autoregressive modules are merged and processed by an additional dense layer before projecting to six softmaxes, one for each of V_{P1} , V_{P2} , V_{NO} , T_{P1} , T_{P2} , and T_{NO} . We compare this model (LSTM Note+Auto) to a version which removes the autoregressive module and only sees the separated score (LSTM Note).

We also measure performance of simple multinomial regression baselines. The non-autoregressive baseline (MultiReg Note) maps the concatenation of N_{P1} , N_{P2} , N_{TR} , and N_{NO} directly to the six categorical outputs representing velocity and timbre (no temporal context). An autoregressive version of this model (MultiReg Note+Auto) takes additional inputs consisting of the previous timestep for the six velocity and timbre categories. Additionally, we show results for simple baselines (per-category unigram and bigram distributions) which do not consider N . Because the noise timbre field T_{NO} is so rarely used (less than 0.2% of all timesteps), we exclude it from our quantitative evaluation. Results are shown in Table 4.

Similarly to the musical notes in the separated composition task (Section 4.1), the high rate of NES-MDB results in substantial redundancy across timesteps. Averaged across all velocity and timbre categories, any of these categories at a given timestep has a 74% chance of having the same value as the previous timestep.

The performance of the LSTM Note model is comparable to that of the LSTM Note+Auto model at POIs, however the global performance of the LSTM Note+Auto model is substantially better. Intuitively, this suggests that the score is useful for knowing *when* to change, while the past velocity and timbre values are useful for knowing

Model	NES-MDB	PM	NH	MD	BC
Random	61.00	61.00	61.00	61.00	61.00
Note 1-gram [2]	8.71	11.05	10.25	11.51	11.06
Chord 1-gram [2]	8.76	27.64	5.94	19.03	12.22
GMM [2]	12.86	15.84	7.87	12.20	11.90
NADE [2]	8.53	10.28	5.48	10.06	7.19
RNN [2]	3.04	8.37	4.46	8.13	8.71
RNN-NADE [2]	2.62	7.48	2.91	6.74	5.83
LSTM	2.54	8.31	3.49	6.35	8.72
LSTM-NADE [17]	2.48	7.36	2.02	5.02	6.00

Table 5: Negative log-likelihoods for various models on the blended score format (Fig. 1a, Eq. 2) of NES-MDB. We also show results for Piano-midi.de (PM), Nottingham (NH), MuseData (MD), and the chorales of J.S. Bach (BC).

what value to output next. Interestingly, the MultiReg Note model has better performance at POIs than the MultiReg Note+Auto model. The latter overfit more quickly which may explain its inferior performance despite the fact that it sees strictly more information than the note-only model.

4.3 Blended composition experiments

In Table 5, we report the performance of several models on the blended composition task (Eq. 2). In NES-MDB, blended scores consist of 88 possible notes with a maximum of three simultaneous voices (noise generator is discarded). This task, standardized in [2], does not preserve the voicing of the score, and thus it is not immediately useful for generating NES music. Nevertheless, modeling blended scores of polyphonic music has become a standard benchmark for sequential models [5, 18], and NES-MDB

may be useful as a larger dataset in the same format.

In general, models assign higher likelihood to NES-MDB than the four other datasets after training. As with our other two tasks, this is likely due to the fact that NES-MDB is sampled at a higher temporal rate, and thus the average deviation across timesteps is lower. Due to its large size, a benefit of examining NES-MDB in this context is that sequential models tend to take longer to overfit the dataset than they do for the other four. We note that our implementations of these models may deviate slightly from those of the original authors, though our models achieve comparable results to those reported in [2, 17] when trained on the original datasets.

5. RELATED WORK

There are several popular datasets commonly used in statistical music composition. A dataset consisting of the entirety of J.S. Bach’s four-voice chorales has been extensively studied under the lenses of algorithmic composition and reharmonization [1, 2, 13, 14]. Like NES-MDB, this dataset has a fixed number of voices and can be represented as a separated score (Fig. 1b), however it is small in size (389 chorales) and lacks expressive information. Another popular dataset is Piano-midi.de, a corpus of classical piano from various composers [27]. This dataset has expressive timing and dynamics information but has heterogeneous time periods and only features solo piano music. Alongside Bach’s chorales and the Piano-midi.de dataset, Boulanger-Lewandowski et al. (2012) standardized the Nottingham collection of folk tunes and MuseData library of orchestral and piano classical music into blended score format (Fig. 1a).

Several other symbolic datasets exist containing both compositional and expressive characteristics. The Magaloff Corpus [10] consists of Disklavier recordings of a professional pianist playing the entirety of Chopin’s solo piano works. The Lakh MIDI dataset [28] is the largest corpus of symbolic music assembled to date with nearly 200k songs. While substantially larger than NES-MDB, the dataset has unconstrained polyphony, inconsistent expressive characteristics, and encompasses a wide variety of genres, instruments and time periods. Another paper trains neural networks on transcriptions of video game music [9], though their dataset only includes a handful of songs.

5.1 Statistical composition

While most of the early research in algorithmic music composition focused on expert systems [25], statistical approaches have since become the predominant approach. Mozer (1994) trained RNNs on monophonic melodies using a formulation similar to Eq. 1, finding the composed results to compare favorably to those from a trigram model. Others have also explored monophonic melody generation with RNNs [8, 26]. Boulanger-Lewandowski et al. (2012) standardize the polyphonic prediction task for blended scores (Eq. 2), measuring performance of a multitude of classical baselines against RNNs [30], restricted Boltz-

mann machines [34], and NADEs [21] on polyphonic music datasets. Several papers [5, 17, 35] directly compare to their results. Statistical models of music have also been employed as symbolic priors to assist music transcription algorithms [2, 4, 24].

Progressing towards models that *assist* humans in composition, many researchers study models to create new harmonizations for existing musical material. Allan and Williams (2005) train HMMs to create new harmonizations for Bach chorales [1]. Hadjeres et al. (2017) train a bidirectional RNN model to consider past and future temporal context (Eq. 3) [13]. Along with [16, 31], they advocate for the usage of Gibbs sampling to generate music from complex graphical models.

5.2 Statistical performance

Musicians perform music expressively by interpreting a performance with appropriate dynamics, timing and articulation. Computational models of expressive music performance seek to automatically assign such attributes to a score [36]. We point to several extensive surveys for information about the long history of rule-based systems [7, 12, 20, 36].

Several statistical models of expressive performance have also been proposed. Raphael (2010) learns a graphical model that automates an accompanying orchestra for a soloist, operating on acoustic features rather than symbolic [29]. Flossmann et al. (2013) build a system to control velocity, articulation and timing of piano performances by learning a graphical model from a large symbolic corpus of human performances [11]. Xia et al. (2015) model the expressive timing and dynamics of piano duet performances using spectral methods [37]. Two end-to-end systems attempt to jointly learn the semantics of composition and expressive performance using RNNs [23, 33]. Malik and Ek (2017) train an RNN to generate velocity information given a musical score [22]. These approaches differ from our own in that they focus on piano performances rather than multi-instrumental music.

6. CONCLUSION

The NES Music Database is a large corpus for examining multi-instrumental polyphonic composition and expressive performance generation. Compared to existing datasets, NES-MDB allows for examination of the “full pipeline” of music composition and performance. We parse the machine code of NES music into familiar formats (e.g. MIDI), eliminating the need for researchers to understand low-level details of the game system. We also provide an open-source tool which converts between the simpler formats and machine code, allowing researchers to audition their generated results as waveforms rendered by the NES. We hope that this dataset will facilitate a new paradigm of research on music generation—one that emphasizes the importance of expressive performance. To this end, we establish several baselines with reproducible evaluation methodology to encourage further investigation.

7. ACKNOWLEDGEMENTS

We would like to thank Louis Pisha for invaluable advice on the technical details of this project. Additionally, we would like to thank Nicolas Boulanger-Lewandowski, Eunjeong Stella Koh, Steven Merity, Miller Puckette, and Cheng-i Wang for helpful conversations throughout this work. This work was supported by UC San Diego's Chancellor's Research Excellence Scholarship program. GPUs used in this research were donated by NVIDIA.

8. REFERENCES

- [1] Moray Allan and Christopher Williams. Harmonising chorales by probabilistic inference. In *Proc. NIPS*, 2005.
- [2] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proc. ICML*, 2012.
- [3] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation-a survey. *arXiv:1709.01620*, 2017.
- [4] Ali Taylan Cemgil. Bayesian music transcription. *PhD thesis, Radboud University Nijmegen*, 2004.
- [5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Workshops*, 2014.
- [6] Karen Collins. *Game sound: an introduction to the history, theory, and practice of video game music and sound design*. MIT Press, 2008.
- [7] Miguel Delgado, Waldo Fajardo, and Miguel Molina-Solana. A state of the art on computational music performance. *Expert systems with applications*, 2011.
- [8] Douglas Eck and Jürgen Schmidhuber. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In *Proc. Neural Networks for Signal Processing*, 2002.
- [9] Otto Fabius and Joost R van Amersfoort. Variational recurrent auto-encoders. In *ICLR Workshops*, 2015.
- [10] Sebastian Flossmann, Werner Goebel, Maarten Grachten, Bernhard Niedermayer, and Gerhard Widmer. The Magaloff project: An interim report. *Journal of New Music Research*, 2010.
- [11] Sebastian Flossmann, Maarten Grachten, and Gerhard Widmer. Expressive performance rendering with probabilistic models. In *Guide to Computing for Expressive Music Performance*. 2013.
- [12] Werner Goebel, Simon Dixon, Giovanni De Poli, Anders Friberg, Roberto Bresin, and Gerhard Widmer. Sense in expressive music performance: Data acquisition, computational studies, and models. 2008.
- [13] Gaëtan Hadjeres and François Pachet. DeepBach: A steerable model for Bach chorales generation. In *Proc. ICML*, 2017.
- [14] Hermann Hild, Johannes Feulner, and Wolfram Menzel. Harmonet: A neural net for harmonizing chorales in the style of JS Bach. In *NIPS*, 1992.
- [15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.
- [16] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. In *Proc. ISMIR*, 2017.
- [17] Daniel D Johnson. Generating polyphonic music using tied parallel networks. In *Proc. International Conference on Evolutionary and Biologically Inspired Music and Art*, 2017.
- [18] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proc. ICML*, 2015.
- [19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.
- [20] Alexis Kirke and Eduardo R Miranda. An overview of computer systems for expressive music performance. In *Guide to computing for expressive music performance*. 2013.
- [21] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proc. AISTATS*, 2011.
- [22] Iman Malik and Carl Henrik Ek. Neural translation of musical style. *arXiv:1708.03535*, 2017.
- [23] Huanru Henry Mao, Taylor Shin, and Garrison W. Cottrell. DeepJ: Style-specific music generation. In *Proc. International Conference on Semantic Computing*, 2018.
- [24] Juhan Nam, Jiquan Ngiam, Honglak Lee, and Malcolm Slaney. A classification-based polyphonic piano transcription approach using learned feature representations. In *Proc. ISMIR*, 2011.
- [25] Gerhard Nierhaus. *Algorithmic composition: paradigms of automated music generation*. Springer Science & Business Media, 2009.
- [26] Jean-Francois Paiement, Samy Bengio, and Douglas Eck. Probabilistic models for melodic prediction. *Artificial Intelligence*, 2009.

- [27] Graham E Poliner and Daniel PW Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, 2006.
- [28] Colin Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.
- [29] Christopher Raphael. Music Plus One and machine learning. In *Proc. ICML*, 2010.
- [30] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.
- [31] Jason Sakellariou, Francesca Tria, Vittorio Loreto, and François Pachet. Maximum entropy model for melodic patterns. In *ICML Workshops*, 2015.
- [32] Craig Stuart Sapp. Comparative analysis of multiple musical performances. In *Proc. ISMIR*, 2007.
- [33] Ian Simon and Sageev Oore. Performance RNN: Generating music with expressive timing and dynamics, 2017.
- [34] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document, 1986.
- [35] Raunaq Vohra, Kratarth Goel, and JK Sahoo. Modeling temporal dependencies in data using a DBN-LSTM. In *Proc. IEEE Conference on Data Science and Advanced Analytics*, 2015.
- [36] Gerhard Widmer and Werner Goebel. Computational models of expressive music performance: The state of the art. *Journal of New Music Research*, 2004.
- [37] Guangyu Xia, Yun Wang, Roger B Dannenberg, and Geoffrey Gordon. Spectral learning for expressive interactive ensemble music performance. In *Proc. ISMIR*, 2015.

AUDIO-ALIGNED JAZZ HARMONY DATASET FOR AUTOMATIC CHORD TRANSCRIPTION AND CORPUS-BASED RESEARCH

Vsevolod Eremenko, Emir Demirel, Baris Bozkurt, Xavier Serra

Music Technology Group, Universitat Pompeu Fabra, Barcelona

firstname.lastname@upf.edu

ABSTRACT

In this paper we present a new dataset of time-aligned jazz harmony transcriptions. This dataset is a useful resource for content-based analysis, especially for training and evaluating chord transcription algorithms. Most of the available chord transcription datasets only contain annotations for rock and pop, and the characteristics of jazz, such as the extensive use of seventh chords, are not represented. Our dataset consists of annotations of 113 tracks selected from “The Smithsonian Collection of Classic Jazz” and “Jazz: The Smithsonian Anthology,” covering a range of performers, subgenres, and historical periods. Annotations were made by a jazz musician and contain information about the meter, structure, and chords for entire audio tracks. We also present evaluation results of this dataset using state-of-the-art chord estimation algorithms that support seventh chords. The dataset is valuable for jazz scholars interested in corpus-based research. To demonstrate this, we extract statistics for symbolic data and chroma features from the audio tracks.

1. INTRODUCTION

Musicians in many genres use an abbreviated notation, known as a lead sheet, to represent chord progressions. Digitized collections of lead sheets are used for computer-aided corpus-based musicological research, e.g., [6, 13, 18, 31]. Lead sheets do not provide information about how specific chords are rendered by musicians [21]. To reflect this rendering, music information retrieval (MIR) and musicology communities have created several datasets of audio recordings annotated with chord progressions. Such collections are used for training and evaluating various MIR algorithms (e.g., Automatic Chord Estimation) and for corpus-based research.

Because providing chord annotations for audio is time-consuming and requires qualified annotators, there are few such datasets available for MIR research. Of the existing datasets, most are of rock and pop music, with very few

available for jazz. A balanced and comprehensive corpus of jazz audio recordings with chord transcription would be a useful resource for developing MIR algorithms aimed to serve jazz scholars. The particularities of jazz also allow us to view the dataset’s format, content selection, and chord estimation accuracy evaluation from a different angle.

This paper starts with a review of publicly available datasets that contain information about harmony, such as chord progressions and structural analysis. Based on this review, we justify the necessity of creating a representative, balanced jazz dataset in a new format. We present our dataset, which contains lead sheet style chords, beat onsets, and structure annotations for a selection of jazz audio tracks, along with full-length annotations for each recording. We explain our track selection principle and transcription methodology, and also provide pre-calculated chroma features [27] for the entire dataset. We then discuss how to evaluate the performance of Automatic Chord Transcription on jazz recordings. Moreover, baseline evaluation scores for two state-of-the-art chord estimation algorithms are shown. Dataset is available online¹.

2. RELATED WORKS

2.1 Chord annotated audio datasets

Here we review existing datasets with respect to their format, content selection principle, annotation methodology, and their uses in research. We then discuss some discrepancies in different approaches to chord annotation, as well as the advantages and drawbacks of different formats.

2.1.1 *Isophonics family*

Isophonics² is one of the first time-aligned chord annotation datasets, introduced in [17]. Initially, the dataset consisted of twelve studio albums by The Beatles. Harte justified his selection by stating that it is a small but varied corpus (including various styles, recording techniques and “complex harmonic progressions” in comparison with other popular music artists). These albums are “widely available in most parts of the world” and have had enormous influence on the development of pop music. A number of related theoretical and critical works was also taken into account. Later the corpus was augmented with some



© Vsevolod Eremenko, Emir Demirel, Baris Bozkurt, Xavier Serra. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Vsevolod Eremenko, Emir Demirel, Baris Bozkurt, Xavier Serra. “Audio-Aligned Jazz Harmony Dataset for Automatic Chord Transcription and Corpus-based Research”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ <http://doi.org/10.5281/zenodo.1290736>. Documentation: <https://mtg.github.io/JAAH>

² Available at <http://isophonics.net/content/reference-annotations>

transcriptions of Carole King, Queen, Michael Jackson, and Zweieck.

The corpus is organized as a directory of “.lab” files³. Each line describes a chord segment with a start time, end time (in seconds), and chord label in the “Harte et al.” format [16]. The annotator recorded chord start times by tapping keys on a keyboard. The chords were transcribed using published analyses as a starting point, if possible. Notes from the melody line were not included in the chords. The resulting chord progression was verified by synthesizing the audio and playing it alongside the original tracks. The dataset has been used for training and testing chord evaluation algorithms (e.g., for MIREX⁴).

The same format is used for the “Robbie Williams dataset”⁵ announced in [12]; for the chord annotations of the RWC and USPop datasets⁶; and for the datasets by Deng: JayChou29, CNPop20, and JazzGuitar99⁷. Deng presented this dataset in [11], and it is the only one in the family which is related to jazz. However, it uses 99 short, guitar-only pieces recorded for a study book, and thus does not reflect the variety of jazz styles and instrumentations.

2.1.2 Billboard

Authors of the Billboard⁸ dataset argued that both musicologists and MIR researchers require a wider range of data [7]. They selected songs randomly from the Billboard “Hot 100” chart in the United States between 1958 and 1991.

Their format is close to the traditional lead sheet: it contains meter, bars, and chord labels for each bar or for particular beats of a bar. Annotations are time-aligned with the audio by the assignment of a timestamp to the start of each “phrase” (usually 4 bars). The “Harte et al.” syntax was used for the chord labels (with a few additions to the shorthand system). The authors accompanied the annotations with pre-extracted NNLS Chroma features [27]. At least three persons were involved in making and reconciling a singleton annotation for each track. The corpus is used for training and testing chord evaluation algorithms (e.g., MIREX ACE evaluation) and for musicological research [13].

2.1.3 Rockcorpus and subjectivity dataset

Rockcorpus⁹ was announced in [9]. The corpus currently contains 200 songs selected from the “500 Greatest Songs of All Time” list, which was compiled by the writers of *Rolling Stone* magazine, based on polls of 172 “rock stars and leading authorities.”

As in the Billboard dataset, the authors specify the structure segmentation and assign chords to bars (and to

beats if necessary), but not directly to time segments. A timestamp is specified for each measure bar.

In contrast to the previous datasets, authors do not use “absolute” chord labels, e.g., C:maj. Instead, they specify tonal centers for parts of the composition and chords as Roman numerals. These show the chord quality and the relation of the chord’s root to the tonic. This approach facilitates harmony analysis.

Each of the two authors provides annotations for each recording. As opposed to the aforementioned examples, the authors do not aim to produce a single “ground truth” annotation, but keep both versions. Thus it becomes possible to study subjectivity in human annotations of chord changes. The Rockcorpus is used for training and testing chord evaluation algorithms [19], and for musicological research [9].

Concerning the study of subjectivity, we should also mention the Chordify Annotator Subjectivity Dataset¹⁰, which contains transcriptions of 50 songs from the Billboard dataset by four different annotators [22]. It uses JSON-based JAMS annotation format.

2.2 Jazz-related datasets

Here we review datasets which do not have audio-aligned chord annotations as their primary purpose, but nevertheless can be useful in the context of jazz harmony studies.

2.2.1 Weimar Jazz Database

The main focus of the Weimar Jazz Database (WJazzD¹¹) is jazz soloing. Data is disseminated as a SQLite database containing transcription and meta information about 456 instrumental jazz solos from 343 different recordings (more than 132000 beats over 12.5 hours). The database includes: meter, structure segmentation, measures, and beat onsets, along with chord labels in a custom format. However, as stated by Pfeleiderer [30], the chords were taken from available lead sheets, “cloned” for all choruses of the solo, and only in some cases transcribed from what was actually played by the rhythm section.

The database’s metadata includes the MusicBrainz¹² Identifier, which allows users to link the annotation to a particular audio recording and fetch meta-information about the track from the MusicBrainz server.

Although WJazzD has significant applications for research in the symbolic domain [30], our experience has shown that obtaining audio tracks for analysis and aligning them with the annotations is nontrivial: the MusicBrainz identifiers are sometimes wrong, and are missing for 8% of the tracks. Sometimes WJazzD contains annotations of rare or old releases. In different masterings, the tempo and therefore the beat positions, differs from modern and widely available releases. We matched 14 tracks from WJazzD to tracks in our dataset by the performer’s name

³ ASCII plain text files which are used by a variety of popular MIR tools, e.g., Sonic Visualizer [8].

⁴ http://www.music-ir.org/mirex/wiki/MIREX_HOME

⁵ <http://ispg.deib.polimi.it/mir-software.html>

⁶ <https://github.com/tmc323/Chord-Annotations>

⁷ <http://www.tangkk.net/label>

⁸ <http://ddmal.music.mcgill.ca/research/billboard>

⁹ http://rockcorpus.midside.com/2011_paper.html

¹⁰ <https://github.com/chordify/CASD>

¹¹ <http://jazzomat.hfm-weimar.de>

¹² A community-supported collection of music recording metadata: <https://musicbrainz.org>

and the date of the recording. In three cases the MusicBrainz Release is missing, and in three cases rare compilations were used as sources. It took some time to discover that three of the tracks (“Embraceable You”, “Lester Leaps In”, “Work Song”) are actually alternative takes, which are officially available only on extended reissues. Beat positions in the other eleven tracks must be shifted and sometimes scaled to match available audio (e.g., for “Walking Shoes”). This may be improved by using an interesting alternative introduced by Balke et al. [3]: a web-based application, “JazzTube,” which matches YouTube videos with WJazzD annotations and provides interactive educational visualizations.

2.2.2 Symbolic datasets

The iRb¹³ dataset (announced in [6]) contains chord progressions for 1186 jazz standards taken from a popular internet forum for jazz musicians. It lists the composer, lyricist, and year of creation. The data are written in the Humdrum encoding system. The chord data are submitted by anonymous enthusiasts and thus provides a rather modern interpretation of jazz standards. Nevertheless, Broze and Shanahan proved it was useful for corpus-based musicology research: see [6] and [31].

“Charlie Parker’s Omnibook data”¹⁴ contains chord progressions, themes, and solo scores for 50 recordings by Charlie Parker. The dataset is stored in MusicXML and introduced in [10].

Granroth-Wilding’s “JazzCorpus”¹⁵ contains 76 chord progressions (approximately 3000 chords) annotated with harmonic analyses (i.e., tonal centers and roman numerals for the chords), with the primary goal of training and testing statistical parsing models for determining chord harmonic functions [15].

2.3 Discussion

2.3.1 Some discrepancies in chord annotation approaches in the context of jazz

An article by Harte et al. [16] de facto sets the standard for chord labels in MIR annotations. It describes the basic syntax and a shorthand system. The basic syntax explicitly defines a chord pitch class set. For example, C: (3, 5, b7) is interpreted as C, E, G, Bb. The shorthand system contains symbols which resemble chord representations on lead sheets (e.g., C: 7 stands for C dominant seventh). According to [16], C: 7 should be interpreted as C: (3, 5, b7). However, this may not always be the case in jazz. According to theoretical research [25] and educational books, e.g., [23], the 5th degree is omitted quite often in jazz harmony.

Generally speaking, since chord labels emerged in jazz and pop music practice in the 1930s, they provide a higher

level of abstraction than sheet music scores, allowing musicians to improvise their parts [21]. Similarly, a transcriber can use the single chord label C: 7 to mark the whole passage containing the walking bass line and comping piano phrase, without even noticing, “Is the 5th really played?” Thus, for jazz corpus annotation, we suggest accepting the “Harte et al.” syntax for the purpose of standardization, but sticking to shorthand system and avoiding a literal interpretation of the labels.

There are two different approaches to chord annotation:

- “Lead sheet style.” Contains a lead sheet [21], which has obvious meaning to musicians practicing the corresponding style (e.g., jazz or rock). It is aligned to audio with timestamps for beats or measure bars. Chords are considered in a rhythmical framework. This style is convenient because the annotation process can be split into two parts: lead sheet transcription done by a qualified musician, and beats annotation done by a less skilled person or sometimes even automatically performed.
- “Isophonics style.” Chord labels are bound to absolute time segments.

We must note that musicians use chord labels for instructing and describing performance mostly within the lead sheet framework. While the lead sheet format and the chord-beats relationship is obvious, detecting and interpreting “chord onset” times in jazz is an unclear task. The widely used comping approach to accompaniment [23] assumes playing phrases instead of long isolated chords, and a given phrase does not necessarily start with a chord tone. Furthermore, individual players in the rhythm section (e.g., bassist and guitarist) may choose different strategies: they may anticipate a new chord, play it on the downbeat, or delay. Thus, before annotating “chord onset” times, we should make sure that it makes musical and perceptual sense. All known corpus-based research is based on “lead sheet style” annotated datasets. Taking all these considerations into account, we prefer to use the lead sheet approach to chord annotations.

2.3.2 Criteria for format and dataset for chord annotated audio

Based on the given review and our own hands-on experience with chord estimation algorithm evaluation, we present our guidelines and propositions for building an audio-aligned chord dataset.

1. Clearly define dataset boundaries (e.g., a certain music style or time period). The selection of audio tracks should be representative and balanced within these boundaries.
2. Since sharing audio is restricted by copyright laws, use recent releases and existing compilations to facilitate access to dataset audio.
3. Use the time-aligned lead sheet approach with “shorthand” chord labels from [16], but avoid their literal interpretation.

¹³ https://musiccog.ohio-state.edu/home/index.php/iRb_Jazz_Corpus

¹⁴ <https://members.loria.fr/KDeguernel/omnibook/>

¹⁵ <http://jazzparser.granroth-wilding.co.uk/JazzCorpus.html>

4. Annotate entire tracks, but not excerpts. This makes it possible to explore structure and self-similarity.
5. Provide the MusicBrainz identifier to exploit meta-information from this service. If feasible, add meta-information to MusicBrainz instead of storing it privately within the dataset.
6. Annotate in a format that is not only machine readable, but convenient for further manual editing and verification. Relying on plain text files and specific directory structure for storing heterogeneous annotation is not practical for users. JSON-based JAMS format introduced by Humphrey et al. [20] solves this issue, but currently does not support lead sheet chord annotation. It is verbose in order to be comfortably used by the human annotators and supervisors.
7. Include pre-extracted chroma features. This makes it possible to conduct some MIR experiments without accessing the audio. It would be interesting to incorporate chroma features into corpus-based research to demonstrate how a particular chord class is rendered in a particular recording.

3. PROPOSED DATASET

3.1 Data format and annotation attributes

Taking into consideration the discussion from the previous section, we decided to use the JSON format. An excerpt from an annotation is shown in Figure 1. We provide the track title, artist name, and MusicBrainz ID. The start time, duration of the annotated region, and tuning frequency estimated automatically by Essentia [5] are shown. The beat onsets array and chord annotations are nested into the “parts” attribute, which in turn could recursively contain “parts.” This hierarchy represents the structure of the musical piece. Each part has a “name” attribute which describes the purpose of the part, such as “intro,” “head,” “coda,” “outro,” “interlude,” etc. The inner form of the chorus (e.g., AABA, ABAC, blues) and predominant instrumentation (e.g., ensemble, trumpet solo, vocals female, etc.) are annotated explicitly. This structural annotation is beneficial for extracting statistical information regarding the type of chorus present in the dataset, as well as other musically important properties. We made chord annotations in the lead sheet style: each annotation string represents a sequence of measure bars, delimited with pipes: “|”. A sequence starts and ends with a pipe as well. Chords must be specified for each beat in a bar (e.g., four chords for 4/4 meter). A simplification of this is possible: if a chord occupies the whole bar, it could be typed only once; and if chords occupy an equal number of beats in a bar (e.g., two beats in 4/4 metre), each chord could be specified only once, e.g., |F G| instead of |F F G G|.

For chord labeling, we use the Harte et al. [16] syntax for standardization reasons, but mainly use the shorthand system and do not assume the literal interpretation of labels

```
{
  "mbid": "e8ace66a-d3f0-4ab9-b489-3deb44432575",
  "duration": 155.42,
  "artist": "Django Reinhardt",
  "title": "Dinah",
  "tuning": 443.36,
  "metre": "4/4",
  "parts": [
    {
      "name": "intro...",
      "parts": [
        {
          "name": "A",
          "beats": [
            4.86, 5.14, 5.42, 5.7, 5.97, 6.25, 6.53, 6.8, 7.07, 7.35, 7.64, 7.92, 8.2, 8.48, 8.76, 9.04, 9.32, 9.6, 9.88, 10.16, 10.44, 10.72, 11.0, 11.28, 11.56, 11.84, 12.12, 12.4, 12.68, 12.96, 13.24, 13.52, 13.8, 14.08, 14.36, 14.64, 14.92, 15.2, 15.48, 15.76, 16.04, 16.32, 16.6, 16.88, 17.16, 17.44, 17.72, 18.0, 18.28, 18.56, 18.84, 19.12, 19.4, 19.68, 19.96, 20.24, 20.52, 20.8, 21.08, 21.36, 21.64, 21.92, 22.2, 22.48, 22.76, 23.04, 23.32, 23.6, 23.88, 24.16, 24.44, 24.72, 25.0, 25.28, 25.56, 25.84, 26.12, 26.4, 26.68, 26.96, 27.24, 27.52, 27.8, 28.08, 28.36, 28.64, 28.92, 29.2, 29.48, 29.76, 30.04, 30.32, 30.6, 30.88, 31.16, 31.44, 31.72, 32.0, 32.28, 32.56, 32.84, 33.12, 33.4, 33.68, 33.96, 34.24, 34.52, 34.8, 35.08, 35.36, 35.64, 35.92, 36.2, 36.48, 36.76, 37.04, 37.32, 37.6, 37.88, 38.16, 38.44, 38.72, 39.0, 39.28, 39.56, 39.84, 40.12, 40.4, 40.68, 40.96, 41.24, 41.52, 41.8, 42.08, 42.36, 42.64, 42.92, 43.2, 43.48, 43.76, 44.04, 44.32, 44.6, 44.88, 45.16, 45.44, 45.72, 46.0, 46.28, 46.56, 46.84, 47.12, 47.4, 47.68, 47.96, 48.24, 48.52, 48.8, 49.08, 49.36, 49.64, 49.92, 50.2, 50.48, 50.76, 51.04, 51.32, 51.6, 51.88, 52.16, 52.44, 52.72, 53.0, 53.28, 53.56, 53.84, 54.12, 54.4, 54.68, 54.96, 55.24, 55.52, 55.8, 56.08, 56.36, 56.64, 56.92, 57.2, 57.48, 57.76, 58.04, 58.32, 58.6, 58.88, 59.16, 59.44, 59.72, 60.0, 60.28, 60.56, 60.84, 61.12, 61.4, 61.68, 61.96, 62.24, 62.52, 62.8, 63.08, 63.36, 63.64, 63.92, 64.2, 64.48, 64.76, 65.04, 65.32, 65.6, 65.88, 66.16, 66.44, 66.72, 67.0, 67.28, 67.56, 67.84, 68.12, 68.4, 68.68, 68.96, 69.24, 69.52, 69.8, 70.08, 70.36, 70.64, 70.92, 71.2, 71.48, 71.76, 72.04, 72.32, 72.6, 72.88, 73.16, 73.44, 73.72, 74.0, 74.28, 74.56, 74.84, 75.12, 75.4, 75.68, 75.96, 76.24, 76.52, 76.8, 77.08, 77.36, 77.64, 77.92, 78.2, 78.48, 78.76, 79.04, 79.32, 79.6, 79.88, 80.16, 80.44, 80.72, 81.0, 81.28, 81.56, 81.84, 82.12, 82.4, 82.68, 82.96, 83.24, 83.52, 83.8, 84.08, 84.36, 84.64, 84.92, 85.2, 85.48, 85.76, 86.04, 86.32, 86.6, 86.88, 87.16, 87.44, 87.72, 88.0, 88.28, 88.56, 88.84, 89.12, 89.4, 89.68, 89.96, 90.24, 90.52, 90.8, 91.08, 91.36, 91.64, 91.92, 92.2, 92.48, 92.76, 93.04, 93.32, 93.6, 93.88, 94.16, 94.44, 94.72, 95.0, 95.28, 95.56, 95.84, 96.12, 96.4, 96.68, 96.96, 97.24, 97.52, 97.8, 98.08, 98.36, 98.64, 98.92, 99.2, 99.48, 99.76, 100.04, 100.32, 100.6, 100.88, 101.16, 101.44, 101.72, 102.0, 102.28, 102.56, 102.84, 103.12, 103.4, 103.68, 103.96, 104.24, 104.52, 104.8, 105.08, 105.36, 105.64, 105.92, 106.2, 106.48, 106.76, 107.04, 107.32, 107.6, 107.88, 108.16, 108.44, 108.72, 109.0, 109.28, 109.56, 109.84, 110.12, 110.4, 110.68, 110.96, 111.24, 111.52, 111.8, 112.08, 112.36, 112.64, 112.92, 113.2, 113.48, 113.76, 114.04, 114.32, 114.6, 114.88, 115.16, 115.44, 115.72, 116.0, 116.28, 116.56, 116.84, 117.12, 117.4, 117.68, 117.96, 118.24, 118.52, 118.8, 119.08, 119.36, 119.64, 119.92, 120.2, 120.48, 120.76, 121.04, 121.32, 121.6, 121.88, 122.16, 122.44, 122.72, 123.0, 123.28, 123.56, 123.84, 124.12, 124.4, 124.68, 124.96, 125.24, 125.52, 125.8, 126.08, 126.36, 126.64, 126.92, 127.2, 127.48, 127.76, 128.04, 128.32, 128.6, 128.88, 129.16, 129.44, 129.72, 130.0, 130.28, 130.56, 130.84, 131.12, 131.4, 131.68, 131.96, 132.24, 132.52, 132.8, 133.08, 133.36, 133.64, 133.92, 134.2, 134.48, 134.76, 135.04, 135.32, 135.6, 135.88, 136.16, 136.44, 136.72, 137.0, 137.28, 137.56, 137.84, 138.12, 138.4, 138.68, 138.96, 139.24, 139.52, 139.8, 140.08, 140.36, 140.64, 140.92, 141.2, 141.48, 141.76, 142.04, 142.32, 142.6, 142.88, 143.16, 143.44, 143.72, 144.0, 144.28, 144.56, 144.84, 145.12, 145.4, 145.68, 145.96, 146.24, 146.52, 146.8, 147.08, 147.36, 147.64, 147.92, 148.2, 148.48, 148.76, 149.04, 149.32, 149.6, 149.88, 150.16, 150.44, 150.72, 151.0, 151.28, 151.56, 151.84, 152.12, 152.4, 152.68, 152.96, 153.24, 153.52, 153.8, 154.08, 154.36, 154.64, 154.92, 155.2, 155.48, 155.76, 156.04, 156.32, 156.6, 156.88, 157.16, 157.44, 157.72, 158.0, 158.28, 158.56, 158.84, 159.12, 159.4, 159.68, 159.96, 160.24, 160.52, 160.8, 161.08, 161.36, 161.64, 161.92, 162.2, 162.48, 162.76, 163.04, 163.32, 163.6, 163.88, 164.16, 164.44, 164.72, 165.0, 165.28, 165.56, 165.84, 166.12, 166.4, 166.68, 166.96, 167.24, 167.52, 167.8, 168.08, 168.36, 168.64, 168.92, 169.2, 169.48, 169.76, 170.04, 170.32, 170.6, 170.88, 171.16, 171.44, 171.72, 172.0, 172.28, 172.56, 172.84, 173.12, 173.4, 173.68, 173.96, 174.24, 174.52, 174.8, 175.08, 175.36, 175.64, 175.92, 176.2, 176.48, 176.76, 177.04, 177.32, 177.6, 177.88, 178.16, 178.44, 178.72, 179.0, 179.28, 179.56, 179.84, 180.12, 180.4, 180.68, 180.96, 181.24, 181.52, 181.8, 182.08, 182.36, 182.64, 182.92, 183.2, 183.48, 183.76, 184.04, 184.32, 184.6, 184.88, 185.16, 185.44, 185.72, 186.0, 186.28, 186.56, 186.84, 187.12, 187.4, 187.68, 187.96, 188.24, 188.52, 188.8, 189.08, 189.36, 189.64, 189.92, 190.2, 190.48, 190.76, 191.04, 191.32, 191.6, 191.88, 192.16, 192.44, 192.72, 193.0, 193.28, 193.56, 193.84, 194.12, 194.4, 194.68, 194.96, 195.24, 195.52, 195.8, 196.08, 196.36, 196.64, 196.92, 197.2, 197.48, 197.76, 198.04, 198.32, 198.6, 198.88, 199.16, 199.44, 199.72, 200.0, 200.28, 200.56, 200.84, 201.12, 201.4, 201.68, 201.96, 202.24, 202.52, 202.8, 203.08, 203.36, 203.64, 203.92, 204.2, 204.48, 204.76, 205.04, 205.32, 205.6, 205.88, 206.16, 206.44, 206.72, 207.0, 207.28, 207.56, 207.84, 208.12, 208.4, 208.68, 208.96, 209.24, 209.52, 209.8, 210.08, 210.36, 210.64, 210.92, 211.2, 211.48, 211.76, 212.04, 212.32, 212.6, 212.88, 213.16, 213.44, 213.72, 214.0, 214.28, 214.56, 214.84, 215.12, 215.4, 215.68, 215.96, 216.24, 216.52, 216.8, 217.08, 217.36, 217.64, 217.92, 218.2, 218.48, 218.76, 219.04, 219.32, 219.6, 219.88, 220.16, 220.44, 220.72, 221.0, 221.28, 221.56, 221.84, 222.12, 222.4, 222.68, 222.96, 223.24, 223.52, 223.8, 224.08, 224.36, 224.64, 224.92, 225.2, 225.48, 225.76, 226.04, 226.32, 226.6, 226.88, 227.16, 227.44, 227.72, 228.0, 228.28, 228.56, 228.84, 229.12, 229.4, 229.68, 229.96, 230.24, 230.52, 230.8, 231.08, 231.36, 231.64, 231.92, 232.2, 232.48, 232.76, 233.04, 233.32, 233.6, 233.88, 234.16, 234.44, 234.72, 235.0, 235.28, 235.56, 235.84, 236.12, 236.4, 236.68, 236.96, 237.24, 237.52, 237.8, 238.08, 238.36, 238.64, 238.92, 239.2, 239.48, 239.76, 240.04, 240.32, 240.6, 240.88, 241.16, 241.44, 241.72, 242.0, 242.28, 242.56, 242.84, 243.12, 243.4, 243.68, 243.96, 244.24, 244.52, 244.8, 245.08, 245.36, 245.64, 245.92, 246.2, 246.48, 246.76, 247.04, 247.32, 247.6, 247.88, 248.16, 248.44, 248.72, 249.0, 249.28, 249.56, 249.84, 250.12, 250.4, 250.68, 250.96, 251.24, 251.52, 251.8, 252.08, 252.36, 252.64, 252.92, 253.2, 253.48, 253.76, 254.04, 254.32, 254.6, 254.88, 255.16, 255.44, 255.72, 256.0, 256.28, 256.56, 256.84, 257.12, 257.4, 257.68, 257.96, 258.24, 258.52, 258.8, 259.08, 259.36, 259.64, 259.92, 260.2, 260.48, 260.76, 261.04, 261.32, 261.6, 261.88, 262.16, 262.44, 262.72, 263.0, 263.28, 263.56, 263.84, 264.12, 264.4, 264.68, 264.96, 265.24, 265.52, 265.8, 266.08, 266.36, 266.64, 266.92, 267.2, 267.48, 267.76, 268.04, 268.32, 268.6, 268.88, 269.16, 269.44, 269.72, 270.0, 270.28, 270.56, 270.84, 271.12, 271.4, 271.68, 271.96, 272.24, 272.52, 272.8, 273.08, 273.36, 273.64, 273.92, 274.2, 274.48, 274.76, 275.04, 275.32, 275.6, 275.88, 276.16, 276.44, 276.72, 277.0, 277.28, 277.56, 277.84, 278.12, 278.4, 278.68, 278.96, 279.24, 279.52, 279.8, 280.08, 280.36, 280.64, 280.92, 281.2, 281.48, 281.76, 282.04, 282.32, 282.6, 282.88, 283.16, 283.44, 283.72, 284.0, 284.28, 284.56, 284.84, 285.12, 285.4, 285.68, 285.96, 286.24, 286.52, 286.8, 287.08, 287.36, 287.64, 287.92, 288.2, 288.48, 288.76, 289.04, 289.32, 289.6, 289.88, 290.16, 290.44, 290.72, 291.0, 291.28, 291.56, 291.84, 292.12, 292.4, 292.68, 292.96, 293.24, 293.52, 293.8, 294.08, 294.36, 294.64, 294.92, 295.2, 295.48, 295.76, 296.04, 296.32, 296.6, 296.88, 297.16, 297.44, 297.72, 298.0, 298.28, 298.56, 298.84, 299.12, 299.4, 299.68, 299.96, 300.24, 300.52, 300.8, 301.08, 301.36, 301.64, 301.92, 302.2, 302.48, 302.76, 303.04, 303.32, 303.6, 303.88, 304.16, 304.44, 304.72, 305.0, 305.28, 305.56, 305.84, 306.12, 306.4, 306.68, 306.96, 307.24, 307.52, 307.8, 308.08, 308.36, 308.64, 308.92, 309.2, 309.48, 309.76, 310.04, 310.32, 310.6, 310.88, 311.16, 311.44, 311.72, 312.0, 312.28, 312.56, 312.84, 313.12, 313.4, 313.68, 313.96, 314.24, 314.52, 314.8, 315.08, 315.36, 315.64, 315.92, 316.2, 316.48, 316.76, 317.04, 317.32, 317.6, 317.88, 318.16, 318.44, 318.72, 319.0, 319.28, 319.56, 319.84, 320.12, 320.4, 320.68, 320.96, 321.24, 321.52, 321.8, 322.08, 322.36, 322.64, 322.92, 323.2, 323.48, 323.76, 324.04, 324.32, 324.6, 324.88, 325.16, 325.44, 325.72, 326.0, 326.28, 326.56, 326.84, 327.12, 327.4, 327.68, 327.96, 328.24, 328.52, 328.8, 329.08, 329.36, 329.64, 329.92, 330.2, 330.48, 330.76, 331.04, 331.32, 331.6, 331.88, 332.16, 332.44, 332.72, 333.0, 333.28, 333.56, 333.84, 334.12, 334.4, 334.68, 334.96, 335.24, 335.52, 335.8, 336.08, 336.36, 336.64, 336.92, 337.2, 337.48, 337.76, 338.04, 338.32, 338.6, 338.88, 339.16, 339.44, 339.72, 340.0, 340.28, 340.56, 340.84, 341.12, 341.4, 341.68, 341.96, 342.24, 342.52, 342.8, 343.08, 343.36, 343.64, 343.92, 344.2, 344.48, 344.76, 345.04, 345.32, 345.6, 345.88, 346.16, 346.44, 346.72, 347.0, 347.28, 347.56, 347.84, 348.12, 348.4, 348.68, 348.96, 349.24, 349.52, 349.8, 350.08, 350.36, 350.64, 350.92, 351.2, 351.48, 351.76, 352.04, 352.32, 352.6, 352.88, 353.16, 353.44, 353.72, 354.0, 354.28, 354.56, 354.84, 355.12, 355.4, 355.68, 355.96, 356.24, 356.52, 356.8, 357.08, 357.36, 357.64, 357.92, 358.2, 358.48, 358.76, 359.04, 359.32, 359.6, 359.88, 360.16, 360.44, 360.72, 361.0, 361.28, 361.56, 361.84, 362.12, 362.4, 362.68, 362.96, 363.24, 363.52, 363.8, 364.08, 364.36, 364.64, 364.92, 365.2, 365.48, 365.76, 366.04, 366.32, 366.6, 366.88, 367.16, 367.44, 367.72, 368.0, 368.28, 368.56, 368.84, 369.12, 369.4, 369.68, 369.96, 370.24, 370.52, 370.8, 371.08, 371.36, 371.64, 371.92, 372.2, 372.48, 372.76, 373.04, 373.32, 373.6, 373.88, 374.16, 374.44, 374.72, 375.0, 375.28, 375.56, 375.84, 376.12, 376.4, 376.68, 376.96, 377.24, 377.52, 377.8, 378.08, 378.36, 378.64, 378.92, 379.2, 379.48, 379.76, 380.04, 380.32, 380.6, 380.88, 381.16, 381.44, 381.72, 382.0, 382.28, 382.56, 382.84, 383.12, 383.4, 383.68, 383.96, 384.24, 384.52, 384.8, 385.08, 385.36, 385.64, 385.92, 386.2, 386.48, 386.76, 387.04, 387.32, 387.6, 387.88, 388.16, 388.44, 388.72, 389.0, 389.28, 389.56, 389.84, 390.12, 390.4, 390.68, 390.96, 391.24, 391.52, 391.8, 392.08, 392.36, 392.64, 392.92, 393.2, 393.48, 393.76, 394.04, 394.32, 394.6, 394.88, 395.16, 395.44, 395.72, 396.0, 396.28, 396.56, 396.84, 397.12, 397.4, 397.68, 397.96, 398.24, 398.52, 398.8, 399.08, 399.36, 399.64, 399.92, 400.2, 400.48, 400.76, 401.04, 401.32, 401.6, 401.88, 402.16, 402.44, 402.72, 403.0, 403.28, 403.56, 403.84, 404.12, 404.4, 404.68, 404.96, 405.24, 405.52, 405.8, 406.08, 406.36, 406.64, 406.92, 407.2, 407.48, 407.76, 408.04, 408.32, 408.6, 408.88, 409.16, 409.44, 409.72, 410.0, 410.28, 410.56, 410.84, 411.12, 411.4, 411.68, 411.96, 412.24, 412.52, 412.8, 413.08, 413.36, 413.64, 413.92, 414.2, 414.48, 414.76, 415.04, 415.32, 415.6, 415.88, 416.16, 416.44, 416.72, 417.0, 417.28, 417.56, 417.84, 418.12, 418.4, 418.68, 418.96, 419.24, 419.52, 419.8, 420.08, 420.36, 420.64, 420.92, 421.2, 421.48, 421.76, 422.04, 422.32, 422.6, 422.88, 423.16, 4
```

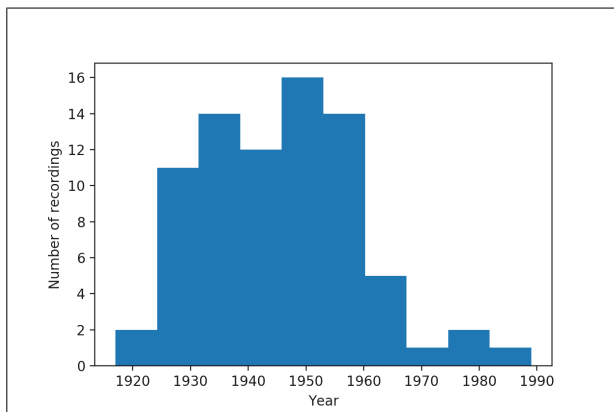


Figure 2. Distribution of recordings from the dataset by year.

If the chords played by the rhythm section are not clearly audible during a solo, chords played in the “head” are replicated. Useful guidelines on chord transcription in jazz are given in the introduction of Henry Martin’s book [26]. The annotators used existing resources as a starting point, such as published transcriptions of a particular performance or Real book chord progressions, but the final decisions for each recording were made by the annotator. We developed an automation tool for checking the annotation syntax and chord sonification: chord sounds are generated with Shepard tones and mixed with the original audio track, taking its volume into account. If annotation errors are found during syntax check or while listening to the sonification playback, they are corrected and the loop is repeated.

4. DATASET SUMMARY AND IMPLICATIONS FOR CORPUS BASED RESEARCH

To date, 113 tracks are annotated with an overall duration of almost 7 hours, or 68570 beats. Annotated recordings were made from music created between 1917 and 1989, with the greatest number coming from the formative years of jazz: the 1920s-1960s (see Figure 2). Styles vary from blues and ragtime to New Orleans, swing, be-bop and hard bop with a few examples of gypsy jazz, bossa nova, Afro-Cuban jazz, cool, and West Coast. Instrumentation varies from solo piano to jazz combos and to big bands.

4.1 Classifying chords in the jazz way

In total, 59 distinct chord classes appear in the annotations (89, if we count chord inversions). To manage such a diversity of chords, we suggest classifying chords as it done in jazz pedagogical and theoretical literature. According to the article by Strunk [32], chord inversions are not important in the analysis of jazz performance, perhaps because of the improvisational nature of bass lines. Inversions are used in lead sheets mainly to emphasize the composed bass line (e.g., pedal point or chromaticism). Therefore, we ignore inversions in our analysis.

According to numerous instructional books, and to theoretical work done by Martin [25], there are only five main

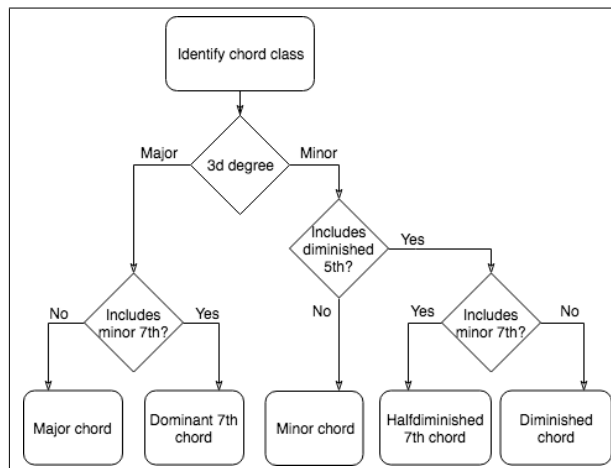


Figure 3. Flow chart: how to identify chord class by degree set.

Chord class	Beats Number	Beats %	Duration (seconds)	Duration %
dom7	29786	43.44	10557	42.23
maj	18591	27.11	6606	26.42
min	13172	19.21	4681	18.72
dim	1677	2.45	583	2.33
hdim7	1280	1.87	511	2.04
no chord	3986	5.81	2032	8.13
unclassified	78	0.11	30	0.12

Table 1. Chord classes distribution.

chord classes in jazz: major (maj), minor (min), dominant seventh (dom7), half-diminished seventh (hdim7), and diminished (dim). Seventh chords are more prevalent than triads, although sixth chords are popular in some styles (e.g., gypsy jazz). Third, fifth and seventh degrees are used to classify chords in a bit of an asymmetric manner: the unaltered fifth could be omitted in the major, minor and dominant seventh (see chapter on three note voicing in [23]); the diminished fifth is required in half-diminished and in diminished chords; and $\flat\flat 7$ is characteristic for diminished chords. We summarize this classification approach in the flow chart in Figure 3.

The frequencies of different chord classes in our corpus are presented in Table 1. The dominant seventh is the most popular chord, followed by major, minor, diminished and half-diminished. Chord popularity ranks differ from those calculated in [6] for the iRb corpus: dom7, min, maj, hdim, and dim. This could be explained by the fact that our dataset is shifted toward the earlier years of jazz development, when major keys were more pervasive.

4.2 Exploring symbolic data

Exploring the distribution of chord transition bigrams and n-grams allows us to find regularities in chord progressions. The term bigram for two-chord transitions was de-

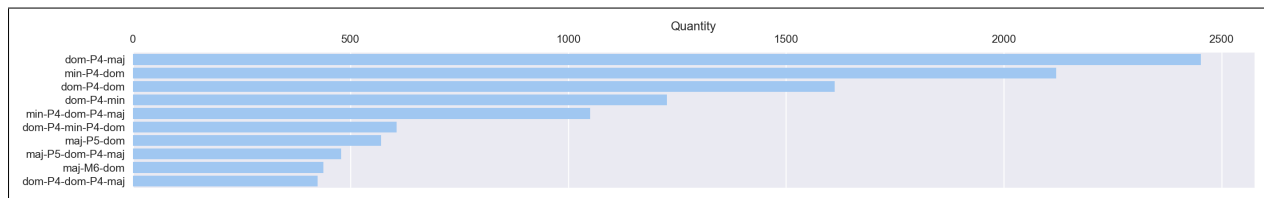


Figure 4. Top ten chord transition n-grams. Each n-gram is expressed as sequence of chord classes (dom, maj, min, hdim7, dim) alternated with intervals (e.g., P4 - perfect fourth, M6 - major sixth), separating adjacent chord roots.

finied in [6]. Similarly, we define an n-gram as a sequence of n chord transitions. The ten most frequent n-grams from our dataset are presented in Figure 4. The picture presented by the plot is what would be expected for a jazz corpus: we see the prevalence of the root movement by the cycle of fifths. The famous IIm-V7-I three-chord pattern (e.g., [25]) is ranked number 5, which is even higher than most of the shorter two-chord patterns.

5. CHORD TRANSCRIPTION ALGORITHMS BASELINE EVALUATION

Now we turn to Automatic Chord Estimation (ACE) evaluation for jazz. We adopt the MIREX¹⁶ approach to evaluating ACE algorithms. The approach supports multiple ways to match ground truth chord labels with predicted labels, by employing the different chord vocabularies introduced by Pauwels [29]. The distinctions between the five chord classes defined in 4.1 are crucial for analyzing jazz performance. More detailed transcriptions (e.g., a distinction between maj6 and maj7, detecting extensions of dom7, etc.) are also important but secondary to classification into the basic five classes. To formally implement this concept of chord classification, we develop a new vocabulary, called “Jazz5,” which converts chords into the five classes according to the flowchart in Figure 3.

For comparison, we also choose two existing MIREX vocabularies: “Sevenths” and “Tetrads,” because they ignore inversions and can distinguish between major, minor and dom7 classes (which together occupy about 90% of our dataset). However, these vocabularies penalize differences within a single basic class (e.g., between a major triad and a major seventh chord). Moreover, the “Sevenths” vocabulary is too basic; it excludes a significant number of chords, such as diminished chords or sixths, from evaluation.

We choose Chordino¹⁷, which has been a baseline algorithm for the MIREX challenge over several years, and CREMA¹⁸, which was recently introduced in [28]. To date, CREMA is one of the few open-source, state-of-the-art algorithms which supports seventh chords.

Results are provided in the Table 2. “Coverage” signifies the percentage of the dataset which can be evaluated using the given vocabulary. “Accuracy” stands for the per-

Vocabulary	Coverage %	Chordino Accuracy %	CREMA Accuracy %
“Jazz5”	99.88	32.68	40.26
MirexSevenths	86.12	24.57	37.54
Tetrads	99.90	23.10	34.30

Table 2. Comparison of coverage and accuracy evaluation for different chord dictionaries and algorithms.

centage of the covered dataset for which chords were properly predicted, according to the given vocabulary.

We see that the accuracy for the jazz dataset is almost half of the accuracy achieved by the most advanced algorithms on datasets currently involved in the MIREX challenge¹⁹ (which is roughly 70-80%). Nevertheless, the more recent algorithm (CREMA) performs significantly better than the old one (Chordino) which shows that our dataset passes a sanity check: it does not contradict technological progress in Automatic Chord Estimation. We see from this analysis that the “Sevenths” chords vocabulary is not appropriate for a jazz corpus because it ignores almost 14% of the data. We also note that the “Tetrads” vocabulary is too punitive: it penalizes up to 9% of predictions. However, this could potentially be tolerable in the context of jazz harmony analysis. We provide code for this evaluation in the project repository.

6. CONCLUSIONS AND FURTHER WORK

We have introduced a dataset of time-aligned jazz harmony transcriptions, which is useful for MIR research and corpus-based musicology. We have demonstrated how the particularities of the jazz genre affect our approach to data selection, annotation, and evaluation of chord estimation algorithms.

Further work includes growing the dataset by expanding the set of annotated tracks and adding new features. Functional harmony annotation (or local tonal centers) is of particular interest, because we could then implement chord detection accuracy evaluation based on jazz chord substitution rules.

¹⁶ http://www.music-ir.org/mirex/wiki/2017:Audio_Chord_Estimation

¹⁷ <http://www.isophonics.net/nnls-chroma>

¹⁸ <https://github.com/bmcftee/crema>

¹⁹ http://www.music-ir.org/mirex/wiki/2017:Audio_Chord_Estimation_Results

7. ACKNOWLEDGMENT

The authors would like to thank all the anonymous reviewers for their valuable comments, which greatly helped to improve the quality of this paper.

8. REFERENCES

- [1] The Smithsonian Collection of Classic Jazz. Smithsonian Folkways Recording, 1997.
- [2] Jazz: The Smithsonian Anthology. Smithsonian Folkways Recording, 2010.
- [3] Stefan Balke, Christian Dittmar, Jakob Abeßer, Klaus Frieler, Martin Pfeiderer, and Meinard Müller. Bridging the Gap: Enriching YouTube Videos with Jazz Music Annotations. *Frontiers in Digital Humanities*, 5, 2018.
- [4] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proc. of the 24th ACM International Conference on Multimedia*, pages 1174–1178, 2016.
- [5] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, O. Mayor, Gerard Roma, Justin Salamon, J. R. Zapata, and Xavier Serra. Essentia: an audio analysis library for music information retrieval. In *Proc. of the of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 493–498, 2013.
- [6] Yuri Broze and Daniel Shanahan. Diachronic Changes in Jazz Harmony: A Cognitive Perspective. *Music Perception: An Interdisciplinary Journal*, 3(1):32–45, 2013.
- [7] John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An Expert Ground-Truth Set for Audio Chord Recognition and Music Analysis. In *Proc. of the of the International Society for Music Information Retrieval Conference (ISMIR)*, number ISMIR, pages 633–638, 2011.
- [8] Chris Cannam, Christian Landone, Mark Sandler, and Juan Pablo Bello. The Sonic Visualiser: A Visualisation Platform for Semantic Descriptors from Musical Signals. In *Proc. of the of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 324–327, 2006.
- [9] Trevor de Clercq and David Temperley. A corpus analysis of rock harmony. *Popular Music*, 30(01):47–70, jan 2011.
- [10] Ken Déguernel, Emmanuel Vincent, and Gérard Asayag. Using Multidimensional Sequences For Improvisation In The OMax Paradigm. In *13th Sound and Music Computing Conference*, 2016.
- [11] Junqi Deng and Yu-kwong Kwok. A hybrid gaussian-hmm-deep-learning approach for automatic chord estimation with very large vocabulary. In *Proc. 17th International Society for Music Information Retrieval Conference*, pages 812–818, 2016.
- [12] Bruno Di Giorgi, Massimiliano Zanoni, Augusto Sarti, and Stefano Tubaro. Automatic chord recognition based on the probabilistic modeling of diatonic modal harmony. In *Proc. of the 8th International Workshop on Multidimensional Systems (nDS)*, pages 1–6, 2013.
- [13] Hubert Léveillé Gauvin. "The Times They Were A-Changin'": A Database-Driven Approach to the Evolution of Harmonic Syntax in Popular Music from the 1960s. *Empirical Musicology Review*, 10(3):215–238, apr 2015.
- [14] Ted Gioia. *The Jazz Standards: A Guide to the Repertoire*. Oxford University Press, 2012.
- [15] Mark Granroth-Wilding. *Harmonic analysis of music using combinatorial categorical grammar*. PhD thesis, University of Edinburgh, 2013.
- [16] C Harte, M Sandler, S Abdallah, and E Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proc. of the of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 56, pages 66–71, 2005.
- [17] Christopher Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, Queen Mary, University of London, 2010.
- [18] Thomas Hedges, Pierre Roy, and François Pachet. Predicting the Composer and Style of Jazz Chord Progressions. *Journal of New Music Research*, 43(3):276–290, jul 2014.
- [19] Eric J Humphrey and Juan P Bello. Four timely insights on automatic chord estimation. In *Proc. of the of the International Society for Music Information Retrieval Conference (ISMIR)*, 2015.
- [20] Eric J Humphrey, Justin Salamon, Oriol Nieto, Jon Forsyth, Rachel M Bittner, and Juan P Bello. Jams: a Json Annotated Music Specification for Reproducible Mir Research. In *Proc. of the International Society for Music Information Retrieval (ISMIR)*, 2014.
- [21] Barry Dean Kernfeld. *The story of fake books : bootlegging songs to musicians*. Scarecrow Press, 2006.
- [22] Hendrik Vincent Koops, Bas de Haas, John Ashley Burgoyne, Jeroen Bransen, and Anja Volk. Harmonic subjectivity in popular music. Technical Report UUCS-2017-018, Department of Information and Computing Sciences, Utrecht University, 2017.
- [23] Mark Levine. *The Jazz Piano Book*. Sher Music, 1989.
- [24] Mark Levine. *The Jazz Theory Book*. Sher Music, 2011.

- [25] Henry Martin. Jazz Harmony: A Syntactic Background. *Annual Review of Jazz Studies*, 8:9–30, 1988.
- [26] Henry Martin. *Charlie Parker and Thematic Improvisation*. Institute of Jazz Studies, Rutgers–The State University of New Jersey, 1996.
- [27] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *Proc. of the of the International Society for Music Information Retrieval Conference (ISMIR)*, number 1, pages 135–140, 2010.
- [28] Brian Mcfee and Juan Pablo Bello. Structured Training for Large-Vocabulary Chord Recognition. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 188–194, 2017.
- [29] Johan Pauwels and Geoffroy Peeters. Evaluating automatically estimated chord sequences. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 749–753. IEEE, 2013.
- [30] Martin Pfeleiderer, Klaus Frieler, and Jakob Abeßer. *Inside the Jazzomat - New perspectives for jazz research*. Schott Campus, Mainz, Germany, 2017.
- [31] Keith Salley and Daniel T. Shanahan. Phrase Rhythm in Standard Jazz Repertoire: A Taxonomy and Corpus Study. *Journal of Jazz Studies*, 11(1):1, nov 2016.
- [32] Steven Strunk. Harmony (i). The New Grove Dictionary of Jazz, pp. 485–496, 1994.

METHODOLOGIES FOR CREATING SYMBOLIC CORPORA OF WESTERN MUSIC BEFORE 1600

Julie E. Cumming

McGill University
julie.cumming
@mcgill.ca

Cory McKay

Marianopolis College
cory.mckay
@mail.mcgill.ca

Jonathan Stuchbery

McGill University
jonathan.stuchbery
@mail.mcgill.ca

Ichiro Fujinaga

McGill University
ichiro.fujinaga
@mcgill.ca

ABSTRACT

The creation of a corpus of compositions in symbolic formats is an essential step for any project in systematic research. There are, however, many potential pitfalls, especially in early music, where scores are edited in different ways: variables include clefs, note values, types of barline, and editorial accidentals. Different score editors and optical music recognition software have their own ways of storing and exporting musical data. Choice of software and file formats, and their various parameters, can thus unintentionally bias data, as can decisions on how to interpret potentially ambiguous markings in original sources. This becomes especially problematic when data from different corpora are combined for computational processing, since observed regularities and irregularities may in fact be linked with inconsistent corpus collection methodologies, internal and external, rather than the underlying music.

This paper proposes guidelines, templates, and workflows for the creation of consistent early music corpora, and for detecting encoding biases in existing corpora. We have assembled a corpus of Renaissance duos as a sample implementation, and present machine learning experiments demonstrating how inconsistent or naïve encoding methodologies for corpus collection can distort results.

1. INTRODUCTION

Because creating accurate corpora is extremely labour intensive, early music researchers often draw on symbolic scores already available online. These collections, however, exhibit many different approaches to encoding scores, depending on the choices of the individual who did each encoding, the music editor used, the particular symbolic music file formats used, and the ways in which those files were generated. Even when transcribing music directly into a music editor, it is important to have clear guidelines for many elements of the transcription. A good corpus, therefore, requires a clear set of guidelines and templates for notation and file creation. It also requires a workflow that integrates correction, and consistent pro-

cesses for generating symbolic files. We describe an effective process for encoding a consistent corpus for research projects on Renaissance music, and use it to create a publicly-available collection of duos. We end with an experiment involving this dataset showing how different or inconsistent encoding methodologies can distort results.

1.1. Related Work

Several collections of symbolic Renaissance scores exist. The Choral Public Domain Library (CPDL) [4] includes large amounts of Renaissance music, but there is no attempt at standardization. The original ELVIS database [5] also aimed for quantity without much curation, but with substantial metadata. The Josquin Research Project (JRP) [21] is carefully curated and extremely consistent. Smaller collections assembled for specific projects, such as [8], [12], [13], [19], [20], and [22], are carefully curated, but each uses a different approach.

2. RESEARCH CORPORA IN RENAISSANCE MUSIC: NOTATIONAL CONSISTENCY

In Renaissance music manuscripts and prints the parts are not aligned in score. Instead they are presented in separate parts (on different parts of the page or in separate partbooks). In order to study this music the parts must be transcribed and combined into a score. Mensuration signs (similar to time signatures) indicate the metrical organization, but the parts have no barlines, and ties are never used. There are multiple different clefs (C clefs on any line; F clefs on three lines; G clef is rare). Performers are expected to add accidentals in specific melodic and contrapuntal situations without explicit accidentals in the score (resulting in debates among performers and editors of early music). Note values are larger than those of common Western notation: between 1450 and 1550 the beat normally falls on the semibreve (whole note).

Modern editors have a wide variety of approaches to transcription, as described in [3] and [14]. Some try to make the edition look like 18th-century music, while others try to preserve elements of the original notation, and everything in between. There are editions of Renaissance music scores in original clefs and modern clefs; with barlines, without barlines, or with *mensurstriche* (barlines that only appear between the staves). We can find scores with original, halved, quartered, and smaller note values.



© Julie E. Cumming, Cory McKay, Jonathan Stuchbery, Ichiro Fujinaga. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Julie E. Cumming, Cory McKay, Jonathan Stuchbery, Ichiro Fujinaga. "Methodologies for Creating Symbolic Corpora of Western Music before 1600", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

Most editors introduce editorial accidentals, but there are multiple possibilities, and few agree on every decision. Editors often also transpose works (for performance by a specific ensemble, or because they believe that the original pitch was higher or lower than the “written” pitch in the original source). The same piece of music edited by different people will look very different (see Figure 1). Transcribing works directly from the original sources is extremely time consuming, however, so if a piece is available in modern transcription, we normally start with that, either by transcribing it or by using an OMR program such as PhotoScore, and then correct it manually.



Figure 1. Contrasting editions of Josquin Desprez, *Missa de beata virgine*, Agnus II, mm. 1–7. Top: original note values with *mensuralstriche* from [10]. Middle: halved note values with bar lines from [9]. Bottom: our edition, with original note values, bar lines, and time signature that matches the measure length.

2.1. Problems Resulting from Inconsistent Notation

When converting published scores into a symbolic corpus for music research (through OMR or transcription with a music editor), or when taking symbolic scores from an online repository, it is essential to make the notation of the scores consistent. Inconsistent notation can cause significant errors in computational analysis, as we show in the experiment described in Section 6 below. For example, when analysing counterpoint we normally sample the score at every minim (half note) in the original notation. If we have one score in original note values, and one in quartered note values, the half note will have a completely different meaning, and the results will not be comparable. The length of a work can also provide information on genre. If the measures are different lengths, because of different editorial decisions, then this data will be incorrect. When looking at issues of mode we normally check *final* and key signature; if a work is transposed, this will distort the data. If the number of beats in a measure does not match the time signature, software such as music21 [7] will not parse the symbolic score correctly.

2.2. Creating and Obtaining Symbolic Scores

The most straightforward way to create a symbolic file is to transcribe the piece into a music editor from images of the original source (Renaissance manuscript or print). While this is time consuming, especially if the original source is difficult to read, or if there are ambiguities in the notation, it results in a file that is very close to the original source.

All the other methods involve working with a modern edition: transcription into a music editor from a modern edition (we do this when the notation of the edition is not suitable for OMR); obtaining symbolic files from online repositories, including the CPDL [4] and the JRP [21]; or using an OMR program such as PhotoScore on a modern edition. Almost all of these files need adjustment with regard to note values, time signatures, editorial accidentals, and pitch level. As we constructed our corpus, we kept finding additional issues that required decisions, which we incorporated into guidelines and templates.

2.3. Our Guidelines for Consistency in Scores of Renaissance Music c. 1450–1550

In order to establish norms it is useful to decide on one source of authority, and to create a clear set of guidelines, as well as a template encapsulating the guidelines. We chose not to follow the standards of a single modern edition. Instead, we stayed as close to the original as possible, given that we are transcribing the pieces into modern notation in score, with bar lines. This means that we use the original notated pitch of the work, original note values, and we do not include editorial accidentals, since these are often a subjective decision of a particular editor and there is rarely complete consensus among experts. For ease of reading we use modern clefs: treble clef, transposing treble clef, and bass clef (see Figure 1). We use time signatures and ties; most of our time signatures use the whole note as the beat (2/1 or 3/1). There are no time-signature change unless there is a real change of meter in the piece, and the time signature must match the length of the measure. The traditional final long is transcribed as two breves, tied over the bar. We only include fermatas found in the original source, and use a fermata symbol that does not affect the rhythmic value of the note. In general, correct and consistent encoding is considered more important than the appearance of the score, and more important than graphic features of the modern edition or the original notation, such as ligature brackets, ranges, and original clefs and note shapes.

3. ENCODING EARLY MUSIC

Once researchers have established notational norms for the corpus, they must also establish norms for encoding. When using pieces available on line, or when more than one person is creating symbolic files for the corpus, there are many possible sources of inconsistency: symbolic files in different formats; the use of different music notation software to generate files; different software versions; and different encoding settings for a given piece of

software. We created a set of basic principles to address such problems, and incorporated them into our workflow and score editor templates.

3.1. Encoding Formats

We generate Sibelius, MIDI, Music XML, **kern, and PDF files for use in several different machine learning and music analysis contexts. Although it is arguably desirable to use purely open file formats when possible (e.g., for long-term compatibility), the ubiquity of a format is also an essential consideration, in order to maximize accessibility. We argue that presenting files in a variety of formats, open and closed, allows us to find a good compromise between these two concerns.

Much of the detail about encoding described here is focused on MIDI, which is important because of its ubiquity and because it requires that certain data be specified rather than left ambiguous (e.g., the tempo of a piece cannot be left undefined, as this will implicitly result in the default MIDI tempo being used). Although there are often good musicological reasons for ambiguity, it can cause serious problems for many systematic analysis, search, display, or feature extraction systems, which may use improper defaults or not work at all when faced with certain kinds of ambiguous data. From the specific perspective of computational music processing, MIDI helpfully forces encoders to specify best estimates in cases where there is ambiguity. The most important reason for choosing MIDI, however, is simply that it can be both parsed and produced by almost any software, and follows a universally accepted and open standard. That being said, MIDI has many well-publicized imperfections and limitations, so it is always advisable to distribute datasets in other formats, as we do.

3.2. Basic Principles for Encoding the Corpus

- Use the same software, software version, operating system, and encoding settings
- Use a uniform and short file naming convention, and only allow ASCII characters, as archiving or moving files between computers or network locations can cause problems with long file names or non-ASCII characters
- Encode provenance information directly in the files themselves, in case encapsulating databases, etc. are lost; use rich character sets when permitted
- Be consistent with:
 - Instrument names (e.g., “alto” singer vs. “alto” viola); be sure there are no missing instrument names that default to incorrect instruments
 - Dynamics
 - Tempo
 - Time signatures and meter changes
 - Key signatures
 - Voice segregation
 - Transposing treble clefs
 - Fermatas

- Playback settings, affecting dynamics, varying tempo, note durations, etc. (disable rubato, swing, and “human playback” settings so that encodings are as rhythmically quantized as possible)
- For MIDI in particular:
 - Use MIDI Type 1
 - Conform to General MIDI instruments
 - Avoid keyboard instruments for non-keyboard parts, as keyboard encodings can sometimes cause individual voices in a polyphonic work to be collapsed into one part
 - Standardize to 960 PPQN (Pulse Per Quarter Note)
 - Set tempo to whole note = 80 BPM (quarter note = 320)
- Avoid:
 - Encoding methodologies that needlessly throw away information
 - Encoding methodologies that permit ambiguity (e.g., in note durations) in cases where automated feature extraction or analysis will be used
 - Format conversions: if they are necessary (e.g., in order to increase accessibility), generate all alternative encodings from a single master file

We dealt with consistency issues by building templates (blank pieces in the notation software with all the correct settings), into which we copied our pieces. These templates are available at [6].

3.3. Choice of Score Editing Software

We chose to use the latest version of Sibelius for compatibility and consistency reasons. It is one of the most widely used score editors, it works well with the PhotoScore OMR software, and it has a scripting language (ManuScript). It is also the only score editor that can be used to create MEI files, using the Sibelius MEI plugin [15]. Although there are certainly important advantages to using open-source software (e.g., MuseScore) when possible, there are no open-source alternatives to Sibelius that offer these essential advantages. That being said, Sibelius did initially cause us problems: the transposing clef often did not encode the voice in the lower octave, even though the “8” below the clef showed in the score. This distorts contrapuntal analysis (e.g., consonant fifths between voices turn into dissonant fourths).

4. WORKFLOW

In the process of developing our corpus we developed a workflow for file creation, including both manual and scripted processes that allowed us to avoid inconsistent file production. This workflow can be used by other researchers who want to create consistent corpora, and is available in more detail at [6]. It can be summarized briefly as follows:

- Create or collect symbolic files
- Copy the corrected symbolic files into the template

- Correct the files in Sibelius, following the guidelines in Section 2.3
- Check the files for problems (by looking at the PDFs, and comparing the files to original sources), and correct them manually when necessary
- Save the verified result as a “master file”
- Once all the desired master files for the corpus are assembled, generate all files in all alternative formats at the same time using a script
- Check MIDI files for consistency using jSymbolic [18] (which reveals inconsistent settings, including meter changes, dynamics, and tempo settings)

5. THE JOSQUIN / LA RUE DUOS CORPUS

We used the workflow and templates introduced above to create a corpus devoted to studying differences in the music of two leading Renaissance composers, Josquin Desprez (c. 1450–1521) and Pierre de la Rue (c. 1452–1518). These two composers are particularly interesting because it is difficult to tell their music apart, even for experts. They are almost exact contemporaries, and there are ten compositions attributed to both composers in different 16th-century sources. Past attempts to describe differences in style are often frustratingly vague, as in this discussion of why a La Rue Mass is not by Josquin: “the rhythmic motion and continuous repetition of the main melodic motif in mm. 45–66 lack the vitality characteristic of Josquin” [11].

Our corpus consists of duos (two-voice sections) from Masses by these two composers. It is important to compare works in the same genre, since different genres can result in different styles, even for the same composer. Also, composers and improvisers in the Renaissance began by learning to work in two voices; this is the purest form of Renaissance counterpoint. For this study we included only duos from Masses securely attributed to the composers (i.e., there is consensus that the Masses are not by another composer). For Josquin, we used the “secure” categories established by Jesse Rodin in the JRP [21]; for La Rue we used the assessments in the La Rue edition [17].

Most of the symbolic files in the corpus came from the JRP [21]. We searched the Masses for duo sections surrounded by double bars (separate sections of longer Mass movements). We downloaded the Music XML files for the relevant movements, opened them in Sibelius, and extracted the duos. Some additional movements were transcribed from the La Rue edition, restoring the original note values.

Our final corpus, titled the JLSDD (Josquin La Rue Secure Duos Dataset), after systematic cleaning, correction, and format translation, consists of 33 secure Josquin duos and 44 secure La Rue duos, each available as Sibelius, Music XML, MIDI, MEI, **kern, and PDF files at [6]. They are distributed with pre-extracted jSymbolic [10] features, and the Sibelius templates used to build the corpus may also be downloaded from [6].

6. EXPERIMENTS: JOSQUIN VS. LA RUE

We performed a series of machine learning-based composer attribution experiments in order to gain empirical insight into the effects of different encoding methodologies. For related studies on systematic composer classification, see [1], [2], and [16].

6.1. Datasets Used

All of the experiments described here made use of the 33 secure Josquin duos and 44 secure La Rue duos introduced in Section 5. We generated three different experimental MIDI datasets from this corpus:

- *Original*: All 77 secure Josquin and La Rue duos, generated from the Sibelius files, as they existed before systematic standards were used to correct, annotate, and encode them. These duos used a variety of General MIDI instrument patches, varying amounts of rubato added by Sibelius, varying amounts of dynamic variation added by Sibelius and inconsistent approaches to metrical annotation (e.g., time signatures of 4/4 and 8/4 vs. 2/1). Notably, these differences were distributed across the music of both composers, and were not meaningfully correlated with either of them.
- *Clean*: All 77 secure Josquin and La Rue duos, generated from the Sibelius files after systematic standardization had been applied. The files were all encoded using General MIDI Patch 53 (voice), all had a tempo of 80 whole-note beats per minute, all had time signatures based on whole-note beats and none had added rubato or dynamics. These are, in effect, the clean release version of the duos corpus described in Section 5.
- *Simulated*: The 33 secure Josquin duos, generated from the Original Sibelius files using systematic settings that differed from the settings used when generating the Clean dataset. This was done in order to allow us to simulate the effects of combining datasets acquired from different sources, where different encoding standards were used. In this case, all files were encoded using General MIDI 1 (piano), a tempo of 120 whole-note beats per minute, no rubato added, and no dynamics added. The choice of a piano patch had the additional effect of causing Sibelius to encode the notes from both voices into a single MIDI channel and track, thereby losing the explicit voice segregation found in the Original and Clean datasets.

6.2. Feature Extraction

Features were extracted from each of the Original, Clean, and Simulated datasets using the newest version (2.2) of the open-source jSymbolic software [18]. jSymbolic extracts 246 unique features from symbolic music files, including a number of multidimensional features, for a total of 1497 values. These features can be loosely grouped into the following categories: pitch statistics; melodic features; chords and vertical intervals; rhythm; instrumentation; texture; and dynamics. jSymbolic was chosen be-

cause it includes far more features than any other musical symbolic feature extraction software, and its extensive documentation and relatively easy-to-use interface make it particularly accessible to musicological researchers who may have less experience with MIR software.

Two sets of features were extracted for each experiment:

- *All Features*: All features implemented by jSymbolic that can be extracted from MIDI files.
- *Safe Features*: A subset of the All Features group that consists of just 173 of jSymbolic's 246 implemented features. These features omit all features associated with tempo, dynamics, instrumentation, and meter, among other things. The intention of these features is that they can be used even when datasets are in fact systematically biased based on encoding methodology (since the features that would be sensitive to these biases are not extracted). All features known to be associated with these qualities were left out, and then a further feature / class correlation check (see below) was performed in order to make sure no bias-sensitive features remained. The Safe Features are a good fit with Renaissance music, in which tempo, dynamics, and instrumentation are not indicated in the musical sources, and are left to the discretion of the performers.

We further analyzed the Clean and Original datasets by calculating the Pearson correlation coefficient between each feature in each dataset we experimented with and the composer class (Josquin or La Rue). For all features with high correlations, we manually checked to see whether the strong correlation was due to an actual meaningful musical difference or to bias introduced by the encoding methodology. For example, all the Clean pieces had a tempo of 80 BPM, and all the Simulated pieces had a tempo of 120 BPM. Thus the tempo feature alone was perfectly correlated to the class when the Simulated Josquin pieces were compared to the Clean La Rue pieces, and thus tempo even by itself perfectly distinguished Josquin from La Rue. Of course, this is in fact due to the arbitrarily chosen tempos assigned when encoding each of these two datasets, so the perfect classification performance of tempo in this example is clearly due solely to an encoding methodology inappropriately correlated to class.

6.3. Machine Learning Methodology

The features extracted from the Original, Clean, and Simulated datasets were used in several supervised 10-fold cross-validation experiments performed using the open-source Weka machine learning software [24]. In particular, Weka's SMO support vector machine implementation was used with default hyper-parameter settings. This particular configuration was chosen because it is a relatively quick-and-easy approach to use, while still being quite effective, and thus simulates what musicolog-

ical researchers with only casual expertise in machine learning might do relatively easily.

6.4. Experimental Results and Analysis

Table 1 shows the classification accuracies for each dataset, averaged across cross-validation folds. In some cases, the pieces compared for each of the two composers come from the same dataset (Original, Clean, and Simulated), in order to explore the internal effectiveness of the encoding methodology used in that dataset. In other cases, the music for one composer was drawn from a different dataset than the music for the other composer, in order to simulate what one might encounter if one were to perform experiments using music that had been encoded using different methodologies.

We can see in Row 1 that the SMO algorithm was able to use the jSymbolic features to correctly distinguish between the Josquin and La Rue duos 87.0% of the time when the Clean dataset was used. This is quite impressive, given how similar the two composers are, and we can be confident that this result is not inflated by encoding bias (because of the systematically consistent way that the Clean data was encoded, and because the features were manually examined to provide additional assurance that no unanticipated bias slipped through).

In Rows 1 and 2 we can see that the Clean data performed 2.6% better than the Original data (87.0% vs. 84.4%). We can be confident that neither of these results are artificially inflated by encoding methods correlated with composers, as manual verification to guard against this was performed here as well. There are, notably, some important differences in how different pieces were encoded in the Original data; these differences are just not correlated with the composer. So, rather than causing classification to improve artificially, these encoding differences could instead deflate classification performance by injecting noise into the features. However, it should be noted that the difference in performance between Rows 1 and 2 is not large enough to be statistically significant (with a p-value of 0.05).

In Rows 4, 5, 9, and 10 we can see that classification results were grossly inflated to 100% when the Simulated data for Josquin was mixed with either the Clean or Original data for La Rue. This is because there were elements associated with instrumentation, tempo, meter, and dynamics that were strongly based on the encoding methods used rather than the underlying music, and these encodings were correlated with the composers. This confirms that, if one is not careful to avoid bias when encoding data, then one can achieve results that seem impressive but are in fact meaningless.

We can see that the Clean / Clean and Original / Original results are quite the same for the All Features (Rows 1 and 2) and Safe Features (Rows 6 and 7) groups. This makes sense, since the Safe Features omit all features that could be biased by the encoding differences in the Clean and Original groups, and the Clean group has no internal

bias based on encoding source, while the Original group has no correlation between the different encoding methodologies used and the particular composers.

Row	Feature Set	Josquin Dataset	La Rue Dataset	CA (%)
1	All	Clean	Clean	87.0
2	All	Original	Original	84.4
3	All	Clean	Original	98.7
4	All	Simulated	Clean	100.0
5	All	Simulated	Original	100.0
6	Safe	Clean	Clean	87.0
7	Safe	Original	Original	84.4
8	Safe	Clean	Original	87.0
9	Safe	Simulated	Clean	100.0
10	Safe	Simulated	Original	100.0

Table 1. Classification accuracies (CA) averaged across 10 folds for each of the 2-class composer attribution experiments. Each experiment is performed once with all 246 unique features (“All Features”) and once with a reduced set of 173 features chosen to be less vulnerable to encoding bias (“Safe Features”). All experiments include the same 33 secure Josquin duos and 44 secure La Rue duos, but the encodings for each vary (“Original,” “Clean,” or “Simulated”).

There is a difference, however, between the All Features and Safe Features performance for the Clean Josquin vs. Original La Rue experiments: the 98.7% achieved by the All Features group (Row 3) was clearly inflated, but the 87% achieved by the Safe Features group (Row 8) was not (in fact, it was identical to the best real results found in the Clean Josquin vs. Clean La Rue experiment). This is because Clean Josquin vs. Original La Rue does include some differences in tempo, meter, instrumentation, rubato and dynamics that are correlated with composer in this case (Clean Josquin is uniform in these parameters, but Original La Rue is not). The All Features set is sensitive to these differences, and thus produces inflated results, but the Safe Features set filters out these problems by ignoring the composer-correlated biased quantities.

It is also notable that both the Simulated Josquin vs. Clean La Rue (Row 9) and Simulated Josquin vs. Original La Rue (Row 10) results were clearly inflated (both 100%), even for the Safe Features. This is because the Simulated encoding compressed the two distinct voices in each duo into a single voice (as a side effect of using a piano patch rather than a voice patch); although no notes were lost in this process, many features that rely on voice segregation were affected. The Safe Features did not omit such voice-linked features, so they were affected by the encoding bias. This serves as a good reminder that even “safe” features may not always be as safe as one thinks, and that cleanly and consistently encoded data is always better when available.

Of course, a reduced set of “safe” features can still be useful when one has no choice but to use data from different sources that have used different encoding methodologies. We could, for example, have made an “Extra Safe Features” group that also avoided features linked to voice segregation. The problem with being too cautious in this way, however, is that one risks omitting features that do in fact reveal musically meaningful insights. For example, examination of the feature values shows that Josquin and La Rue used voice crossing to different extents, so features related to voice crossing distinguish the two composers meaningfully; if one omits all voice-related features out of fear of biased results, then such insights will never be revealed. “Safe” feature sets must always strike a balance between security against encoding bias on the one hand and openness to musically meaningful information on the other.

6.5. Summary of Experimental Results

Using consistently and systematically encoded music can potentially play an essential role in:

- Avoiding inflated performances due to encoding biases correlated with class
- Avoiding deflated performance due to feature noise not correlated with class

Using “safe” features chosen to minimize sensitivity to encoding bias is a viable approach if one has no choice but to use data encoded in different ways, but it is inferior to using uniformly encoded data because:

- Overly cautious safe features may eliminate features that would reveal musically meaningful insights
- Insufficiently cautious safe features may admit unanticipated biases into the feature values if one does not perform careful checks to avoid this

7. CONCLUSIONS

We have established that notational consistency and encoding consistency are essential to reliable computer-aided research on Renaissance music. Our experience assembling corpora with a small team of people (including undergraduates, graduate students, post-docs, and professors) showed that establishing clear guidelines and creating templates enabled us to reach the desired level of consistency; that consistency then allows us to conduct compelling research. Our corpus, templates and workflow are available online at [6]. If other scholars adopt the same conventions for their corpora, large and small, and make them available, we will be on the path to large-scale research into Renaissance music; a composite corpus that is both varied and consistent.

8. ACKNOWLEDGEMENTS

This work was supported by the Social Sciences and Humanities Research Council of Canada and the Fonds de Recherche du Québec - Société et Culture. We would also like to thank Laura Beauchamp, Nathaniel Condit-Schultz, Néstor Nápoles López, and Ian Lorenz for their help with multiple aspects of this paper.

9. REFERENCES

- [1] M. W. Beauvois, “A Statistical Analysis of the Chansons of Arnold and Hugo de Lantins,” *Early Music*, Vol. 45, No. 4, pp. 527–543, 2017, <https://doi.org/10.1093/em/cax108>. [Accessed: Jun. 9, 2018].
- [2] A. Brinkman, D. Shanahan and C. Sapp, “Musical Stylometry, Machine Learning and Attribution Studies: A Semi-Supervised Approach to the Works of Josquin,” *Proc. of the Biennial Int. Conf. on Music Perception and Cognition*, pp. 91–97, 2016.
- [3] J. Caldwell, *Editing Early Music*, Clarendon Press, Oxford, 1995.
- [4] *Choral Public Domain Library*, 2018. [Online]. Available: <http://www.cpdl.org/wiki/>. [Accessed: Jun. 7, 2018].
- [5] J. E. Cumming et al. *ELVIS Database*, 2016. [Online]. Available: <https://database.elvisproject.ca/>. [Accessed: Jun. 7, 2018].
- [6] J. E. Cumming, C. McKay, J. Stuchbery, and I. Fujinaga, JLSDD (Josquin La Rue Secure Duos Dataset) *GitHub.com*, 2018. [Online]. Available: <https://github.com/ELVIS-Project/mass-duos-corpus-josquin-larue/tree/Methodologies-for-Creating-Symbolic-Music-Corpora>. [Accessed: Jun. 7, 2018].
- [7] M. S. Cuthbert and C. Ariza, “music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data,” *Proc. of ISMIR*, pp. 637–642, Utrecht, Netherlands, 2010. [Online.] <http://web.mit.edu/music21/>. [Accessed: Jun. 12, 2018.]
- [8] K. Desmond et al., *Measuring Polyphony: Digital Encodings of Late Medieval Music*, 2018. [Online]. Available: <http://measuringpolyphony.org/>. [Accessed: Jun. 7, 2018].
- [9] J. Desprez, “27 Duos by Josquin Desprez or not, edited and adapted for instruments, especially recorders and keyboard instruments or harp,” A. den Teuling, Ed. Assen, NL, 2014, p. 11. *IMSLP/Petrucchi Music Library: Free Public Domain Sheet Music*. [Online]. Available: [http://imslp.org/wiki/27_Duos_\(Josquin_Desprez\)](http://imslp.org/wiki/27_Duos_(Josquin_Desprez)). [Accessed: Jun. 7, 2019].
- [10] J. Desprez, *Missa de Beata Virgine: zu 4 und 5 Stimmen*, 2. Aufl., ed. Friedrich Blume, Das Chorwerk, Heft 42. Wolfenbüttel: Mösel Verlag, 1951, p. 48. [Online]. Available: http://ks.petrucchimusiclibrary.org/files/imglnks/usimg/5/56/IMSLP48537-PMLP102712-Das_Chorwerk_042_-_Desprez,_Josquin_-_Missa_De_Beata_Virgine.pdf. [Accessed: Jun. 7, 2018].
- [11] W. Elders, *New Josquin Edition*, vol. 4, *Masses based on Gregorian chants 2: Critical Commentary*, p. 102, Koninklijke Vereniging voor Nederlandse Muziekgeschiedenis, Amsterdam, 2000.
- [12] R. Freedman, D. Fiala, R. Vigiante, and V. Besson. *Citations: The Renaissance Imitation Mass (CRIM)*. [Online]. Available: <https://sites.google.com/haverford.edu/crim-project/home>; <https://www.dropbox.com/sh/lyka868ojkgz12/AADa3dYzGTfqB8YMU48jbIuUa?dl=0>; <http://159.65.177.99:8000/pieces/>. [Accessed: Jun. 7, 2018].
- [13] R. Freedman and P. Vendrix, *The Lost Voices Project*, 2014. [Online]. Available: <http://digitalduchemin.org/>; <https://www.dropbox.com/sh/f2z4iyks2fk9y1a/AAD5qJXwlYQdVC-kuPgBv3Mha?dl=0>; <http://digitalduchemin.org/mei/DC0407.xml>. [Accessed: Jun. 7, 2018].
- [14] J. Grier, *The Critical Editing of Music: History, Method, and Practice*, Cambridge Univ. Press, 1996.
- [15] A. Hankinson, “Sibelius MEI Plugin,” *GitHub.com*, 2017. [Online]. Available: <https://github.com/music-encoding/sibmei>. [Accessed: Jun. 7, 2018].
- [16] D. Herremans, D. Martens, and K. Sörensen, “Composer Classification Models for Music-Theory Building,” in *Computational Music Analysis*, D. Meredith, Ed. Cham: Springer International Publishing, 2016, pp. 369–392. [Online]. Available: https://www.researchgate.net/profile/Dorien_Herremans/publication/283321533_Composer_Classification_Models_for_Music-Theory_Building/links/5633449c08ae242468db84a9/Composer-Classification-Models-for-Music-Theory-Building.pdf. [Accessed: Jun. 7, 2018].
- [17] P. de La Rue, *Opera Omnia*, vol. 7, *Mass Dubia*, ed. N. Davison, J. E. Kreider, T. H. Keahey, American Institute of Musicology, Neuhausen, 1998.
- [18] C. McKay et al., “jSymbolic 2.2: Extracting features from symbolic music for use in musicological and MIR research,” *Proc. of the Int. Soc. For Music Information Retrieval Conf.*, accepted for publication, 2018.

- [19] E. Parada-Cabaleiro, A. Batliner, A. Baird, and B. W. Schuller, "The SEILS Dataset: Symbolically Encoded Scores in Modern-Early Notation for Computational Musicology," *Proc. of the 18th ISMIR*, pp. 575-481, Souzhou, China, 2017. "The SEILS Dataset," *GitHub.com*, 2017. [Online]. Available: <https://github.com/SEILSdataset/SEILSdataset>. [Accessed: Jun. 7, 2018].
- [20] E. Ricciardi and C. S. Sapp, *Tasso in Music Project*. [Online]. Available: <http://www.tassomusic.org/>. [Accessed: Jun. 7, 2018].
- [21] J. Rodin and C. S. Sapp, *Josquin Research Project*. [Online]. Available: <http://josquin.stanford.edu/>; <http://josquin.stanford.edu/about/attribution/>. [Accessed: Jun. 7, 2018].
- [22] P. Vendrix et al. *Gesualdo Online*. [Online]. Available: <https://ricercar.gesualdo-online.cesr.univ-tours.fr/>. [Accessed: Jun. 7, 2018].
- [23] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufman, New York, 2011.

PRECISION OF SUNG NOTES IN CARNATIC MUSIC

Venkata Subramanian Viraraghavan^{1,2}

Hema A Murthy³

R Aravind²

¹ TCS Research and Innovation, Embedded Systems and Robotics, Bangalore, India

² Department of Electrical Engineering, Indian Institute of Technology, Madras

³ Department of Computer Science and Engineering, Indian Institute of Technology, Madras

venkatasubramanian.v@tcs.com, hema@cse.iitm.ac.in, aravind@ee.iitm.ac.in

ABSTRACT

Carnatic music is replete with continuous pitch movement called *gamakas* and can be viewed as consisting of constant-pitch notes (CPNs) and transients. The stationary points (STAs) of transients – points where the pitch curve changes direction – also carry melody information. In this paper, the precision of sung notes in Carnatic music is studied in detail by treating CPNs and STAs separately. There is variation among the nineteen musicians considered, but on average, the precision of CPNs increases exponentially with duration and settles at about 10 cents for CPNs longer than 0.5 seconds. For analyzing STAs, in contrast to Western music, *rāga* (melody) information is found to be necessary, and errors in STAs show a significantly larger standard deviation of about 60 cents.

To corroborate these observations, the music was automatically transcribed and re-synthesized using CPN and STA information using two interpolation techniques. The results of perceptual tests clearly indicate that the grammar is highly flexible. We also show that the precision errors are not due to poor pitch tracking, singer deficiencies or delay in auditory feedback.

1. INTRODUCTION

The precision of sung notes in Western classical music has been well studied [3, 13, 19]. However, as far as we know, they have not been published for Indian classical music. Previous controlled precision studies were typically concerned with long constant-pitch notes (e.g. [3]), or vibratos [18]. This approach is not suitable for Carnatic Music (CM), where *gamakas* are characterized by expansive pitch movements. Previous work on Indian music, such as [11], studied *gamakas* by analyzing *svaras*. However, the term ‘*svara*’ denotes both the note in the musical scale and the *gamakas* that embellish it. Thus, separating the steady parts of the pitch from the continuous movement is beneficial [5] [17].

In this paper, we quantify the precision of constant-pitch notes (CPNs) and stationary points (STA) separately (see

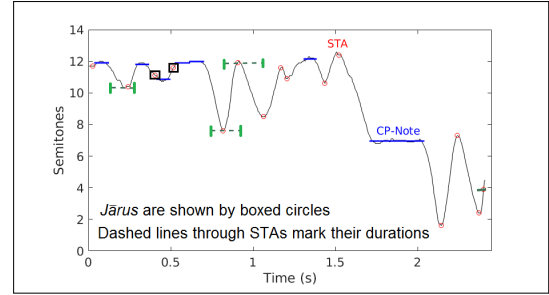


Figure 1. (From [21]) Blue lines are CPNs and red circles, STAs. Some STAs’ neighbors and duration are shown.

Name	Sa	Ri	Ga	Ma	Pa	Da	Ni
Carnatic	S	R1 R2	G2 G3	M1 M2	P	D1 D2	N2 N3
Western	C	C# D	D# E	F F#	G	G# A	A# B

Table 1. *Svara*-names and positions of the 12 semitones/octave for Carnatic & Western music. C is the tonic; the correspondence is well-defined only for CPNs.

Figure 1 for examples). We adapt below their definitions from [21] and use the *svara* names given in Table 1.

1. *Silence*-segments (SIL) are identified by the pitch-tracking algorithm [15].
2. A *constant-pitch note* (CPN) is one whose pitch does not vary from its mean pitch by more than 0.3 semitones and lasts for at least 80 ms. Non-SIL and non-CPN regions are called *transients*. *Anchor note(s)* are CPN(s) that flank transient(s).
3. *Stationary points* (STAs) [4, 20], are pitch positions where a continuous pitch curve changes direction. In [4] STAs also occur in CPNs, but they are restricted to the transients in this paper. STAs carry melody information [12] and are useful analytically [4, 14].
4. The *duration* of CPNs and SILs is fairly straightforward. The duration of a STA, typically 100 ms, is defined in [21]. See Figure 1 for an illustration.

The rest of the paper is organized as follows. Section 2 describes a method to statistically analyze precision in CM. Section 2.2 then focuses on precision-errors in CPNs



© V S Viraraghavan, H A Murthy, R Aravind. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** V S Viraraghavan, H A Murthy, R Aravind. “Precision of Sung Notes in Carnatic Music”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

(CPN-errors). In Section 2.3, we discuss the ambiguity inherent in measuring the precision-errors in STAs (STA-errors) and propose the use of *rāga*-specific information to overcome it. In Section 3, we observe that STAs have about half the precision of short CPNs, suggesting a flexible grammar. Section 4 describes two re-synthesis techniques with different interpolation schemes, which are used in a listening experiment that confirms this flexibility. Section 5 discusses the nature of this flexibility in the grammar.

2. PRECISION-ERROR MEASUREMENTS

2.1 Database, CPNs and STAs

For this work, the database comprised the same 84 concert pieces used in [21], which is a subset of [8]. These are in the *rāgas tōḍī*, *bhairavī*, *kharaharapriyā*, *kāmbhōji*, *śankarābharanam*, *varāḷī*, and *kalyāṇī*. The database has the dominant pitch (strictly ‘fundamental frequency’) tracked according to [15] and the tonic identified by the algorithm specified in [7]. Silence segments identified by the pitch tracker are ignored. For non-SIL segments in each piece in the database, we convert the fundamental frequency values to semitones (or equivalent cents) with reference to its tonic. Henceforth, the term ‘pitch’ in this paper implies measurement in semitones or cents.

Algorithm 1 of [21] is run hierarchically for the duration-threshold of CPNs set to 1000 ms, 300 ms, and 80 ms to get an initial set of CPNs (CPN-set-f). It is then run backwards (in time) to get CPN-set-b. Only CPNs in the intersection of these two sets are retained. Algorithm 2 of the same work is used to identify STAs. STAs adjacent to two CPNs of unequal pitch on either side, and having an intermediate pitch value (*jārus*, see Figure 1) are ignored. Nineteen professional singers, whose renditions had sufficient data for analysis are chosen.

2.2 Precision-errors of CP-notes

We measure the statistics of the error of the *mean value* of a CPN compared to a target. Instead of assuming a musical scale, the target pitch-values of CPNs are obtained statistically as the mean-values of a pitch class [13, 19]. That is, the locations of the significant peaks ($> 0.01 \times \text{max value}$) in the histogram of CPN pitch values folded to one octave are chosen as the target pitch values. This step is repeated for each piece independently and only CPNs longer than 150 ms are considered in finding target CPN pitch-values. Two examples are shown in Figures 2(a) and 2(b). In Figure 2(a), which corresponds to a piece of length just under 49 minutes, the important notes of the *rāga śankarābharanam*, S, G3 and P are evident. Three other notes (M1, R2 and D2) that seldom occur in the *rāga* as CPNs or anchor notes, have very small peaks. There is no peak at N3, which reflects its rarity as a CPN in the *rāga*. In Figure 2(b), corresponding to a piece of length 47 minutes, the peak at R2 is not in the defined scale of *rāga tōḍī*, but Carnatic musicians are aware of its use as an anchor note.

A CPN-error is defined as the difference of a CPN’s mean in semitones from the closest target CPN pitch-value. Qualitatively, it is expected that longer CPNs have better precision. To study this behavior, CPNs were grouped by duration according to Equation 1, where the bin-width, $B_w = 40$ ms. An additional bin was used for any duration over 440 ms.

$$\text{Bin}_i = [iB_w, (i+1)B_w], i \in \{2, 3, \dots, 10\} \quad (1)$$

Figure 3(a) shows the histogram of CPN-errors for three duration bins. Figures 3(b) and 3(c) show the quantile plots [22] for two duration ranges. Note that the number of samples for the longer CPNs are smaller than for shorter ones and thus show more outliers. We also ran the Shapiro-Wilk parametric hypothesis test of composite normality¹, with the default confidence level, $\alpha = 0.05$. The test showed that CPN-errors are, in general, *not* normally distributed. In fact, less than a dozen duration-bins out of over 200 across all singers showed a normal distribution. Nevertheless, we focus on the first two orders of statistics – mean and standard deviation – of the CPN-errors and STA-errors. Further, we treat the standard deviation of the errors as a quantitative measure of the precision.

The means and standard deviations of CPN-errors for the 19 singers are shown in Figure 4 as a function of duration. The means are ± 3 cents for all duration-bins. While there is variation among singers, there is a trend of the standard deviation of CPN-errors decreasing with duration.

2.3 Statistics of Stationary Points

2.3.1 Ambiguity in defining STA targets

The peaks identified in Figure 2(a) correspond well with *rāga*-characteristics even though explicit *rāga*-information is not used in identifying them. This result is encouraging, and the natural step is to adopt the same procedure for identifying STA target pitch-values. Figure 5(a) shows the histogram of STAs, with significant peaks identified exactly as for CPNs for the same piece that corresponds to Figure 2(a). Clearly, they do *not* cluster around scale notes. Further, where the peaks are visible, they are wider than in the case of CPNs. This suggests a larger tolerance for STA pitch errors and is worth verifying. Figure 6 shows a manually annotated spectrogram (using the method of reassignment [1, 10]) of an excerpt from a piece by a very famous singer, known for her exceptional tonal purity. Manual annotation removes the possibility of errors in fundamental frequency tracking. Sixteen of 37 STAs are at semitone values that are not expected in the *rāga*, but on listening to this sample, there is no hint of pitch errors. With STA-errors being of the order of a semitone, the simple histogram-based technique used for CPNs will not suffice. Thus, we propose the use of domain knowledge from CM to define target pitch-values for STAs.

¹ <https://in.mathworks.com/matlabcentral/fileexchange/13964-shapiro-wilk-and-shapiro-francia-normality-tests?focused=3823443&tab=function>

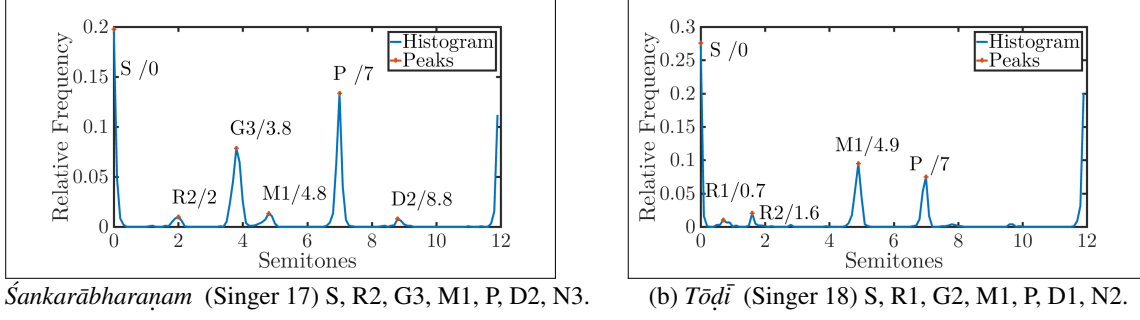


Figure 2. Histogram of CPNs of two sample *rāgas*. The CPNs cluster around centers close to the notes of the just-tempered scale. Indian music note names and their values in semitones are marked per peak. Each *rāga*’s scale-sequence is also given.

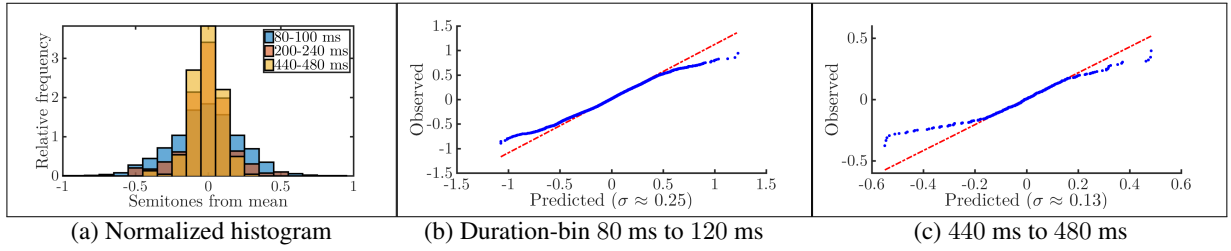


Figure 3. CPN-errors for Singer 04: Histogram and quantile-plots two duration-bins. Unmarked axes are in semitones.

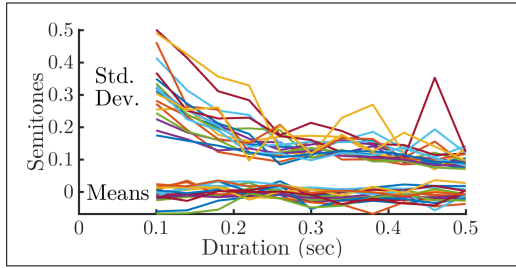


Figure 4. CPN-error statistics for 19 singers. Mean-values are ± 3 cents, except for two singers. The standard deviations are in a wider band, but decrease with duration.

2.3.2 Restricting Measurements to Specific STAs

As explained above, when a sequence of adjacent STAs is encountered, their target pitch-values are not easy to define. To reduce ambiguity, we propose restricting the measurements to a specific type of STA: one that is adjacent to at least one CPN. This effectively pegs one side of the continuous pitch movement, thus providing a practically usable reference. We can then define the precision-error of such a STA with respect to its adjacent CPN. Let such a CPN have a mean pitch p_c in semitones. Then, the target scale-note of this STA is from one of $\mathbb{S} = \{[p_c \pm 1], [p_c \pm 2], [p_c \pm 3]\}$, where $[\cdot]$ denotes rounding the pitch to the nearest integer semitone. In the rare cases that a STA is adjacent to a CPN on both sides, \mathbb{S} is a union of the sets formed by each adjacent CPN. Note that the elements of \mathbb{S} are integer semitones. The mean errors will be affected by a few cents, but as we shall see later, the standard deviation of STA-errors is much larger than the differences between corresponding notes of different mu-

<i>Gamakas</i>	Elements from \mathbb{S}	In \mathbb{S}'
To {R2, M1, P}	$\{p_c - 2, p_c + 1, p_c + 3\}$	Yes
To {R1, G2, M2}	$\{p_c - 3, p_c - 1, p_c + 2\}$	No

Table 2. Oscillatory *gamakas* at G3 in *śankarābharanam*

sical scales. Thus, the equal-tempered scale, or any other similar scale, can be used to define target pitch-values for STAs, with only a marginal effect on the measured precision. For consistency with Section 2.2, only CPNs and STAs that have a duration ≥ 150 ms are included in the measurement.

For each CPN in a *rāga*, the valid STA pitch-targets are a subset \mathbb{S}' of \mathbb{S} . For example, with the context being an anchor note, say $p_c = G3$ in the *rāga śankarābharanam*, the choices shown in Table 2 can be made. Such rules are not fully documented and are known more by practice. A (proprietary) synthesis algorithm that uses these rules was used to check and correct them in an iterative manner. Finally, overshoots and undershoots of STAs have been reported in the literature [17]. To account for them, STAs in ascending movements of pitch, i.e. where a STA is a local maximum, and in descending movements, where a STA is a local minimum, were measured separately. These subsets of STAs show histograms with sharp peaks in Figures 5(b) and 5(c). Corresponding to Table 2, the upward *gamakas* from G3 (4 semitones) to M1 (5 semitones) and P (7 semitones) are visible in Figure 5(b) and the downward *gamaka* to R2 (2 semitones), in Figure 5(c). Further, the upward *gamaka* from D2 (9 semitones) to N2 (10 semitones) in Figure 5(b) matches CM practice, although N2 is not in the *rāga*’s scale.

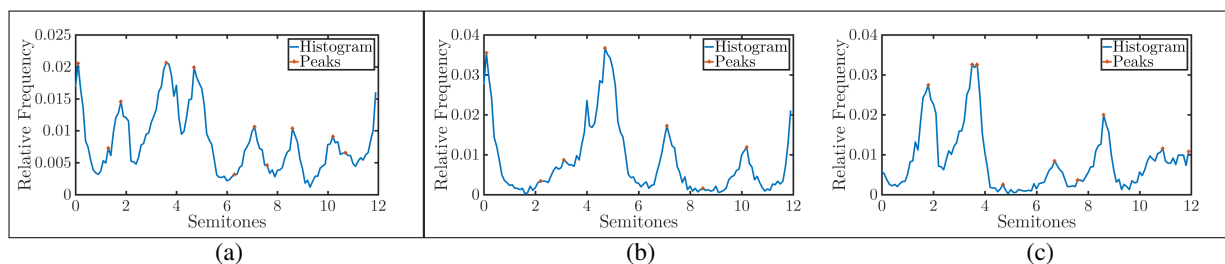


Figure 5. (a) Histogram of STAs for the same piece as in Figure 2(a). Peaks identified automatically are not clustered around notes of any musical scale. However, visually, STAs adjacent to anchor notes show sharp peaks in both (b) upward and (c) downward movements. To avoid clutter, note names and exact peak-locations are not given.

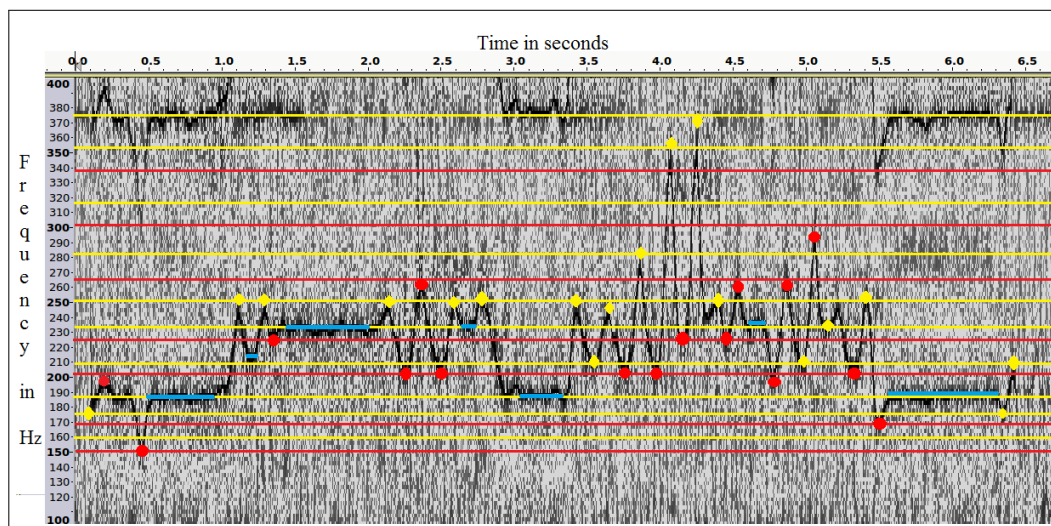


Figure 6. CPNs (blue lines) and STAs (circles and diamonds) in the spectrogram of an excerpt in the *rāga śankarābharaṇam*. The lower dark black curve is the pitch emphasized by reassignment. The tonic (Sa) is 188 Hz. Horizontal lines mark semitones from D1 to Ś. Yellow diamonds mark STAs at expected pitch values and red circles, at unexpected ones. This musician is famous for tonal purity and singer-errors can be discounted.

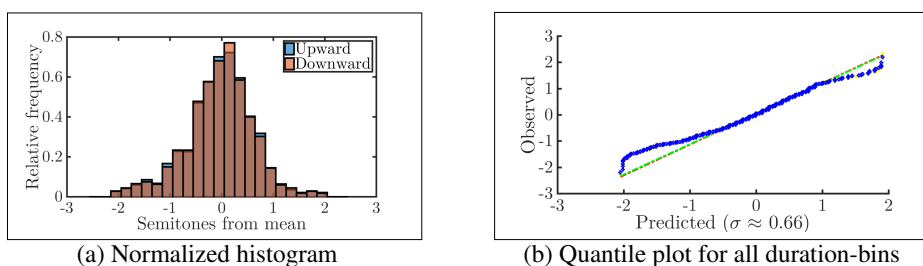


Figure 7. STA-errors for Singer 04: Histogram and quantile-plot. Unmarked axes are in semitones.

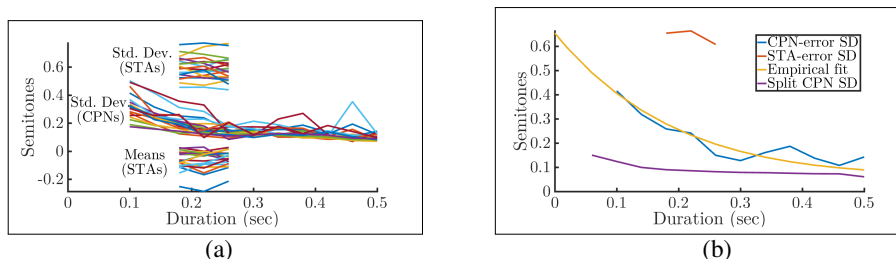


Figure 8. (a) Means and standard deviations (SDs) of precision errors in STAs for 19 singers. The SDs are mostly in a band of ± 10 cents and is more or less constant, unlike that of CPNs (also shown for comparison). (b) SDs of CPN-errors and STA-errors for Singer 04, and for smaller CPNs 'split' from larger ones. The empirical fit for this singer is also shown.

The precision error, ϵ , of any STA chosen thus, with pitch value p semitones, is measured as:

$$I = \arg \min_i |p - p_i| \quad (2)$$

$$\epsilon = p - p_I \quad (3)$$

where i indexes \mathbb{S}' per anchor note per *rāga*. Table 2 does not consider *gamakas* traversing more than 3 semitones, which are rare in CM. Even so, STAs for which $|\epsilon| > 2$ semitones are not included in the measurement. With this more reliable definition of the precision error of these STAs, their statistics were collected per singer. The histogram and quantile plots of STA-errors (Singer 04) are presented in Figures 7(a) and 7(b), which have two virtually indistinguishable plots each: one for STAs in ascending movements, and another for downward movements. The STA-errors also do not follow a normal distribution.

3. ANALYSIS OF OBSERVATIONS

Figure 8(a) shows the means and standard deviations of the STA-errors for the 19 singers. Unlike that of CPN-errors, the standard deviation of STA-errors does *not* vary much with duration. The standard deviation of STA-errors is also about twice as large as that of CPN-errors for CPNs of duration around 100 ms. The slight negative bias of the means of STA-errors is not yet understood.

Modeling the observed precision can be useful in applications such as transcription. We propose a singer-dependent, composite empirical model to predict the standard deviations of *both* CPN-errors and STA-errors. In this model with two components, the first exponentially decreases with time, and can be expressed as:

$$\sigma_x(t) = \sigma_s e^{-t/T} \quad (4)$$

As most singers do not ever reach zero precision-error for even very long CPNs, we need to introduce a constant term σ_r . Thus, the overall standard deviation, as a function of time t , can be written as:

$$\sigma(t) = \sqrt{\sigma_x^2(t) + \sigma_r^2} \quad (5)$$

The forms of Equations 4 and 5 serve to emphasize the first component for low values of t (STAs and short CPNs) and the second, for larger t (long CPNs). For each singer, σ_r was set as the average of the CPN standard deviations for the last three duration-bins. The values of t were chosen as the mid-points of the duration-bins of Equation 1, i.e. $(i + 0.5)B_w$. We also propose that STAs can be viewed as ‘point-CPNs’. For example, the short CPN around 1.4 seconds in Figure 1 can be shrunk to a point, which would make it a STA. Practically, a STA lasts at least as long as the shift in windowing algorithms. For the data presented in this paper, this shift is 4.44 ms. Consequently, we set the value of $\sigma(0)$ as the average value of the standard deviation of STA-errors.

The best values of T and σ_s were found by minimizing the mean squared error of the standard deviation predicted by Equations 4 and 5 over the following ranges of

Singer ID	σ_r	T	σ_s	RMSE
01	11	120	45	2.6
03	9	115	60	2.7
04	13	170	65	2.5
05	10	155	55	0.8
06	9	200	45	2.6
07	10	165	55	2.0
08	13	225	65	3.7
09	8	155	70	3.7
10	10	165	55	1.8
13	15	145	50	2.4
17	9	165	60	3.1
18	11	180	85	6.0
19	8	115	60	2.0
20	19	140	50	7.0
21	11	210	40	2.8
27	9	110	50	1.8
28	11	120	65	1.8
30	10	75	65	1.8
31	14	145	55	3.9

Table 3. Prediction parameters for the nineteen musicians aliased by Singer ID. The parameter T is in ms and σ_r , σ_s and the root mean squared error (RMSE) are in cents.

values: $T \in \{20, 21, \dots, 300\}$ ms and $\sigma_s \in \{i\theta, i = 1, 2, \dots, 20\}$, $\theta = 0.05$ semitones. An example for Singer 04 is given in Figure 8(b), which shows a good fit of the model with the observations. The quantitative measure of the fit (RMSE) and the values of T , σ_s and σ_r for the 19 singers are given in Table 3. The typical value of σ_r being in the range 0.08 to 0.15 semitones (one outlier at 0.19) is in good agreement with the precision range of 0.1 to 0.15 semitones reported for choir singers [19]. It remains to be seen if STAs of types other than in Section 2.3 also follow the same statistics.

The cause(s) of the precision-error trend is (are) not fully clear, but we eliminate one possibility here. Two types of auditory feedback have been reported in the literature. The first is involuntary, and takes about 100 ms to take effect, and another, voluntary taking about 300 ms [9]. For the smaller duration bins, the voluntary mechanism does not have time to effect corrections. Even the involuntary mechanism does not seem to explain all of the variance. Specifically, the precision error of CPNs is not mirrored in successively longer initial segments of CPNs. That is, for each CPNs of duration $t \geq 300$ ms, and preceded by SIL, several CPNs of duration $iT_{\text{split}}, i \in \{1, 2, \dots, \lfloor \frac{t}{T_{\text{split}}} \rfloor\}$, where $T_{\text{split}} = 20$ ms, were split from it and their precision for the duration-bins (Equation 1) were calculated. This result for Singer 04 is also shown in Figure 8(b). It is clear that, for durations around 100 ms, the standard deviation of precision error for such ‘split notes’ is far lower than that for CPNs found from the definition. Thus, it cannot explain the trend seen in CPN-errors. Further, this was seen to be true for all the singers.

Pair-set	U, cosine	R, cosine	R, linear
1	B	T	NA
2	NA	B	T

Table 4. Pair-sets *per rāga* for the quantization algorithms (U or R) and interpolation schemes (linear or cosine) of the synthesized samples in the pair-wise comparison test.

4. EXPERIMENTAL CONFIRMATION

A musicological view (e.g [12]) is that STAs should be precise, which is not consistent with the observations presented hitherto. In a previous experiment², the STAs at R1 and D1 in the *rāga pantuvārālī* were shifted to R2 and D2 respectively. The shift is not perceivable in the audio synthesized from manual notation [16]. While this experiment confirms the relatively large precision-error of the STAs, perceptual tests were not conducted. Independently, we designed an experiment³ to confirm the large variability in the pitch-values of STAs. We concatenated one excerpt at slow speed, and another relatively fast one, both chosen from *ālāpanas* in four important, *gamaka*-heavy *rāgas*. These pieces of approximately one-minute duration were transcribed in two ways. In uniform quantization (U), the pitch in semitones of each CPN or STA, p , was set to $p' = [p]$, where $[·]$ denotes rounding towards the nearest integer semitone. In this method, 13% to 19% of STAs were quantized to pitch values not in the *rāga*-specific list \mathbb{R} , i.e. Table 2 extended to all anchor notes and octaves. In the second method (R), with i indexing \mathbb{R} , and $e_i \in \mathbb{R}$, a CPN/STA pitch (p) was set to $p' = e_i$ where:

$$I = \arg \min(|p - e_i|) \quad (6)$$

The STAs and CPNs were then synthesized by constructing a pitch curve that was constant at CPN-locations. STAs and CPNs (and SILs) were connected to each other by using linear or cosine-interpolation. For the latter, the phase was set to $0(\pi)$ at a starting higher (lower) STA/CPN and to $\pi(0)$ at the ending lower (higher) STA/CPN. These pitch curves, sampled at 1 kHz, and the short-term energy of the original excerpts resampled to 1 kHz, were fed to a good-quality, 5-harmonics synthesis algorithm [6]. We asked listeners to rate pair-wise, the synthesis samples on the basis of adherence to the *rāga*. The *pair-sets* *per rāga* are given in Table 4. Twenty four participants (twelve experts) heard all *rāgas* of pair-set 1 in the order *kāmbhōji*, *śankarābharaṇam*, *tōḍī*, and *bhairavī*. Within a pair, the order was random. This was then repeated for pair-set 2.

Table 5 shows the results of the listening test. For each pair, the preference-percentages, the average rating across participants and *rāgas*, the average difference between ratings, and the average absolute-difference between the ratings are given. All measures indicate that there is no clear

Measure	U vs. R		Cosine vs. Linear	
	U	R	Cosine	Linear
Preference	34 (33)	34 (35)	25 (33)	26 (21)
percentage	Equal: 31 (31)		Equal: 49 (46)	
Avg. rating	3.5 (3.3)	3.4 (3.3)	3.6 (3.4)	3.6 (3.4)
Avg. diff.	0.0 (0.1); 0.7 (0.7)		0.0 (0.1); 0.7 (0.7)	

Table 5. Results of the pair-wise comparison test (expert-ratings in brackets). In the last row, average differences and average absolute-differences are separated by semicolons.

preference among the possibilities. This result can also explain why many interpolation schemes for *gamakas* – Bezier curves [2], Hermite polynomials [6, 14], Gaayaka software [16], sine curves [17] etc. – all seem to work.

5. CONCLUSIONS

We presented the statistics of precision errors of CPNs and STAs, and measured their means and standard deviations as a function of duration. While the analysis was done separately, the precision-errors for both CPNs and STAs were empirically fitted in a single model. We also presented the results of a listening experiment using the outputs of two synthesis algorithms, both of which also treat CPNs and STAs separately. The key conclusions that can be drawn from this work are:

1. The standard deviation of the precision error in CPNs decreases with duration. A nominal value of 20 cents may be used for a duration of 200 ms, and 10 cents for long CPNs.
2. The standard deviation of the precision error in STAs is independent of duration (45 to 85 cents across singers). A nominal value of 60 cents may be used.
3. Even experts could not tell apart samples that had STAs quantized to notes within a *rāga*'s grammar and those that did not. Also, samples that used linear and cosine interpolation were not distinguishable.

Thus, it appears that there is a large tolerance for both the precision of STAs and the way they are connected, which implies a highly flexible grammar for CM. Point 3 suggests that a rich transcription for a CM piece need not be unique. It may also indicate that re-synthesis quality cannot be used to rate algorithms for rich transcription. However, given that Figures 5(b) and 5(c) show peaks only at expected locations, the existence of unique rich transcription with a large tolerance for STAs is likely.

Finally, it should be noted that the flexibility in its grammar does **not** mean that 'CM is imprecise' or that the precision of STAs is unimportant. Instead, this flexibility should be seen as natural in a form of music that employs continuous pitch movement in profusion.

² <http://carnatic2000.tripod.com/maya.zip>

³ <https://www.iitm.ac.in/donlab/pctestmusic/index.html?owner=venkat1&testid=test1&testcount=8>

6. ACKNOWLEDGMENTS

This research was partly funded by the European Research Council under the European Unions Seventh Framework Program, as part of the CompMusic project (ERC grant agreement 267583). We thank all participants of the listening test and Ms. Anju Leela Thomas and Mr. Krishnaraj Sekhar for helping set it up.

7. REFERENCES

- [1] Audacity 2.1.2 [Audio editor] (2016). Latest version freely available from: <http://www.audacityteam.org/download/>. Accessed: 2017-04-16.
- [2] Bret Battey. Bezier spline modeling of pitch-continuous melodic expression and ornamentation. *Computer Music Journal*, 28(4):25–39, 2004.
- [3] Simone Dalla Bella, Jean-François Giguère, and Isabelle Peretz. Singing proficiency in the general population. *The journal of the Acoustical Society of America*, 121(2):1182–1189, 2007.
- [4] Shrey Dutta and Hema A Murthy. A modified rough longest common subsequence algorithm for motif spotting in an alapana of carnatic music. In *Communications (NCC), 2014 Twentieth National Conference on*, pages 1–6. IEEE, 2014.
- [5] Kaustuv Kanti Ganguli, Ashwin Lele, Saurabh Pinjani, Preeti Rao, Ajay Srinivasamurthy, and Sankalp Gulati. Melodic shape stylization for robust and efficient motif detection in hindustani vocal music. In *Proc. of the National Conference on Communication (NCC)*, 2017.
- [6] Kaustuv Kanti Ganguli and Preeti Rao. Discrimination of melodic patterns in indian classical music. In *Communications (NCC), 2015 Twenty First National Conference on*, pages 1–6. IEEE, 2015.
- [7] Sankalp Gulati, Ashwin Bellur, Justin Salamon, Vignesh Ishwar, Hema A Murthy, and Xavier Serra. Automatic tonic identification in indian art music: approaches and evaluation. *Journal of New Music Research*, 43(1):53–71, 2014.
- [8] Sankalp Gulati, Joan Serrà, Kaustuv Kanti Ganguli, Sertan Sentürk, and Xavier Serra. Time-delayed melody surfaces for rāga recognition. In *ISMIR*, pages 751–757, 2016.
- [9] Timothy C Hain, Theresa A Burnett, Swathi Kiran, Charles R Larson, Shajila Singh, and Mary K Kenney. Instructing subjects to make a voluntary response reveals the presence of two components to the audio-vocal reflex. *Experimental Brain Research*, 130(2):133–141, 2000.
- [10] Kunihiro Koda, Roger Gendrin, and C de Villedary. Analysis of time-varying signals with small bt values. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):64–76, 1978.
- [11] Gopala Krishna Koduri, Joan Serrà Julià, and Xavier Serra. Characterization of intonation in carnatic music by parametrizing pitch histograms. In *Proceedings of the 13th International Society for Music Information Retrieval Conference; 2012 Oct 8-12;*, 2012.
- [12] TM Krishna and Vignesh Ishwar. Carnatic music: Svara, gamaka, motif and raga identity. In *Serra X, Rao P, Murthy H, Bozkurt B, editors. Proceedings of the 2nd CompMusic Workshop; 2012 Jul 12-13; Istanbul, Turkey*. Universitat Pompeu Fabra, 2012.
- [13] Peter Q Pfordresher, Steven Brown, Kimberly M Meier, Michel Belyk, and Mario Liotti. Imprecise singing is widespread. *The Journal of the Acoustical Society of America*, 128(4):2182–2190, 2010.
- [14] HG Ranjani, Deepak Paramashivan, and Thippur V Sreenivas. Quantized melodic contours in indian art music perception: Application to transcription. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pages 174–180, 2017.
- [15] Justin Salamon and Emilia Gómez. Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics. *IEEE Transactions on Audio, Speech and Language Processing*, 20:1759–1770, 2012.
- [16] M Subramanian. Carnatic music-automatic computer synthesis of gamakams. *Sangeet Natak*, 43(3):28–36, 2009.
- [17] Srikumar K Subramanian, Lonce Wyse, and Kevin McGee. A two-component representation for modeling gamakas of carnatic music. In *Proc. of 2nd CompMusic Workshop*, 2012.
- [18] Johan Sundberg. Acoustic and psychoacoustic aspects of vocal vibrato.
- [19] Sten Ternström and Johan Sundberg. Intonation precision of choir singers. *The Journal of the Acoustical Society of America*, 84(1):59–69, 1988.
- [20] Venkatasubramanian V, K R Ramakrishnan, and H V Sahasrabudde. Music information retrieval using continuity. In *Proceedings of the Symposium on the Frontiers of Research on Speech and Music (FRSM 2004)*, number 228, pages 74–83. Annamalai University, Chidambaram, 2004.
- [21] Venkata S Viraraghavan, R Aravind, and Hema Murthy. A statistical analysis of gamakas in carnatic music. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pages 243–249, 2017.
- [22] Martin B Wilk and Ram Gnanadesikan. Probability plotting methods for the analysis for the analysis of data. *Biometrika*, 55(1):1–17, 1968.

REVISITING SINGING VOICE DETECTION: A QUANTITATIVE REVIEW AND THE FUTURE OUTLOOK

Kyungyun Lee¹ Keunwoo Choi² Juhan Nam³

¹ School of Computing, KAIST

² Spotify Inc., USA

³ Graduate School of Culture Technology, KAIST

kyungyun.lee@kaist.ac.kr, keunwooc@spotify.com, juhannam@kaist.ac.kr

ABSTRACT

Since the vocal component plays a crucial role in popular music, singing voice detection has been an active research topic in music information retrieval. Although several proposed algorithms have shown high performances, we argue that there is still room for improving the singing voice detection system. In order to identify the area of improvement, we first perform an error analysis on three recent singing voice detection systems. Based on the analysis, we design novel methods to test the systems on multiple sets of internally curated and generated data to further examine the pitfalls, which are not clearly revealed with the currently available datasets. From the experiment results, we also propose several directions towards building a more robust singing voice detector.

1. INTRODUCTION

Singing voice detection (or VD, vocal detection) is a music information retrieval (MIR) task to identify vocal segments in a song. The length of each segment is typically at a frame level, for example, 100 ms. Since singing voice is one of the key components in popular music, VD can be applied to music discovery and recommendation as well as various MIR tasks such as melody extraction [7], audio-lyrics alignment [31], and artist recognition [2].

Existing VD methods can be categorized into three different classes. *First*, the early approaches focused on the acoustic similarity between singing voice and speech, utilizing cepstral coefficients [1] and linear predictive coding [10]. The *second* class would be the majority of existing methods, where the systems take advantages of machine learning classifiers such as support vector machines or hidden Markov models, combined with large sets of audio descriptors (e.g., spectral flatness) as well as dedicated new features such as the Fluctogram [14]. *Lastly*, there is a recent trend towards feature learning using deep neural networks, with which the VD systems learn optimized

features for the task using a convolutional neural network (CNN) [27] and a recurrent neural network (RNN) [11]. They have achieved state-of-the-art performances on commonly used datasets with over 90% of the true positive rate (recall) and accuracy.

We hypothesize that there are common problems in existing VD methods in spite of such well-performing metrics that have been reported. Our scope primarily includes methods in the second and third classes since they significantly outperform those in the first class. Our hypothesis was inspired by inspecting the assumptions in the existing algorithms. The most common one, for example, has been made on the spectro-temporal characteristics of singing voices; that they include frequency modulation (or vibrato) [15, 24], which leads to our analysis on whether there are any problems by pursuing to be a vibrato detector. We can also raise similar questions on the behavior of the systems in the third class, the deep learning-based systems, by examining on their assumptions and results. Based on the analysis, we invent a set of empirical analysis methods and use them to reveal the exact types of problems in the current VD systems.

Our contributions are as follows :

- A quantitative analysis to clarify and classify common errors of three recent VD systems (Section 4)
- An analysis using curated and generated audio contents that exploit the discovered weakness of the systems (Section 5)
- Suggestions on future research directions (Section 6)

In addition, we review previous VD systems in Section 3 and summarize the paper in Section 7.

2. BACKGROUND

2.1 Problem definition

Singing voice detection is usually defined as a binary classification task about whether a short audio segment input includes singing voice. However, the details have been rather empirically decided. By ‘short’, the segment length for prediction is often 100 ms or 200 ms. ‘Audio’ can be provided as stereo, although they are frequently downmixed to mono. More importantly, singing voice is not clearly defined, for example, leaving the question that



© Kyungyun Lee, Keunwoo Choi, Juhan Nam. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Kyungyun Lee, Keunwoo Choi, Juhan Nam. “Revisiting Singing Voice Detection: A quantitative review and the future outlook”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

	Size	Annotations	Past VD papers	Notes
Jamendo Corpus	93 tracks (443 mins)	Vocal activation	[11], [24], [12], [13], [27], [26]	Train/valid/test split from [22]
RWC Popular Music	100 tracks (407 mins)	Vocal activation, instrument annotation	[26], [27], [14] [13]	VD annotation by [16]
MIR-1K	100 short clips (113 mins)	Vocal activation, pitch contours	[9]	Regular speech files provided
MedleyDB	122 tracks (437 mins)	Melody annotation, pitch annotation	[26]	Multitrack

Table 1: A summary of public datasets relevant to singing voice detection

background vocals should be regarded as singing voice or not. In previous works, this problem has been neglected since the majority of songs in datasets do not include background vocals that are independent of the main vocals. These will be further discussed in Section 6.

2.2 Public Datasets

In Table 1, four public datasets for evaluating VD systems are summarized. Three of them are well described by Lehner et al. [12]: Jamendo Corpus [22], RWC Popular Music Database [4] and MIR-1K Corpus [8]. In addition, we add MedleyDB [3], which is a multitrack dataset, composed of raw mono recordings for each instrument as well as processed stereo mix tracks. Although it does not provide annotations for vocal/non-vocal segments, we utilize the annotations for the instrument activation, which considers vocals as one of the instruments. There can be more benefits by using the multitrack dataset for VD research, which will be discussed in Section 6.

2.3 Audio Representation

In this section, we present the properties as well as the underlying assumptions of various audio representations in the context of VD. Previous works have used a combination of numerous audio features, seeking easier ways for the algorithm to detect the singing voice. They range from audio representations such as short-time Fourier transform (STFT) to high-level features such as onsets and pitch estimations.

- **STFT** provides a 2-dimensional representation of audio, decomposing the frequency components. STFT is probably the most basic (or ‘raw’) representation in VD, based on which some other representations are either designed and computed, or learned using deep learning methods.
- **Mel-spectrogram** is a mel-scaled frequency representation and usually more compressive than STFTs and originally inspired by the human perception of speech. Being closely related to speech provides a good motivation to be used in VD, therefore mel-spectrogram has been actively used as an input representation of CNNs [27] and RNNs [11]. When deep learning methods are used, mel-spectrogram is often preferred due to its efficiency compared to STFT.

- **Spectral Features** such as spectral centroid and spectral roll-off are statistics of a spectral distribution of a single frame of time-frequency representations (e.g., STFT). A particular and most noteworthy example is **Mel-Frequency Cepstral Coefficients** (MFCCs). MFCCs have originally been designed for automatic speech recognition and take advantages of mel-scale and Fourier analysis for providing approximately pitch-invariant timbre-related information. They are often (assumed to be) relevant to MIR tasks including VD [12, 25]. Spectral features, in general, are not robust to additive noise, which means that they would be heavily affected by the instrumental part of the music when used for VD.

3. MODELS

In this section, we introduce three recent and distinctive VD systems that have improved the state-of-the-art performances along with the details of our re-implementation of them.¹ They are briefly illustrated in Figure 1, where x and y indicate the input audio signal and the output prediction, respectively.

3.1 Lehner et al. [14] (FE-VD)

This feature engineering (FE) method, FE-VD is based on the Fluctogram, spectral flatness, vocal variance and other hand-engineered audio features. We select this model for its rich and task-specific feature extraction process to compare with the other models. Although the features are ultimately computed frame-wise, context from the adjacent frames are taken into account, supposedly enabling the system to use dynamic aspect of the features. The features are aimed to reduce the false positive rate caused by the confusion between singing voice and pitch-varying instruments such as woodwinds and strings. Random forest classifier was adopted as a classifier, achieving an accuracy of 88.2% on the Jamendo dataset. While their methods have shown reduction in the false positive rates on strings, Lehner et al. mentions woodwinds such as pan flutes and saxophones still show high error rate.

Following [14], we extract 6 different audio features (the Fluctogram, spectral flatness, spectral contraction, vocal variances, MFCCs and delta MFCCs), resulting in 116-dimensional features per frame. We use input size of

¹ <http://github.com/kyungyunlee/ismir2018-revisiting-svd>

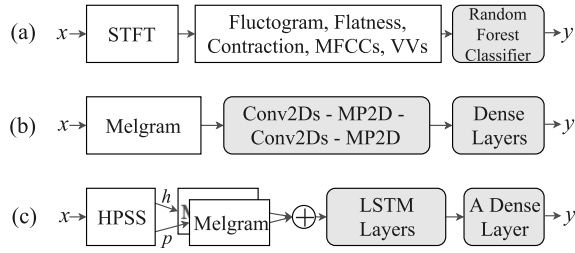


Figure 1: Block diagrams for three VD systems – (a) FE-VD [14], (b) CNN-VD [27], and (c) RNN-VD [11]. x and y for input audio signal and output prediction (probability of singing voice). Rounded and gray blocks are trainable classifiers or layers. The details of the features in (a) are explained in [14]. In (c), ‘+’ indicates frequency-axis concatenation and ‘h’ and ‘p’ are the separated harmonic/percussive components.

1.1 seconds as the input to the random forest classifier, where we perform grid search to find optimal parameters. As a post-processing step, we apply the median filter of 800 ms on the output predictions.

3.2 Schlüter et al. [27] (CNN-VD)

Recently, VD systems using deep learning models have shown the state-of-the-art results [11, 26, 27]. These systems often use basic audio representations such as STFT as an input to the models such as CNN and RNN, expecting the relevant features are learned by the model. We first introduce a CNN-based system [27].

Schlüter et al. suggested a deep CNN architecture with 4 3-by-3 2D convolution layers. We name the CNN model CNN-VD. As a result, the system extracts ~~trained~~, relevant *local* time-frequency patterns from its input, a mel-spectrogram. During training, they apply data augmentation such as pitch shifting and time stretching on the audio representation. They reported that it reduces the error rate from 9.4% to 7.7% on the Jamendo dataset.

Our CNN architecture is identical to the original one and uses an input size of 115 frames (1.6 sec). However, we do not perform data augmentation or threshold optimization for a fair comparison with other models. Thus, we use 0.5 as the threshold value for the prediction. Here, we also apply median filter of 800 ms for smoothing.

3.3 Leglaive et al. [11] (RNN-VD)

As another deep learning-based system, Leglaive et al. [11] proposed a recurrent neural network with bi-directional long short-term memory units (Bi-LSTMs) [6], with an assumption that temporal information of music can provide valuable information for detecting vocal segments. We name this system RNN-VD. For the classifier input, the system performs double-stage harmonic-percussion source separation (HPSS) [20] on the audio signal to extract signals relevant to the singing voice. For each frame, Mel-spectrograms of the obtained harmonic and percussive components are concatenated as an input for the classifier. Several recurrent layers followed by a shared densely-

	FE-VD	CNN-VD	RNN-VD
Acc.(%)	87.9	86.8	87.5
Recall(%)	91.7	89.1	87.2
Precision(%)	83.8	83.7	86.1
F-measure(%)	87.6	86.3	86.6
FPR(%)	15.3	15.1	12.2
FNR(%)	8.3	10.9	12.8

Table 2: Results of our implementations on the Jamendo test set. FPR and FNR refer to false positive rate and false negative rate, respectively.

connected layer (also known as time-distributed dense layer) yield the output predictions for each input frame. This model achieves the state-of-the-art result without data augmentation, showing accuracy of 91.5% on the Jamendo dataset. From this result, although the contributions from additional preprocessing vs. recurrent layers may be combined, we can assume that past and future temporal context help to identify vocal segments.

For our RNN architecture, we use the best performing model from the original article [11], one with three hidden layers of size 30, 20 and 40. The input to the model is 218 frames (3.5 seconds) and the threshold value of 0.5 is used to predict the presence of singing voice as done in [11].

4. EXPERIMENT I: ERROR CATEGORIZATION

The purpose of this experiment is to identify common errors in the VD systems through our implementation of models from Section 3. The results and observations lead to the motivation of experiments in Section 5. Librosa [18] is used in audio processing and feature extraction stages.

4.1 Data and Methods

Three systems (FE-VD, CNN-VD, and RNN-VD) are trained on the Jamendo dataset with the suggested split of 61, 16 and 16 for training, validation and test sets [22], respectively. They are primarily tested on the Jamendo test set. For qualitative analysis, we also utilize MedleyDB.

4.2 Results

The test results of our implementation are shown in Table 2. We did not focus on fine-tuning individual models because three systems altogether are used as a tool to get a generalized view of the recent VD systems, thus showing slightly lower performances compared to the results in original papers. Overall, FE-VD, CNN-VD and RNN-VD show a negligible difference on the test scores. We observe trends that are similar to the original papers in terms of performance and the precision/recall ratio.

Upon listening to the misclassified segments, we categorize the source of errors into three classes – pitch-fluctuating instruments, low signal-to-noise ratio of the singing voice, and non-melodic sounds.

Song Title	Confusing inst	FE-VD	CNN-VD	RNN-VD
Llrlandaise	Woodwind, Synth	46.6	29.5	22.0
Castaway	Elec. Guitar	62.5	56.5	24.2
Say me Good Bye	N/A	2.8	3.0	2.5
Inside	N/A	5.9	6.7	5.0

Table 3: False positive rate (%) of each system for 4 songs from the Jamendo test set. The top 2 songs are the ones ranked within the top 5 lowest accuracy and the bottom 2 songs are the ones ranked within the top 5 highest accuracies at song level across all three systems.

4.2.1 Pitch-fluctuating instruments

Classes of instruments such as strings, woodwinds and brass exhibit similar characteristics as the singing voice, which we refer to as being ‘voice-like’ [28]. By ‘voice-like’, we consider three aspects of the signal, namely, pitch range, harmonic structure, and temporal dynamics (vibrato). Especially, we find temporal dynamics as important attributes that are recognized by the VD systems to identify vocal segments.

Frequency modulation, also known as vibrato, resembles the modulation created from the vowel component of singing voice. This is illustrated in Figure 2, where mel-spectrograms of both female vocalist and an electric guitar show curved lines. We observe that this similarity causes further confusion in the system.

In Table 3, we list two songs found among the top 5 least/most accurately predicted songs in the test set of all three systems. The woodwind in ‘05 - Llrlandaise’ causes high false positives, which may be due to the presence of vibrato and the similarity in pitch range to that of soprano singers (above 220 Hz). FE-VD and CNN-VD show poor performance on woodwinds, probably because the Fluctogram of FE-VD and small 2D convolution kernels of CNN-VD are specifically designed to detect vibrato as one of the features for identifying singing voice. In the same song, all three systems show confusion with the synthesizer. Synthesizers mimicking pitch-fluctuating instruments are particularly challenging as it is difficult to characterize them as a specific instrument type.

In addition, electric guitars are one of the most frequently found sources of false positives, as can be seen from ‘03 - castaway’, mostly caused by the recognizable vibrato patterns. We find the confusion worse when the guitar is played with effects such as wah-wah, which imitates the vowel sound of the human. Lastly, we note that some of the other problematic instruments in our test sets include saxophones, trombones and cellos, which are well-known ‘voice-like’ instruments.

This observation, regarding the system pitfalls on vibrato patterns, is further investigated in Section 5.1.

4.2.2 Signal-to-noise ratio and the performance

Lastly, we note that all the three systems are affected by the signal-to-noise ratio (SNR), or the relative gain of vocal component, as one can easily expect. All of the three

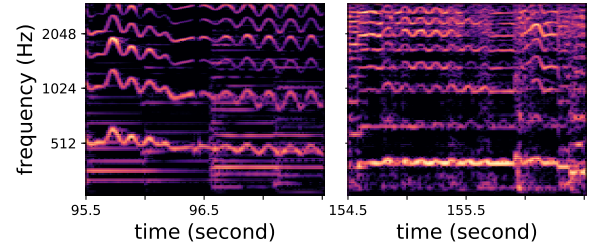


Figure 2: Excerpts of mel-spectrograms from MedleyDB: ‘Handel_TornamiAVagheggiar’ with female vocalist (left) and ‘PurlingHiss_Lolita’ with electric guitar (right) (see Section 4.2.1.)

systems exhibit high false negative rate when the vocal signal is relatively at a low level.

In systems such as FE-VD, where audio features such as MFCCs or spectral flatness are used, the performance varies by SNR because the features are statistics of the whole bandwidth which includes not only the target signal (vocal) but also additive noise (instrumental). VD systems with deep neural networks are also not free from this issue since the low-level operation in the layers of deep neural networks may end up being a simple pattern matching by computing correlation.

This is a common phenomenon in other tasks as well, e.g., speech recognition, and we continue the discussion to a follow-up experiment in Section 5.2 and finally a suggestion on the problem definition and dataset composition in Section 6.

4.2.3 Non-melodic Sources

Although the interest of most VD systems appears to lie mainly in the melodic component of the song, we expected the system to learn percussive nature of the singing voice as well, which is exhibited by consonants from the singers. Therefore, our hypothesis is whether the system is confused by the consonants of singing voice and percussive instruments, resulting in either *i*) missing consonant parts (false negative) or *ii*) mis-classifying percussive instruments (false positive).

From our test results, we encounter false positive segments containing snare drums and hi-hats, but the exact cause of this misclassification is unclear. We further tested the system with drum set solos for potential false positives and with a collection of consonant sounds such as plosives and fricatives from the human voice for potential false negatives, but we did not observe a clear pattern in misclassification. Although we do not conduct further experiment on this, it suggests a deeper analysis, which may also lead to a clear understanding of preprocessing strategies including HPSS.

5. EXPERIMENT II: STRESS TESTING

5.1 Testing with artificial vibrato

Based on the confusion between ‘voice-like’ instruments and singing voice, we hypothesize that the current VD sys-

tems use vibrato patterns as one of the main tools for vocal segment detection. We explore the degree of confusion for each VD system by testing them on synthetic vibratos with varying rate, extent and formant frequencies.

5.1.1 Data Preparation

We create a set of synthetic vibratos with low pass-filtered sawtooth waveforms with $f_0=220$ Hz. We vary the modulation rate and frequency deviation (f_Δ) to investigate their effects. Furthermore, we apply 5 bi-quad filters at the corresponding formant frequencies (3 for each) to synthesize so that they would sound like the basic vowel sounds, 'a', 'e', 'i', 'o', 'u' [29]. The modulation rate ranges in $\{0.5, 1, 2, 4, 6, 8, 10 \text{ Hz}\}$ and the frequency deviation ranges in $\{0.01, 0.1, 0.3, 0.6, 1, 2, 4, 8 \text{ semitones}\}$ with respect to its f_0 . As a result, the set consists of 7 (rates) \times 8 (f_Δ 's) \times 6 (5 formants + 1 unfiltered) = 336 variations.

5.1.2 Results

Figure 3 shows the result of the prediction by the three VD systems on the synthetic vibratos. The accuracy of 1.0 indicates that the system does not confuse the artificial vibratos with singing voice. Here, we observe the performance difference of each model, which were not visible from looking at the scores in Table 2. In general, confusion areas tend to be concentrated on the bottom left to the center area of the graph. The extent and rate of the artificial tones that are highly misclassified seem to be around the range of vibratos of singers, which is said to be around 0.6 to 2 semitone with rate around 5.5 to 8 Hz [30]. We also observe a within-system difference, i.e., the presence and the type of formants affect the models. For instance, vibratos mimicking the vowel 'a' cause higher misclassification in all three models.

FE-VD performs much better than the latter two systems. Note that FE-VD is a feature engineering model, where unique features, such as the Fluctogram and vocal variance, are mostly adapted from the ones used in speech recognition task. As these features were intentionally designed to reduce false positives from pitch-varying instruments, it appears to significantly reduce error rate on vibratos with rate and extent that are beyond the range of human singers.

CNN-VD confuses slightly wider range of vibratos. This is expected to some extent since the model prominently uses 3×3 filters on mel-spectrogram to detect local features, which can be regarded as a *local* pattern detector. In other words, the locality of CNN results in a system that is easily confused by frequency modulation regardless of the non-singing voice aspects of the signal. This implies that the model may benefit from looking at a varying range of time and frequency to learn vocal-specific characteristics such as timbre [21].

Lastly, RNN-VD performs better than the CNN-VD, though worse than FE-VD. On detecting vocal and non-vocal segments, it seems natural, even for humans, that past and future temporal context help. Also, we presume that the preprocessing of double stage HPSS contributes to

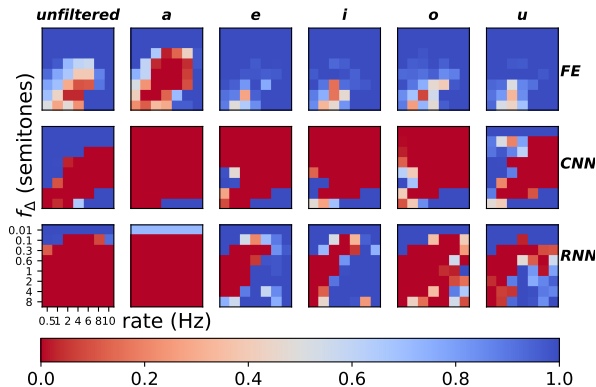


Figure 3: Heat-maps of the accuracies of the vibrato experiment result. Each row corresponds to VD systems (FE-VD, CNN-VD, RNN-VD) and each column corresponds to the formant ('unfiltered', 'a', 'e', 'i', 'o', 'u'). Within each heat map, x- and y-axes correspond to the vibrato rate and frequency deviation as annotated on the lower-left subplot (see Section 5.1)

the robustness of the system against vibrato. Again, this observation leaves a question of separating the contributions from preprocessing and model structure.

5.2 Testing with SNR

In this experiment, VD systems are tested with vocal gain adjusted tracks to further explore the behavior of the systems on various scenarios, which can reflect the real-world audio settings of live recordings and radios, for example.

5.2.1 Data preparation

We create a modified test set using 61 vocal-containing tracks provided by MedleyDB. We use the first 30 seconds of the songs to build a pair of (vocal, instrumental) tracks. Vocal tracks are modified with SNR of $\{+12 \text{ dB}, -12 \text{ dB}, +6 \text{ dB}, -6 \text{ dB}, 0 \text{ dB}\}$.

5.2.2 Results

The results of the energy level robustness test are presented in Figure 4 with false positive rate, false negative rate, and overall error rate. We see a consistent trend across the performance of all three VD systems, which is once again an expected pattern as aforementioned in Section 4.2.2 – that increasing SNR helps to reduce false negatives. Overall error rate also exhibits a noticeable decrease in common with higher SNRs. In practice, one could take advantage of data augmentation with changing SNR to build a more robust system. More importantly, it can be part of the evaluation procedure for VD, as we discuss in Section 6.

While the VD systems behave similarly on all test cases, we note that FE-VD, owing to its additional features, shows lowest variance and lowest value for the false positive rate. Also, our assumption that the double-stage HPSS, which filters out vocal-related signals, would make RNN-VD more robust against SNR is observed to be not necessarily true as we clearly see performance differences across the varying SNR test cases.

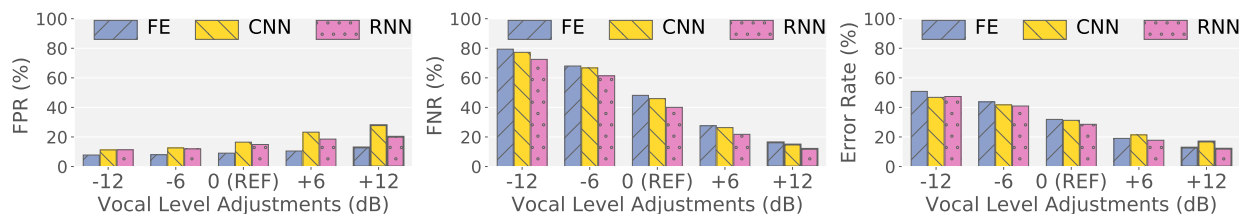


Figure 4: False positive rates, false negative rates, and overall error rates for the three systems in the stress testing with controlling SNR (see Section 5.2).

6. DIRECTIONS TO IMPROVE

6.1 Defining the problem and the datasets

6.1.1 Defining singing voice

By using the annotations in datasets such as Jamendo, many VD systems implicitly assume that the target ‘singing voice’ is defined as vocal components that correspond to the *main melody*. Other voice-related components such as backing vocal, narration, humming, and breathing are not clearly defined to be singing voice or not.

In some applications, however, they can be of interest. For example, a system may want to find purely instrumental tracks, avoiding tracks with backing vocal. In this case, the method should consider backing vocal as singing voice. However, for Karaoke applications, only the singing voice of the main melody would matter.

Therefore, an improvement can be made on defining the VD problem and creating datasets. For the annotation, a hierarchy among the voice-related components can be useful for both structured training and evaluation of a system [17, 23]. For the audio input, we see a great benefit of multitracks, where main vocal melody, backing vocal, and other components are provided separately.

6.1.2 Varying-SNR scenarios

For a long while, varying SNR had been one of the common ways to evaluate speech recognition or enhancement using dataset such as Aurora [5]. As observed in Section 4.2.2, it can be used as a ‘test-set augmentation’ to measure the performance of a system more precisely. Also, it can be an additional data augmentation method along with the ones in [27] to build a VD system more robust to various audio settings, such as audios from user generated videos. These can both be easily achieved with a multitrack dataset in practice.

6.1.3 Measuring dataset noise

Human annotators are neither perfect or identical, thus causing annotation noise and disagreement. Since VD is a binary classification problem, we may remain optimistic by assuming that the annotation noise is a matter of temporal precision, which is arbitrary and not agreed among many datasets so far. For example, in RWC Popular Music [16], “short background segments of less than 0.5-second duration were merged with the preceding region” and the annotations have 8 decimal digits (in second), while in Jamendo, they are 3 decimal digits. The optimal precision

may depend on human perception of sound which is often said around 10 ms in general [19]. Although it would require a deeper investigation, the current temporal precision may be too high, leading to evaluate the systems with an overly precise annotation.

6.2 Learning from human perception

The characteristic of voice was the main motivation in the very early works exploiting speech-related features [1, 10]. Clearly, however, those approaches that solely relied on speech features showed limited performances. While following works has improved the performance, as our experiments have demonstrated through this paper, the systems do not completely take advantage of the cues that human is probably using, e.g., the global formants, linguistic information, musical knowledge, etc.

6.3 Preprocessing

A light-weight VD system was introduced in [12] where only MFCCs were used to achieve an accuracy of 84.8% on the Jamendo dataset. This implies that there is a possibility to achieve better performance by optimizing the preprocessing stage. One of the unanswered questions is the effect of the preprocessing stage in RNN-VD [11] as well as whether similar processing could lead to better performance with other systems, e.g., CNN [27].

7. CONCLUSIONS

In this paper, we suggested that there still are several areas to improve for the current singing voice detectors. In the first set of experiments, we identified the common errors through error analysis on three recent systems. Our observations that the main sources of error are pitch-fluctuating instruments and low signal-to-noise ratios of the singing voice motivated us to further perform stress tests. Testing with synthetic vibratos revealed that some systems (FE-VD) are more robust to non-vocal vibratos than others (CNN-VD and RNN-VD). SNR-varying test showed that SNR manipulation greatly affects the current VD systems, thus it can potentially be used to strengthen the VD systems to become invariant to a wider range of audio settings. As we propose several directions for a more robust singing voice detector, we note that defining the VD problem is dependent on the goal of the system, thus using multitrack datasets can be beneficial. Our future interest is to further investigate on SNR to extend VD systems on uncontrolled audio settings and to examine different components of individual systems, including the preprocessing stage.

8. ACKNOWLEDGEMENTS

We thank Bernhard Lehner and Simon Leglaive for active discussion and code, Jeongsoo Park for sharing Ono's code. This work was supported by the National Research Foundation of Korea (Project 2015R1C1A1A02036962).

9. REFERENCES

- [1] Adam L Berenzweig and Daniel PW Ellis. Locating singing voice segments within music signals. In *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the*, pages 119–122. IEEE, 2001.
- [2] Adam L Berenzweig, Daniel PW Ellis, and Steve Lawrence. Using voice segments to improve artist classification of music. In *Audio Engineering Society Conference: 22nd International Conference: Virtual, Synthetic, and Entertainment Audio*. Audio Engineering Society, 2002.
- [3] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A multitrack dataset for annotation-intensive mir research. In *ISMIR*, volume 14, pages 155–160, 2014.
- [4] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Popular, classical and jazz music databases. In *Proc. of the 3rd International Society for Music Information Retrieval Conference (ISMIR)*, volume 2, pages 287–288, 2002.
- [5] Hans-Günter Hirsch and David Pearce. The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)*, 2000.
- [6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [7] Chao-Ling Hsu, Liang-Yu Chen, Jyh-Shing Roger Jang, and Hsing-Ji Li. Singing pitch extraction from monaural polyphonic songs by contextual audio modeling and singing harmonic enhancement. In *Proc. of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 201–206, 2009.
- [8] Chao-Ling Hsu and Jyh-Shing Roger Jang. On the improvement of singing voice separation for monaural recordings using the MIR-1K dataset. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(2):310–319, 2010.
- [9] Chao-Ling Hsu, DeLiang Wang, Jyh-Shing Roger Jang, and Ke Hu. A tandem algorithm for singing pitch extraction and voice separation from music accompaniment. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(5):1482–1491, 2012.
- [10] Youngmoo E Kim and Brian Whitman. Singer identification in popular music recordings using voice coding features. In *Proc. of the 3rd International Conference on Music Information Retrieval (ISMIR)*, volume 13, page 17, 2002.
- [11] Simon Leglaive, Romain Hennequin, and Roland Badeau. Singing voice detection with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 121–125. IEEE, 2015.
- [12] Bernhard Lehner, Reinhard Sonnleitner, and Gerhard Widmer. Towards light-weight, real-time-capable singing voice detection. In *Proc. of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, pages 53–58, 2013.
- [13] Bernhard Lehner, Gerhard Widmer, and Sebastian Böck. A low-latency, real-time-capable singing voice detection method with lstm recurrent neural networks. In *Signal Processing Conference (EUSIPCO), 2015 23rd European*, pages 21–25. IEEE, 2015.
- [14] Bernhard Lehner, Gerhard Widmer, and Reinhard Sonnleitner. On the reduction of false positives in singing voice detection. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 7480–7484. IEEE, 2014.
- [15] Maria E Markaki, André Holzapfel, and Yannis Stylianou. Singing voice detection using modulation frequency feature. In *SAPA@ INTERSPEECH*, pages 7–10, 2008.
- [16] Matthias Mauch, Hiromasa Fujihara, Kazuyoshi Yoshii, and Masataka Goto. Timbre and melody features for the recognition of vocal activity and instrumental solos in polyphonic music. In *Proc. of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 233–238, 2011.
- [17] Brian McFee and Juan Pablo Bello. Structured training for large-vocabulary chord recognition. In *Proc. of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- [18] Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, et al. librosa 0.5. 0, 2017.
- [19] Brian CJ Moore. *An introduction to the psychology of hearing*. Brill, 2012.
- [20] Nobutaka Ono, Kenichi Miyamoto, Jonathan Le Roux, Hirokazu Kameoka, and Shigeki Sagayama. Separation of a monaural audio signal into harmonic/percussive components by complementary dif-

- fusion on spectrogram. In *Signal Processing Conference, 2008 16th European*, pages 1–4. IEEE, 2008.
- [21] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra. Timbre analysis of music audio signals with convolutional neural networks. In *Signal Processing Conference (EUSIPCO), 2017 25th European*, pages 2744–2748. IEEE, 2017.
- [22] Mathieu Ramona, Gaël Richard, and Bertrand David. Vocal detection in music with support vector machines. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 1885–1888. IEEE, 2008.
- [23] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6517–6525, 2017.
- [24] Lise Regnier and Geoffroy Peeters. Singing voice detection in music tracks using direct voice vibrato detection. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 1685–1688. IEEE, 2009.
- [25] Martin Rocamora and Perfecto Herrera. Comparing audio descriptors for singing voice detection in music audio files. In *Brazilian symposium on computer music, 11th. san pablo, brazil*, volume 26, page 27, 2007.
- [26] Jan Schlüter. Learning to pinpoint singing voice from weakly labeled examples. In *Proc. of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 44–50, 2016.
- [27] Jan Schlüter and Thomas Grill. Exploring data augmentation for improved singing voice detection with neural networks. In *Proc. of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 121–126, 2015.
- [28] Emery Schubert and Joe Wolfe. Voicelikeness of musical instruments: A literature review of acoustical, psychological and expressiveness perspectives. *Musicae Scientiae*, 20(2):248–262, 2016.
- [29] Julius Orion Smith. *Introduction to digital filters: with audio applications*, volume 2. Julius Smith, 2007.
- [30] Renee Timmers and Peter Desain. Vibrato: Questions and answers from musicians and science. In *Proc. Int. Conf. on Music Perception and Cognition*, volume 2, 2000.
- [31] Ye Wang, Min-Yen Kan, Tin Lay Nwe, Arun Shenoy, and Jun Yin. Lyrically: automatic synchronization of acoustic musical signals and textual lyrics. In *Proc. of the 12th annual ACM international conference on Multimedia*, pages 212–219. ACM, 2004.

VOCALS IN MUSIC MATTER: THE RELEVANCE OF VOCALS IN THE MINDS OF LISTENERS

Andrew Demetriou^{1,2}

Andreas Jansson²

Aparna Kumar²

Rachel M. Bittner²

¹ Multimedia Computing Group, TU Delft, The Netherlands

² Spotify Inc., New York City, USA

ABSTRACT

In music information retrieval, we often make assertions about what features of music are important to study, one of which is vocals. While the importance of vocals in music preference is both intuitive and anticipated by psychological theory, we have not found any survey studies that confirm this commonly held assertion. We address two questions: (1) what components of music are most salient to people’s musical taste, and (2) how do vocals rank relative to other components of music, in regards to whether people like or dislike a song. Lastly, we explore the aspects of the voice that listeners find important. Two surveys of Spotify users were conducted. The first gathered open-format responses that were then card-sorted into semantic categories by the team of researchers. The second asked respondents to rank the semantic categories derived from the first survey. Responses indicate that vocals were a salient component in the minds of listeners. Further, vocals ranked high as a self-reported factor for a listener liking or disliking a track, among a statistically significant ranking of musical attributes. In addition, we open several new interesting problem areas that have yet to be explored in MIR.

1. INTRODUCTION

The Music Information Retrieval (MIR) community has historically focused on content-based understanding of music. The type of content-based analysis studied over time is typically driven by the data available to the task, or the interests of the specific researchers. An alternative motivator could be to study topics that are salient in the minds of listeners, especially with respect to listener’s musical preference. Specifically, understanding which attributes of music contribute the most to music preference, and their relative weight, could help guide research efforts. One attribute of music we would expect to be salient in the minds of listeners is the singing voice.

Psychology research anticipates the importance of the human voice as a salient stimulus, and as a component of

music in particular. The human ability to communicate exceeds that of any other species studied thus far, with both speech and singing being cultural universals reliant on vocal production. It is theorized that the advanced human ability to communicate, discriminate, and to experience emotional responses in vocalizations has allowed for the emergence of music [8]. Our emotions are often accompanied by involuntary changes in our physiology and nonverbal expressions, such as facial expressions and vocalizations [15]. Our reactions to the emotional content expressed in the vocals in music may have similar effects. As such, much psychological research has focused on the singing voice even more than speech, due to the precision required to execute and process musical vocalizations [5]. This makes musical vocals a well-anticipated candidate for study as a feature of music, as we would expect people to have a sophisticated ability to deliver, empathize with, and process vocal communications.

We would therefore expect that the vocals in music would be an especially salient component, if not the most salient. While a complete review is beyond the scope of this paper, some research is particularly worth noting. For example, it has been shown that both adults [18] and children [17] recall melodies more correctly when sung with the voice than when played with instruments. Hutchins and Moreno [5] review literature that shows relatively precise perception of pitch in the human voice, yet fewer noticeable pitch errors in the voice relative to musical instruments or synthesized voices [6]. Neuroscience studies show specific areas of the brain involved in processing human voices [2]. Although similar regions of the brain are involved in processing both music and voices, there is differential processing of the human voice relative to music [1]. As such, the human voice may be processed as a uniquely significant sound.

However, while prior research suggests that vocals would be especially relevant to music preference, no study to our knowledge has assessed the importance of the voice in music, relative to other musical components. To address this gap, we test the hypothesis that the voice is as or more important than other musical components across implicit and explicit datasets, using traditional social science techniques, as well as data mining techniques. First, we mine data available from Spotify, including playlist titles, search data and artist biographies, to test whether terms related to vocals are prevalent. However, we show that the results of the data mining are inconclusive as to whether or not



© Andrew Demetriou, Andreas Jansson, Aparna Kumar, Rachel M. Bittner. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Andrew Demetriou, Andreas Jansson, Aparna Kumar, Rachel M. Bittner. “Vocals in Music Matter: the Relevance of Vocals in the Minds of Listeners”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

vocals are salient in the minds of listeners. Specifically, it is not clear whether the vocals can be disentangled from other factors in playlist titles and search queries, such as genre. For more conclusive results, we gather data from users explicitly. To this aim we conduct two online survey studies: the first gathered subjective data on the salient components of music directly from listener reports, which were separated into semantic categories using card-sorting. The second asked participants to rank the semantic categories from the first study in terms of importance to their musical preference. We conclude that two aspects related to the voice are especially salient, namely the voice itself, and the lyrics of the song. Furthermore, we highlight the importance of gathering explicit data to complement implicit techniques, in situations where factors may not be easily disentangled.

2. VOCALS IN SEMANTIC DATA

Prior research has shown that semantic descriptors of music may be an appropriate means for users to query music databases [12]. Given the large amount of semantic data available to Spotify such as playlist titles, search results, and artist biographies, one might hypothesize that terms describing the vocals would commonly appear in this implicit data.

2.1 Playlist Tags and Search Queries

Non-common words or groups of words and emojis appearing in the titles of a large number of Spotify’s user-generated playlists were aggregated to create a list of the 1000 most frequently occurring *tags*. Each of these 1000 tags was assigned a category by a professional curator based on the tag itself and information from the tracks most frequently associated with the tag. The categories, determined by the curator, were Genre (e.g. “K-Pop”), Mood (e.g. “sad”), Activity (e.g. “gym”), Popularity (e.g. “Today’s hits”), Artist (e.g. “Justin Timberlake”), Era (e.g. “70’s”), Culture (e.g. “Latin”), Lyrics (e.g. “clean”), Rhythm (e.g. “groove”), Instrument (e.g. “guitar”), Tempo (e.g. “slow”), Voice (e.g. “female singers”), or Other (e.g. “favorites”, “Jenna”, “hi”). The percentage of playlists containing each of these tag categories is displayed in Figure 1, top.

Surprisingly, we see that tags explicitly related to vocals are not at all common compared to other types of tags, with the most common tags being related to genre, mood, or activity. Playlist titles can be viewed as labels for groups of music, and this analysis suggests that people do not often label groups of music based on explicit characteristics of the vocals. However, specific vocal characteristics (as well as many other musical attributes) may be implicit in many of the other tag categories, particularly for genre, mood, and artist. As vocal delivery style and genre are closely related, emotions communicated by the voice and the mood of the collection of songs may be related, and as each artist has a unique voice, we conclude that the relative weight of vocals may not have been disentangled from other factors.

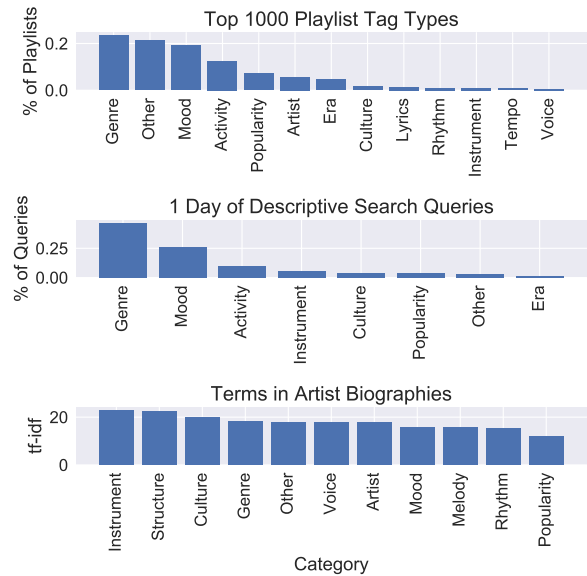


Figure 1: (Top) Percentage of Spotify playlists containing one of the top 1000 tags corresponding to each category. (Middle) Percentage of descriptive search queries corresponding to each tag category, sampled from one day of search data. (Bottom) tf-idf for each term category in artist biographies compared with Wikipedia term frequencies.

We perform a similar analysis on descriptive terms from one day’s worth of Spotify search queries, and obtained similarly inconclusive results, shown in Figure 1, middle.

2.2 Artist Biographies

Finally, we analyze descriptive terms that occur in 100,000 professionally authored artist biographies on Spotify. We use TF-IDF [16] to retrieve terms that are distinctive to music writers, by comparing the frequency of terms in artist biographies to the frequency of the same terms in Wikipedia. The 100 most distinctive terms, grouped into semantic categories, are displayed in Figure 1, bottom. While many terms are much more frequent in music text (e.g. “bassist”, “jazz”, “songwriter”), vocals specifically were not more frequently mentioned than other musical aspects. One can hypothesize that the TF-IDF method is insufficient for this particular task, due to vocals being commonly discussed outside the context of music, and thus a relatively more common word in Wikipedia.

2.3 Conclusions

Our results thus far do not show support for our general hypothesis. It may be the case that the intuitive notion of the relevance of vocals to user preference is misleading. On the other hand, it may also be the case that the importance of vocals is implicit in this data, as certain vocal styles are indicative of genre or mood. As such, the overlap between the voice and a number of the tags and descriptors analyzed prevents us from disentangling the unique effect of the voice from other musical components.

3. VOCALS IN SURVEY DATA

In order to disentangle the unique effect of the voice among other components, we gathered explicit data from users. Specifically, we conducted two online survey studies in order to collect self-reported data on 1) the salient components of music, and 2) their relative ranking. Unlike prior surveys, such as [12] that presented users with short musical excerpts and groupings of adjectives to rate, we allowed the users to freely enter their responses to the question “When you listen to music, what things about the music do you notice?”. This allowed us to assess whether vocals would emerge as a salient component of music. In addition, we explored what aspects of the voice users report as being important to their musical taste.

3.1 Survey 1: Semantic Components of Music

The aim of our first survey was to establish an unranked set of self-reported salient components of music. While our hypothesis was that the vocals would be prominent, it was crucial to avoid biasing respondents as the data collected were explicit. As such, our first survey asked participants what they notice when listening to music that might make them like or dislike a song. We deliberately did not specify anything further, such as the type of music, or that we were interested in components of music, nor were participants asked to listen to musical excerpts so as not to bias responses. As an exploratory measure, we then asked participants to describe what about vocals specifically might make them like or dislike a song *after* the previous open ended questions, so as not to bias responses. Responses to these two open-response questions were manually sorted into semantic categories by the researchers.

3.1.1 Recruitment

A random sample of 50,000 people was drawn from the database of Spotify’s Monthly Active Users (MUAs), divided approximately equally between the United States and Canada. 860 individuals responded to the survey, however 224 did not respond to any questions beyond the consent form, and 9 were removed for giving nonsensical responses. 626 individuals — 338 women (average age 33.6 years with a standard deviation of 16.1); 288 men (average age 30.6 years with a standard deviation of 15.5) — completed the survey in its entirety.

3.1.2 Survey

An online consent form was first presented to respondents. We then asked:

Q1: When you listen to music, what things about the music do you notice? Please list as many as you can think of here:

The respondents were shown a screen with open-response format fields to complete, in which they could complete up to seven fields. On the following screen, respondents were presented with a list of their responses in random order, and asked:

emotions	When I can either relate or empathize with them and when the song projects the emotions onto me.
emotions	If it doesn't feel like there's emotion behind it, or somehow lacking.

Figure 2: Survey 1 sample answers for *Q3*. (Top) Card for an answer to *Q3a*. (Bottom) Card for an answer to *Q3b*.

Q2: Please rank how important the aspects you listed are to your musical preference, where 1 is the most important.

They were then asked the following two questions about the items they ranked from 1 to 3:

Q3: (a) What about ____ would make you like a song? (b) What about ____ would make you dislike a song?

Lastly, to explore what aspects of vocals may be relevant, participants responded to the following:

Q4: (Please ignore these questions if you’ve already mentioned the vocals, the voice, the singer/rapper etc.) (a) When would vocals make you like a song? (b) When would vocals make you dislike a song?

They were then given the opportunity to comment on the survey, and were shown a final debriefing screen.

3.1.3 Semantic Categorization

A number of partially completed surveys contained responses sufficiently complete for card sorting. 317 sufficient responses — 262 from the completed surveys as well as 55 sufficiently complete partial — were then card-sorted by a team of researchers. Card-sorting is a common technique used in social sciences and elsewhere to discover clusters of related concepts [14]. Traditionally, individuals are presented with physical paper “cards” that have terms and/or descriptions printed on them, printed pictures, or a group of objects. They are then asked to group items in a way that makes sense, given the research question. Here, we apply card-sorting to derive semantically meaningful groupings of musical components from the freely entered words and phrases that participants entered in each field.

Participant responses to *Q1* (i.e. “When you listen to music, what things do you notice?”) were printed twice, once next to their response to *Q3a* (“What about _ would make you like a song?”), and again next to the response to *Q3b* (“What about _ would make you dislike a song?”). As such, researchers had respondents’ top 3 terms printed out twice, once next to the positive descriptive aspects of the term, and once next to the negative descriptive aspects. A term (e.g. “the lyrics”) and its descriptor (e.g. “when they have meaning”) comprised a card. Figure 2 shows examples of positive and negative cards that were used in card sorting.

As some responses were unclear (e.g. “the melody” was mentioned, but the descriptor clearly focused on the quality of the singer’s voice), the research team was instructed

to look at both the term and its descriptor when determining its semantic category. The researchers then reviewed the cards a second time, and defined sub-categories where necessary.

3.1.4 Results

The output of this study was two sets of semantic categories: broad semantic categories of music, and vocal-specific semantic categories. Statistical testing was not possible, given the intentionally imprecise nature of the responses. However, out of the 626 responses to the first question, 186 (29.7%) mentioned the vocals, the voice, or the singer, 348 (55.6%) mentioned the lyrics, or the words, and 101 (16.1%) mentioned both. While this is no indication of relative importance, it does demonstrate that the voice and the lyrics were salient musical components to our respondents.

The broad semantic categories determined by the researchers are presented in the left column of Table 1 (note that the other results in Table 1 are from Study 2). The category of *Emotion/mood* referred to the ability of a song to evoke emotion, whether the emotion was a match or a mismatch to the current or desired mood or current activity, whether the emotion was desirable or undesirable, and nostalgia. *Voice* included genre related terms (e.g. mumble rap, metal, auto-tune, speechiness/rapping), descriptions of how the voice is used (e.g. unique/novel, screaming, pitch/pitch range, presence or absence of effects, intensity/effort/power, emotionality, authenticity, whininess/nasality, melodic-ness), skill, the innate qualities of the voice, liking/disliking, and the mix/blend. The *Lyrics* category represented items that indicated whether or not lyrics were present, their intelligibility, the presence of profanity, how “well” crafted they were, the “message”, the meaning behind them or general lyrical content and how relatable they are. *Beat/Rhythm* referred to whether it was liked/disliked, whether it “fit” the song, danceability, and uniqueness. The *Structure/complexity* of songs included liking or disliking the hook or chorus, and the song length. Instrumentation referred to drums, bass, and guitar. *Sound* referred to audio quality and related concerns. Self-explanatory categories included *Tempo/BPM*, the mention of a *Specific Artist*, *Genre*, *Harmony*, *Chords*, *Musician-ship*, *Melody*, and *Popularity/Novelty*.

3.2 Survey 2: Component Ranking

While the first study aimed at determining what attributes of music were salient in the minds of listeners, the aim of the second survey was to determine the relative importance of each of the components. Specifically, we explored whether the voice would be ranked highest among a list of musical attributes. To accomplish this, participants were asked to rank a list of attributes derived from the results of our first survey, thus allowing an assessment of whether or not vocals rank above other components.

3.2.1 Recruitment

A randomized sampling method was employed among the database of Spotify’s Monthly Active Users (MAUs) that had not opted-out of email correspondence. An email with a link to an online survey was sent to 50,000 potential respondents, approximately equally divided among the United States and Canada.

A total of 531 respondents — 263 of which were women (average age 31.8 years, with a standard deviation of 16.5); 268 were men (average age 34.2 years, with a standard deviation of 14.8) — completed the survey in its entirety. 429 participants completed the first half of the survey (broad semantic categories), whereas 360 participants completed the second half (vocal semantic categories).

3.2.2 Survey

An online consent form was first presented to respondents. The derived semantic categories were rephrased to be more easily understood (see Table 1, Description). Participants were presented with the new list of descriptions in random order, and asked to “Please click all the items below that would make you like or dislike a song.” They were then presented with a list of all the items they had clicked, also in random order, and asked to rank them.

As a continuation of our exploratory study of vocal characteristics, a second list was then presented, comprised of terms derived from the vocal and lyrics semantic categories. For clarity, the terms were rephrased as they appear in Table 2.

3.2.3 Analytic Strategy

Responses were subjected to Borda counting [3] and Robust Rank Aggregation [9]. Borda counting is a simple procedure for aggregating votes by summing ranks. The Borda score B_i for an item i is computed as $B_i = \sum_{p=0}^N (|r_p| - r_{p,i})$ where N is the number of participants, $r_{p,i}$ is participant p ’s rank of item i , starting at zero, and $|r_p|$ is the number of items ranked by p . The Borda method does not naturally extend to partial lists [4] — we have chosen to award higher scores to preferred items in long lists.

To verify the statistical significance of our findings we supplement the Borda count with Robust Rank Aggregation (RRA), in which we compare our survey results to a null hypothesis. Each item receives a score based on its observed position, compared to an expected random ordering. Upper bounds to p -values are computed using Bonferroni correction, with values of 1.0 indicating null findings. In this work we used the implementation provided by the ROBUSTRANKAGGREG package¹.

3.2.4 Results and Conclusion

Results can be found in Tables 1 and 2, with categories ordered by descending Borda count. We are able to show statistical significance of both the most salient broad and vocal semantic categories. Importantly, our results show that the Vocals and Lyrics ranked second and third among

¹ cran.r-project.org/web/packages/RobustRankAggreg

Broad Semantic Category	Description	Borda score	<i>p</i> -value
Emotion/mood	How it makes you feel - the emotions/mood	4641	<0.001
Voice	Voice/vocals	3688	<0.001
Lyrics	Lyrics	3656	<0.001
Beat/rhythm	Beat/rhythm	3460	<0.001
Structure/Complexity	How it's composed, the hook, the structure	2677	1.000
Musicianship	Skill of the musicians, musicianship	2583	1.000
Melody	The main melody	2577	1.000
Sound	The "sound", or the recording quality	2406	1.000
Specific Artist	The specific artist	2349	1.000
Genre	The specific genre	2293	1.000
Instrumentation	The musical instruments (e.g. drums, bass, guitar)	2084	1.000
Tempo/BPM	How fast or slow the song is	1828	1.000
Harmony	Harmony	1763	1.000
Chords	The chords	1086	1.000
Popularity/Novelty	How popular or unique it is	777	1.000

Table 1: Broad semantic categories and their clarifying descriptions created during Study 1, ordered by rankings from Study 2 (see Study 1 results for attribute descriptions). The Borda scores and *p*-values from Study 2 are reported in columns 3 and 4. Statistically significant *p*-values are shown in bold. *p*-values of 1.000 indicate that the ranking is no different from random.

the list of components (Borda scores and RRA agree on the order of the first four broad categories). This indicates that, relative to other musical components, respondents overall indicated the importance of the vocals and lyrics.

4. NEW AVENUES FOR RESEARCH

While the musical attributes related to the broad musical categories (Table 1) are well studied in MIR, the attributes related to vocals (Table 2) present a number of exciting and unexplored research directions. A limiting factor to studying some of these problems, as is often the case, is the availability of data, and we encourage researchers to focus data collection efforts in these areas as well. A further limiting factor is that users of online musical platforms may come from a specific demographic, e.g. regular internet users typically younger than 35, who engage in music related activities in about one third of the online time, have had at least some musical education, and have a preference for pop, rock and classical music [12]. In addition, our sample was derived from the U.S. and Canada. As such, a cross-cultural sample may differ in their relative preference for vocals.

Our exploratory data suggest that there is a vast space of research in tagging and measuring different qualities of the singing voice, such as whether a singing voice is authentic, powerful, natural, melodic, nasal, or emotional (Table 2, rows E, H, I, K, M and G). In addition to these categories, determined by untrained listeners, there are a number of other more specific categories such as modes of phonation that could be explored. Further, in addition to vocal qualities, there are genre-centric vocal styles, such as identifying rap or screaming (Table 2, rows S and O).

Another interesting and (as far as we are aware) unexplored research area is to measure whether a voice fits or

blends well with the background music (Table 2, row B). This is somewhat related to the problem of determining "mashability" in automatic-mashup generation. This is a broad problem that is likely based on many factors, such as the style of the vocalist compared to the background, the way the song is mixed, and the overall expectations of the musical genre. We suspect this could be most easily studied when isolated vocals/backgrounds are available in order to automatically generate examples of vocals that do not match the background by blending random combinations.

The problem of identifying whether a voice is "unique" is likely challenging (Table 2, row F), as it is not necessarily a quality that can be determined in isolation, but rather relative to many other voices. One possible approach to this problem would be to treat the problem as one of outlier detection.

Production effects applied to the singing voice are increasingly common, especially different types of distortion or the infamous auto-tune (Table 2, row Q). Automatic identification of these production effects presents an interesting challenge, and one where data could be automatically generated with the help of plugins for generating effects and databases with isolated vocals with corresponding backgrounds.

Measuring the relatability (Table 2, row J) of a singer is a quality that is relative to the listener, rather than absolute. Factors that could affect a singer's relatability could include the age, gender, culture or language of the singer relative to the listener, which might require automatic identification of each of these attributes of the singer.

Lyric intelligibility (Table 2, row L) has not been well studied, and also presents a novel challenge [7]. This problem does not necessarily directly require lyric transcription, and may be able to be determined from qualities of

	Vocal Semantic Categories	Borda score	<i>p</i> -value
A	Singing skill	3423	< 0.001
B	How well the voice fits or matches the rest of the music	3380	< 0.001
C	Lyrical skill / cleverness / wit	3145	< 0.001
D	The meaning, or the “message” of the words	3038	0.048
E	Authenticity / “realness”	2884	< 0.001
F	Uniqueness	2780	< 0.001
G	If the voice is emotional	2771	0.006
H	Voice strength / intensity / effort	2721	1.000
I	If the voice sounds natural	2480	1.000
J	Being able to relate	2256	1.000
K	If the voice is melodic	2202	1.000
L	Whether or not you can understand the lyrics	2056	1.000
M	If it’s whiny or nasal	1801	1.000
N	Whether or not there’s screaming	1771	1.000
O	The overall pitch, or the range of the pitch	1400	1.000
P	Whether or not there are lyrics	1250	1.000
Q	Whether it has production effects on it, like autotune	1230	1.000
R	Profanity, explicit lyrics	1086	1.000
S	Whether or not there is rapping	909	1.000

Table 2: Vocal-specific semantic categories from Study 1, ordered by rankings from Study 2. Columns 2 and 3 show the Borda scores and *p*-values. Statistically significant *p*-values are shown in bold. *p*-values of 1.000 indicate that the ranking is no different from random.

the audio. Similarly, determining whether a singing voice contains lyrics or is wordless has not been studied (Table 2, row P).

Automatic lyric transcription has been studied [11, 13] but is not yet solved, and would power the automatic estimation of many of these vocal attributes. For lyric-related terms, given textual lyrics, while some attributes would be relatively simple to estimate (e.g. whether or not there is profanity), others present interesting NLP challenges, such as estimating whether the lyrics are “clever” or are “meaningful” (Table 2, rows R, C, and D).

5. DISCUSSION AND CONCLUSIONS

While our analyses of playlist titles and search queries were inconclusive, we show evidence that English-speaking respondents from the U.S. and Canada clearly indicated that the voice is a salient component of music. Specifically, Spotify users were asked what they notice about music while listening. Despite the unassuming nature of the question, our results showed that the voice was indeed salient among the group of reported musical attributes. Furthermore, users ranked the voice as the second most important component to their musical preference, after emotions.

Our results have a number of implications. With regards to MIR research specifically, our results suggest that the voice and lyrics are indeed relevant attributes that warrant further study. While individuals may not necessarily want or know how to describe vocals themselves, i.e. in their playlists or search queries, surveying listeners directly does indicate that they find vocals to be important.

As such, clarifying how the voice relates to music preference is an important topic for future research.

Secondly, users indicated that the ability of a song to evoke emotions was the most important factor. This confirms findings in prior research of the relevance of emotional content in music, and how it is linked to musical preference, e.g. [10]. Therefore, examining how music affects the emotions of listeners remains an important theme. Interestingly, while genre was the most frequent term used to label playlists or search for music, respondents did not rank the specific genre as important relative to the other attributes. Understanding why this is the case warrants further study.

More relevant to our hypothesis, is that the vocals and the lyrics of a song were ranked second and third by respondents who were directly asked what components of music are important to their preferences. Therefore the link between emotions perceived in the voice and lyrics, and the emotions felt in listeners, is very relevant to questions of music preference. Clarification of these links was out of scope in these studies, and could be addressed in future research.

Lastly, we show the relevance of explicitly collected data that might guide future research. While we showed inconclusive findings regarding the prevalence of vocals in implicit data, we did show that the unique effect of vocals on music preference may be observed using survey data. As such, explicit data-gathering techniques often found in the social sciences, as well as collaborations with social scientists, may be of great use to MIR researchers.

6. REFERENCES

- [1] Jorge L Armony, William Aubé, Arafat Angulo-Perkins, Isabelle Peretz, and Luis Concha. The specificity of neural responses to music and their relation to voice processing: An fmri-adaptation study. *Neuroscience letters*, 593:35–39, 2015.
- [2] Pascal Belin, Robert J Zatorre, Philippe Lafaille, Pierre Ahad, and Bruce Pike. Voice-selective areas in human auditory cortex. *Nature*, 403(6767):309, 2000.
- [3] Jean C de Borda. Mémoire sur les élections au scrutin. *Histoire de l'Academie Royale des Sciences*, 1781.
- [4] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622. ACM, 2001.
- [5] Sean Hutchins and Sylvain Moreno. The linked dual representation model of vocal perception and production. *Frontiers in psychology*, 4:825, 2013.
- [6] Sean Michael Hutchins and Isabelle Peretz. A frog in your throat or in your ear? searching for the causes of poor singing. *Journal of Experimental Psychology: General*, 141(1):76, 2012.
- [7] Karim M Ibrahim, David Grunberg, Kat Agres, Chitralekha Gupta, and Ye Wang. Intelligibility of sung lyrics: A pilot study. International Society for Music Information Retrieval Conference, 2017.
- [8] Patrik N. Juslin and Petri Laukka. Communication of emotions in vocal expression and musical performance: Different channels, same code? *Psychological Bulletin*, 129:770–814, 2003.
- [9] Raivo Kolde, Sven Laur, Priit Adler, and Jaak Vilo. Robust rank aggregation for gene list integration and meta-analysis. *Bioinformatics*, 28(4):573–580, 2012.
- [10] Carol Lynne Krumhansl. Listening niches across a century of popular music. *Frontiers in psychology*, 8:431, 2017.
- [11] Anna M Kruspe and IDMT Fraunhofer. Retrieval of textual song lyrics from sung inputs. In *INTER-SPEECH*, pages 2140–2144, 2016.
- [12] Micheline Lesaffre, Liesbeth De Voogdt, Marc Leman, Bernard De Baets, Hans De Meyer, and Jean-Pierre Martens. How potential users of music search and retrieval systems describe the semantic quality of music. *Journal of the Association for Information Science and Technology*, 59(5):695–707, 2008.
- [13] Matt McVicar, Daniel PW Ellis, and Masataka Goto. Leveraging repetition for improved automatic lyric transcription in popular music. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3117–3121. IEEE, 2014.
- [14] George A Miller. A psychological method to investigate verbal concepts. *Journal of mathematical psychology*, 6(2):169–191, 1969.
- [15] Stephen W Porges. The polyvagal theory: phylogenetic substrates of a social nervous system. *International Journal of Psychophysiology*, 42(2):123–146, 2001.
- [16] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.
- [17] Michael W Weiss, E Glenn Schellenberg, Sandra E Trehub, and Emily J Dawber. Enhanced processing of vocal melodies in childhood. *Developmental Psychology*, 51(3):370, 2015.
- [18] Michael W Weiss, Sandra E Trehub, and E Glenn Schellenberg. Something in the way she sings: Enhanced memory for vocal melodies. *Psychological Science*, 23(10):1074–1078, 2012.

VOCAL MELODY EXTRACTION WITH SEMANTIC SEGMENTATION AND AUDIO-SYMBOLIC DOMAIN TRANSFER LEARNING

Wei-Tsung Lu and Li Su

Institute of Information Science, Academia Sinica

s603122001@gmail.com, lisu@iis.sinica.edu.tw

ABSTRACT

The melody extraction problem is analogue to semantic segmentation on a time-frequency image, in which every pixel on the image is classified as a part of a melody object or not. Such an approach can benefit from a signal processing method that helps to enhance the true pitch contours on an image, and, a music language model with structural information on large-scale symbolic music data to be transfer into an audio-based model. In this paper, we propose a novel melody extraction system, using a deep convolutional neural network (DCNN) with dilated convolution as the semantic segmentation tool. The candidate pitch contours on the time-frequency image are enhanced by combining the spectrogram and cepstral-based features. Moreover, an adaptive progressive neural network is employed to transfer the semantic segmentation model in the symbolic domain to the one in the audio domain. This paper makes an attempt to bridge the semantic gaps between signal-level features and perceived melodies, and between symbolic data and audio data. Experiments show competitive accuracy of the proposed method on various datasets.

1. INTRODUCTION

Melody extraction of polyphonic music has been accounted a key towards bridging the semantic gap in music processing, as melody is an intermediate object that correlates to both low-level signal attributes such as pitch and high-level semantics, i.e. the difference between melody and accompaniment, of music [3, 12, 29]. However, it is challenging because the notion of melody is complicated by two levels of information extraction and data modalities. For information extraction, both pitch detection and *semantic segmentation* levels are required to specify the position and shape of a melody out of other pitch contours in a time-frequency representation. As to data modalities, the problem arises from the difference of melody-related features between the *composed* data (e.g., symbolic data such as MIDI) and the *performed* data (e.g., audio data): the former provides structural information such

as voiced/unvoiced segments and chord/non-chord notes, while the latter provides interpretational information such as sliding and vibrato. Both kinds of information are essential for accurately identifying the melody pitch contour.

We perform vocal melody extraction using semantic segmentation techniques. Semantic segmentation partitions an image into semantically meaningful objects with precise boundaries. Rendered as a pixel-wise classification problem and able to be implemented by an encoder-decoder network with 2-D convolutional feature mappings, it brings great success in computer vision [6, 7, 14, 25]. Semantic segmentation also makes a breakthrough in solving the source separation problem in music processing [17], which analogously needs to resolve components coexisting in a time-frequency image. In this work, a deep convolutional neural network (DCNN) is adopted with dilated convolution for semantic segmentation as it achieves better performance in multi-resolution images.

To fully utilize the advance of semantic segmentation in vocal melody extraction, we further attend to the aforementioned issues, pitch detection and multiple data modalities, both of which are absent from typical image-based semantic segmentation. For pitch detection, we notice that when performing melody extraction with semantic segmentation, the spectrogram is usually suboptimal since it captures the harmonic peaks and information unrelated to the melody, which accounts for one of the major errors among all the melody extraction methods. This issue is addressed by modifying the spectrogram with cepstral-features, which results in a novel time-frequency representation that enhances the true pitch contour while also suppresses harmonic contours [26, 32].

The modality difference between symbolic and audio data is relatively less noticed in melody extraction. We address this issue with transfer learning: we first train a melody extraction model with symbolic data, and the model parameters are then reused in the vocal melody extraction model trained with audio data. In this way, the symbolic-based model assists in music language modeling that audio-based models may fall short of. Incorporating symbolic music data is of great potential to mitigate the data scarcity problem, since building a symbolic dataset with melody annotations is much easier than building an audio one, and it is also very straightforward to perform data augmentation on symbolic data. In this work, we adopt the *progressive neural network* (PNN) [1], a network structure providing cross-domain network parameter shar-



© Wei-Tsung Lu and Li Su. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Wei-Tsung Lu and Li Su. "Vocal melody extraction with semantic segmentation and audio-symbolic domain transfer learning", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

ing to accomplish symbolic-audio transfer learning task.

To sum up, this paper attempt to apply image semantic segmentation to vocal melody extraction, forming a systematic method to perform singing voice activity detection, pitch detection and melody extraction all at the same time. This segmentation method gives competitive results on pitch accuracy, and even works unprecedentedly well on singing voice activity detection compared to other deep-learning-based methods. With the integration with the PNN, we leverage large-scale symbolic data to train the model, and attain similar performance to the segmentation method with less training time.

2. RELATED WORK

Melody extraction of polyphonic music has been widely investigated with various signal processing and machine approaches. Recent works using convolution neural networks (CNNs) or recurrent neural networks (RNNs) are mostly classification-based, where the output is the frame-level likelihood score of every pitch at a time instance [4, 22, 27, 30, 37]. [2] adopts a fully convolution neural network and output a salience representation at the song level. Advanced semantic segmentation networks such as the U-net [28] have been utilized in source separation [17] and shows high potential in melody extraction.

Most of the melody extraction studies focus on the signal processing level, possibly because signal-level characteristics such as slides and vibrato are still the principal factors in recognizing a melody contour. In contrast, melody extraction on symbolic data is rarely discussed in the literature. Although not the main topic of this work, we manage to pose the problem of *symbolic melody extraction* and emphasize its importance in music language modeling for cross-domain transfer learning.

Previous works on transfer learning for music information retrieval mostly aim under the same type of input data representation [8, 13]. Contrarily, transfer learning across the data from different *domains*, such as adapting a model learned from symbolic data to another learned from audio data, is relatively less discussed. Previous works dealing with cross-domain data mainly focus exploring audio-to-MIDI or audio-to-sheet correspondence [10, 11].

3. METHOD

An overview of the proposed model is shown in Fig.1. The model contains a feature extractor which computes the audio data representation and a PNN which consists of two segmentation models, with one trained on the symbolic data and the other on the audio data. The filter is for dimension reduction of the audio representation to fit the symbolic segmentation model in the PNN. Details of the model are discussed below.

3.1 Audio data representation

In music processing, designing a data representation suitable for the machine learning models to better identify and capture the information of interest can help significantly

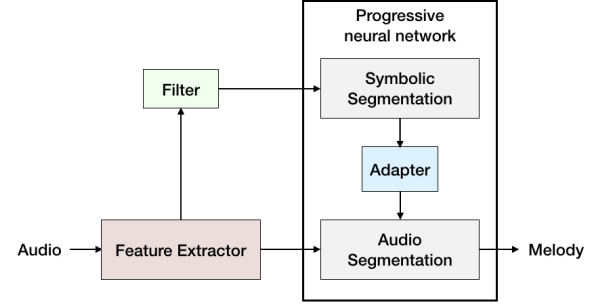


Figure 1: The system diagram of the proposed method.

improve the performance [18]. In the task of pitch detection in polyphonic music, related methods include the feature scaling [18], the harmonic constant-Q transform (HCQT) that combines the CQTs based on different octave numbers [2], the combined frequency and periodicity (CFP) representation that intergrates a temporal or spectral representation with its Fourier dual [26, 32, 34], and others. All of these methods are designed to emphasize the saliency of pitch contours in the music signal.

We adopt the data representation used in [33], which has been shown effective in enhancing the true pitch components of polyphonic signals. The adopted data representation is essentially the product of a *generalized cepstrum* (GC), a classical time-based pitch detection function [16, 20, 21, 35, 36], and a *generalized cepstrum of spectrum* (GCoS), a modified spectrum lying in the frequency domain [32]. The GC and GCoS are complementary: a GCoS reveals the presence of a pitch object by its fundamental frequency (f_0) and harmonics (nf_0), while a GC reveal it by its f_0 and sub-harmonics (f_0/n) [26, 32, 34]. By simply multiplying GC by GCoS, we effectively suppress the harmonic and sub-harmonic peaks, and at the same time localize a pitch object.

The GC and GCoS are both computed by the discrete Fourier transform (DFT) and nonlinear activation functions. Consider an input signal $\mathbf{x} := \mathbf{x}[n]$ where n is the index of time. Let the magnitude of the short-time Fourier transform (STFT) of \mathbf{x} be \mathbf{X} . Given an N -point DFT matrix \mathbf{F} , high-pass filters \mathbf{W}_f and \mathbf{W}_t for eliminating the DC terms, and activation functions σ_i , the power-scaled spectrogram, GC and GCoS are represented as:

$$\mathbf{Z}_S[k, n] := \sigma_0(\mathbf{W}_f \mathbf{X}), \quad (1)$$

$$\mathbf{Z}_{GC}[q, n] := \sigma_1(\mathbf{W}_t \mathbf{F}^{-1} \mathbf{Z}_S), \quad (2)$$

$$\mathbf{Z}_{GCoS}[k, n] := \sigma_2(\mathbf{W}_f \mathbf{F} \mathbf{Z}_{GC}), \quad (3)$$

$$\sigma_i(\mathbf{Z}) = |\text{relu}(\mathbf{Z})|^{\gamma_i}, \quad i = 0, 1, 2 \quad (4)$$

where $\text{relu}(\cdot)$ represents a rectified linear unit, $|\cdot|^{\gamma_0}$ is an element-wise root function, and we choose $(\gamma_0, \gamma_1, \gamma_2) = (0.24, 0.6, 1)$ for a feature scaling in the power scale [32].

Besides, to fit the perceptive scale of musical pitches, \mathbf{Z}_{GC} and \mathbf{Z}_{GCoS} are mapped onto the log-frequency scale, by $88 * 4 = 352$ triangular filters ranging from 27.5 Hz

(A0) to 4487 Hz , with 48 bands per octave. The GC and GCoS after the filterbank are then both on the pitch scale, as denoted by $\tilde{\mathbf{Z}}_{GC}$ and $\tilde{\mathbf{Z}}_{GCoS}$. The final 2-D data representation for semantic segmentation is

$$\mathbf{C}[p, n] = \tilde{\mathbf{Z}}_{GC}[p, n] \tilde{\mathbf{Z}}_{GCoS}[p, n], \quad (5)$$

where p is the index on the log-frequency scale. The audio files are resampled at 16 kHz and merged into one mono channel. Data representations are computed with a Hann window of 2048 samples. The hop size is 320 samples, and therefore the time step is 20ms. The upper two subplots of Figure 4 illustrate a comparison between the spectrogram and \mathbf{C} . We can observe that with the aid of cepstral feature, the unwanted harmonic peaks are highly suppressed in \mathbf{C} .

3.2 Semantic segmentation

The proposed segmentation model for vocal melody extraction is mainly based on the DeepLabV3 and its improved version, DeepLabV3+ [6, 7], which are the state-of-the-art models for semantic segmentation tasks. The model is a fully convolution neural network with an encoder-decoder architecture. The encoder is implemented by a ResNet [15], followed by an atrous spatial pyramid pooling process, and a decoder implemented by stacks of decoder blocks, as shown in Figure 2.

One major utility in DeepLabV3 is the use of dilated convolution, which can be represented as a generalized version of the standard convolution as follows:

$$\mathbf{y}[i] = \sum_k \mathbf{x}[i + r \cdot k] \mathbf{w}[k] \quad (6)$$

where \mathbf{x} and \mathbf{y} denotes the input and output 2-D feature maps, respectively, \mathbf{w} is the convolution filter and i indicates the locations on the feature maps. The number r is the dilated rate which determines the stride with which the input are sampled and standard convolution is a special case when $r = 1$. To capture the context in different ranges, one can apply dilated convolution with different values of r on the same input feature map parallelly, called Atrous Spatial Pyramid Pooling (ASPP) in [6]. The outputs of these parallel convolution operations are then concatenated to provide information collected from various scales, as shown in Figure 2c.

Different from normal image segmentation task that target objects usually holds certain area compared to the whole image, the melody part of music occupies only a small portion and appears as thin lines when visualized in a 2-D image. To overcome this difficulty, We proposed two modifications to improve the performance of the model.

First, the decoder module in DeepLabV3, which is originally an up-sampling operation, is replaced by stacks of convolution and transpose convolution layers for fine-grained outputs. It is shown in [7] that by doing this, the small and detailed objects in an image can be better recognized. Also, better performance is achieved by introducing the U-net [28] structure, which lets the output from each layer of the encoder be concatenated to the corresponding block of the decoder. This idea is also mentioned in [7].

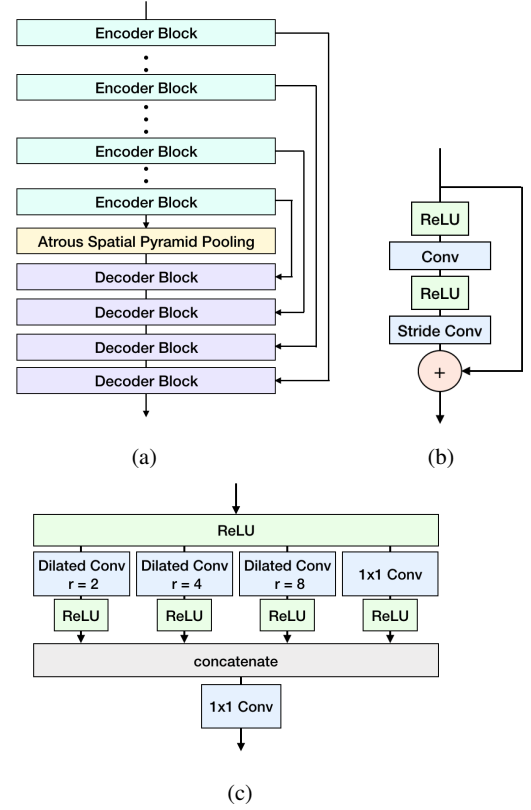


Figure 2: Model descriptions. (a) The overall structure of the segmentation model. (b) The encoder block. The stride rate in *Stride Conv* is (2,2). *Stride Conv* can be replaced with standard convolution so it allows more layers in the encoder. It can also be changed to transpose convolution with stride (2,2), so the block can serve as a decoder block. (c) The Atrous Spatial Pyramid Pooling unit.

Second, we adopt the *focal loss* [23] as the loss function for the proposed model, in order to solve the class imbalance problem, where the negative labels, i.e., the time-frequency pixels corresponding to accompaniment and silence parts, could dominate in the input feature and thus affect the performance. The focal loss is represented as:

$$\text{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t), \quad (7)$$

where p_t denotes the model's estimated probability for an input to be classified to class t , $\alpha_t \in [0, 1]$ is a weighting factor for balancing the importance of positive and negative examples and the term $(1 - p_t)^\gamma$ acts as a modulating factor with γ controlling the rate at which dominant examples are down-weighted. Following [23], we set $\alpha_t = 0.25$, $\gamma = 2$ in this work.

3.3 Domain adaptation

Most of the existing deep learning models require a large amount of training data to reach good performance. However, annotating melody pitch contours on audio data precisely is quite challenging; it is labor-intensive and also needs strong expertise in music. Recent attempts to ad-

dress the issue of data scarcity mostly focus on weakly supervised learning [17, 24, 31].

In this work, we consider the potential of domain-adaptive transfer learning, which incorporates the information in MIDI data to assist in training the audio melody extraction model. The primary motivation for using MIDI files is the capability of data augmentation: one can use MIDI files to easily create large-scale symbolic dataset with detailed and precise notations. Besides, the symbolic data also present some musical characteristics clearer than the audio data do. This therefore gives more insights to the music language modeling, such as musical structures and phrases. Moreover, the space efficiency of symbolic data also allows more training examples than audio data given the same memory resource.

To discuss transfer learning between audio and symbolic data, we first discuss the difference in their data formats. One difference is the pitch resolution, which is 0.25 semitones in the audio data (i.e., 48 bins per octave), and 1 semitone in the symbolic data; this results in the difference of dimension between the audio and the symbolic data. As for the time resolution, there are some more flexible ways to define it. Therefore, we consider two types of time resolution for the symbolic data: the first is *time-based* resolution with its unit length in time (e.g., 20 ms), and the second is *note-based* resolution with its unit length in note name (e.g., a 32nd note). Both the symbolic and audio data can be represented in *time-based* resolution. Symbolic data can also be represented in a more musically informative *note-based* resolution since obtaining beat and tempo information in symbolic data is more straightforward.

To achieve domain-adaptive transfer learning for two different domains, we adopt the progressive neural network (PNN) [1], in which an *adapter* network (see Figure 3) is designed to make one network connected to another in different domains, regardless of the difference in data dimension. In the general scenario of PNN, multiple networks trained on various tasks are connected layer-to-layer in parallel through the adapters, so the trained networks can transfer the previously learned knowledge into a new task and to accelerate the training speed or to improve the performance of the new task.

In our melody extraction method, we first trained a segmentation model using the symbolic dataset. We connect the symbolic segmentation model to another segmentation model, and the latter model is then trained on the audio dataset, with the parameters in the symbolic segmentation model frozen. In the testing phase, the input audio representation is fed into both of the segmentation models. To make the dimension of audio representation match the symbolic segmentation model, a triangular filterbank is used to map the pitch resolution from 0.25 to 1 semitone, as illustrated in Figure 1.

The adapter between two models in the PNN is illustrated in Figure 3. It modifies the dimension of the inter-layer outputs and make such information be propagated ahead. In the proposed method, transpose convolution layers are adopted for the adapter networks, since transpose

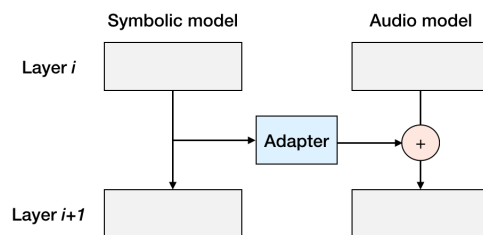


Figure 3: The connection between the two networks in the proposed model. The parameter of the i -th layer in the symbolic model is first fed into the adapter, and then connected to the $(i + 1)$ -th layer of the audio model with an addition operation.

convolution can up-sample the output of the symbolic representation (with lower pitch resolution) in order to fit the audio representation (with higher pitch resolution).

3.4 Inference

Since the segmentation model only allows a limited range of input at one time, to perform melody extraction on a given score, we slide a window along the score and then superpose all the resulting matrices. The analysis window with a fixed dimension is shifted from one time-step to another. As to the beginning and ending time, we pad the score with zeros for it captures the process in which information feeds only the last column then gradually filling up all the columns in the beginning, and gradually leaving the window column by column at the end. After the process above, the segmentation output is a superposed image representing the salience of vocal melody in the time-frequency plane. We then find the max value for each column of the image and set all the other elements to zero, i.e., unvoiced. Finally, the elements smaller than the average of each column's maximum are also set to zero, and the remaining non-zero elements is considered as voiced.

3.5 Implementation details

The models are implemented using the Keras [9] library with tensorflow as the back end. The width of the input window equals 128 timesteps, and for computational convenience, we pad the dimension of pitch from 88 to 128, and 352 to 384 for the symbolic and audio data, so the input dimension will be $(128, 128, 1)$ and $(128, 384, 1)$ for the symbolic and audio model, respectively. As shown in Fig.1, the input feature will first be passed into a 29-layer encoder based on Resnet. Then, the output from the encoder which is 16 times smaller than the original input will be fed into the ASPP unit. Finally, a decoder which contains 4 decoder blocks will up-sample the dense features to the original shape by transpose convolutional layers with strides equal $(2, 2)$. The output dimension will be $(128, 128, 2)$ and $(128, 384, 2)$ for the symbolic and audio model, respectively, with the first channel indicating the presence melody and the other is for non-melody.

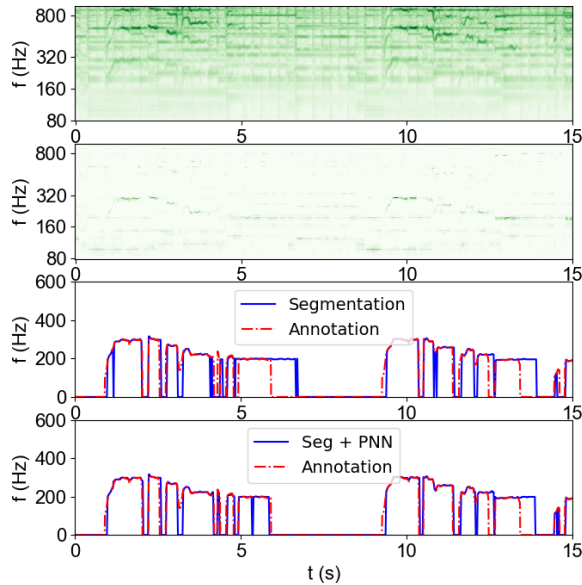


Figure 4: Data representation and melody extraction results of the first 15s of 'train06.wav' in MIREX005 as input. From top to bottom: power-scale spectrogram, data representation C , the result using segmentation, and the result using segmentation and note-based PNN.

The superposition in the inference process is performed on the first channel. To implement the PNN, two segmentation networks with same structure are connected using the adapters which is composed of transpose convolution layer. These connections happen in layers with dimension changing. Batch normalizations are applied after each activations, and a dropout rate of 30% is added after the batch normalizations. ADAM [19] is used for optimization. Source codes can be found at <https://github.com/s603122001/Vocal-Melody-Extraction>.

4. EXPERIMENT

4.1 Data

The training data for the audio comes from two datasets, one is the MIR1K¹, which contains 1000 Chinese karaoke clips, another is MedleyDB [5], where 48 songs with vocal tracks are included. The total dataset contains about 3 hours of audio and without data augmentation.

A MIDI corpus contains 600 folk songs with a melody track is used as the training data for the symbolic model.² In the training process, we perform data augmentation, by pitch-shifting each song up and down by at most 6 semi-tones in order to cover all possible keys. In addition, half of the pieces in the dataset are modified by shifting the melody by one octave down. As a result, we produce 7673 pieces of symbolic training data. The pieces in the dataset are represented in two different formats. One is the *time-based* with 20 ms length in each time step and the other is

the *note-based* that each time step equals a thirty-second note. Due to limited computational resources, we only use 2048 pieces when training the time-based model since time-based data is space consuming.

The testing data are from three datasets: ADC2004, MIREX05,³ and MedleyDB. As the proposed model is trained solely for singing voice melody, we follow [22] and select only samples having melody sung by human voice from ADC2004 and MIREX05. As a result, 12 clips in ADC2004 and 9 clips in MIREX05 are selected. To obtain the annotation of singing voice in medleyDB, 12 songs having singing voice included in their 'MELODY2' annotations are selected. The vocal melody labels are obtained from the MELODY2 annotations occurring in the intervals labeled by 'female singer' or 'male singer'. These 12 songs are not included in the training data.

4.2 Experiment setting

To assess the performance of semantic segmentation and the effects of transfer learning on vocal melody extraction, we experiment on the following three different settings:

1) *Segmentation*: using simply the audio-level semantic segmentation model. This audio-only semantic segmentation model is trained on the MIR1K dataset.

2) *Segmentation with note-based progressive neural network (Seg + note PNN)*: using both the audio-level and symbolic-level segmentation models. The symbolic segmentation model is first trained using the note-based symbolic dataset, then this model is incorporated into the training stage of the audio segmentation model with the PNN.

3) *Segmentation with time-based progressive neural network (Seg + time PNN)*: similar to 2), while the symbolic model is trained with the time-based symbolic dataset.

We compare the above-mentioned models with three baseline methods in deep learning approaches: the multi-column DNN (MCDNN) [22], the patch-based CNN (patch-CNN) [33], and the deep salience map (DSM), for which on-line source code with the vocal option is available [2]. Since the detection results of DSM are sensitive to the thresholding parameter, the parameter is tuned from 0 to 0.9 for all datasets to find the optimal value for better comparison. The resulting optimal threshold $th=0.1$ is used in the experiment.

The performance metrics include overall accuracy (OA), raw pitch accuracy (RPA), raw chroma accuracy (RCA), voice recall (VR) and voice false alarm (VFA);⁴ all these metrics are computed from the `mir_eval` standard with the tolerance of pitch detection being 50 cents.

4.3 Result

Table 1 lists the performance metrics of all the proposed methods together with the baselines on the three testing datasets. Among the three proposed models, *Segmentation* outperforms the other two PNN-based models in terms of OA for all datasets except MedleyDB, where *Segmentation* performs on par with *Seg + note PNN*. Through the

¹ <https://sites.google.com/site/unvoicedsoundseparation/mir-1k>

² <https://goo.gl/aPgZrW>

³ <https://labrosa.ee.columbia.edu/projects/melody/>

⁴ http://www.music-ir.org/mirex/wiki/2016:Audio_Melody_Extraction

Method	OA	RPA	RCA	VR	VFA
Segmentation	74.9	71.7	74.8	73.8	3.0
Seg + note PNN	73.5	70.2	73.2	72.2	3.1
Seg + time PNN	73.2	70.4	72.9	73.2	5.4
MCDNN [22]	73.1	75.8	78.3	88.9	41.2
Patch-CNN [33]	72.4	74.7	75.7	90.1	41.3
DSM [2]	70.8	77.1	78.8	92.9	50.5

(a) ADC2004 (vocal)

Method	OA	RPA	RCA	VR	VFA
Segmentation	85.8	82.2	82.9	87.3	7.9
Seg + note PNN	84.5	79.6	80.3	84.7	6.9
Seg + time PNN	84.8	82.3	83.0	87.3	9.9
MCDNN	68.4	76.3	77.4	87.0	49.0
Patch-CNN	74.4	83.1	83.5	95.1	41.1
DSM	69.6	76.3	77.3	93.6	42.8

(b) MIREX2005 (vocal)

Method	OA	RPA	RCA	VR	VFA
Segmentation	70.0	68.3	70.0	77.9	22.4
Seg + note PNN	70.0	67.1	68.7	77.0	21.5
Seg + time PNN	69.1	67.4	69.0	78.7	23.6
Patch-CNN	55.2	59.7	63.8	78.4	55.1
DSM	66.2	72.0	74.8	88.4	48.7

(c) MedleyDB (vocal)

Table 1: Vocal melody extraction results of the proposed methods and other methods on various datasets. The proposed methods are: segmentation, segmentation with note-based progressive neural network (Seg + note PNN), and segmentation with time-based progressive neural network (Seg + time PNN).

melody extraction accuracies of the segmentation model are not improved by introducing the PNN structure, there is still a notable improvement when comparing training efficiency. In fact, it takes 6 epochs for *Segmentation* to converge, but *Seg + note PNN* reach similar performance with only 2 epochs of training. Therefore, introducing the PNN improves the training speed.

One reason why PNN does not improve the accuracy is related to the symbolic dataset we are using: the symbolic data contains only one style of music and turns out to be of low diversity. Another reason is the lack of *intensity* labels in symbolic data. Our pilot study indicated that a segmentation model trained on symbolic data may result in high RCA and RPA but also relatively high VFA. However, a segmentation model trained on the audio data gives inverse results, with low VFA, as shown here. This might have something to do with the sound intensity in the audio signal, which is an important sign for to determine the present of melody. However, our symbolic data do not have such labels on intensity. Model training with a larger symbolic music dataset with higher diversity and with MIDI velocity labels are for future investigation.

The two PNN-based methods, *Seg + note PNN* and *Seg + time PNN*, achieve similar OA, while the former model

has lower VFA. This implies that the performance of the symbolic model trained with note-based symbolic data is better than training with time-based data. One reason may be that compiling symbolic data in time-based resolution may result in the ambiguity of musical information; in time-based data, the same type of note may have different lengths in time due to different tempi among the music pieces. This could affect the model capability in learning the musical structure.

Comparing the proposed *Segmentation* model to the baseline methods, we observe that *Segmentation* outperforms all of them in terms of OA. Particularly, in MIREX2005, *Segmentation* achieves an OA at 85.8%, a high accuracy outperforming DSM by 16.2%, patch-CNN by 11.4% and MCDNN by 17.4%. In other two datasets, *Segmentation* also outperforms other methods by around 1 ~ 4% in terms of OA. These experiment results reveal the competitiveness of the proposed semantic segmentation method in audio melody extraction. On the other hand, when focusing on the pitch accuracy (i.e., RPA and RCA), DSM is still competitive among all.

The high OA of *Segmentation* is mainly resulted from the excellent performance of VFA with the semantic segmentation approach. Among all methods and datasets, the proposed methods significantly outperform the baseline methods by a 20-40% reduction in VFA. In ADC2004, *Segmentation* further achieves a low VFA of 3.0%. This implies that the proposed melody extraction method itself is highly robust to non-vocal interference, and is without the need of a voice activity detector [27]. In other words, the semantic segmentation model with fully convolutional layers itself behaves as a melody pitch classifier and a voice activity detector at the same time.

Finally, the lower two subplots in Figure 4 illustrate two melody extraction results using *Segmentation* without and with a note-based PNN. Both methods perform well in segmenting the main melody part from the representation C shown in second subplot in Figure 4. This example also demonstrates one part that using the note-based PNN does well: in the lowest subplot, the *Seg + note PNN* method well detects the unvoiced part between the 6th and the 9th second, in which the *Segmentation* method regards the extended instrument part as melody.

5. CONCLUSION

We proposed a melody extraction method utilizing the semantic segmentation model, the input combining spectral and cepstral representations, and domain-adaptive transfer learning. Experiments using a low-diversity training data indicate the competitiveness of the segmentation model with the data representation, especially in reducing voice false alarm. Incorporating large-scale symbolic data provides better efficiency and exhibits potential in enhancing contextual information. Future work will focus on the improvement of domain adaption. Note-level segmentation can be considered as a future work as it is also feasible applying symbolic-audio transfer learning and would also benefit the melody extraction task.

6. ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work is partially supported by MOST Taiwan, under the contract MOST 106-2218-E-001-003-MY3.

7. REFERENCES

- [1] R. Andrei A., R. Neil C., D. Guillaume, S. Hubert, K. James, K. Koray, P. Razvan, and H. Raia. Progressive neural networks. *eprint arXiv:1606.04671*, 2016.
- [2] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello. Deep salience representations for f_0 estimation in polyphonic music. In *18th Int. Soc. for Music Info. Retrieval Conf.*, Suzhou, China, Oct. 2017.
- [3] R. M. Bittner, J. Salamon, J. J. Bosch, and J. P. Bello. Pitch contours as a mid-level representation for music informatics. In *Audio Engineering Society Conference: 2017 AES International Conference on Semantic Audio*. Audio Engineering Society, 2017.
- [4] R. M. Bittner, J. Salamon, S. Essid, and J. P. Bello. Melody extraction by contour classification. In *Proc. ISMIR*, pages 500–506, 2015.
- [5] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello. Medleydb: A multitrack dataset for annotation-intensive mir research. In *Proc. ISMIR*, volume 14, pages 155–160, 2014.
- [6] L.-C. Chen, P. George, S. Florian, and A. Hartwig. Rethinking atrous convolution for semantic image segmentation. *eprint arXiv:1706.05587*, 2017.
- [7] L.-C. Chen, Y. Zhu, P. George, S. Florian, and A. Hartwig. Encoder-decoder with atrous separable convolution for semantic image segmentation. *eprint arXiv:1802.02611*, 2018.
- [8] K. Choi, G. Fazekas, M. Sandler, and K. Cho. Transfer learning for music classification and regression tasks. *arXiv preprint arXiv:1703.09179*, 2017.
- [9] F. Chollet et al. Keras. <https://github.com/fchollet/keras>, 2015.
- [10] R. B. Dannenberg and C. Raphael. Music score alignment and computer accompaniment. *Communications of the ACM*, 49(8):38–43, 2006.
- [11] M. Dorfer, A. Arzt, and G. Widmer. Learning audio-sheet music correspondences for score identification and offline alignment. In *18th Int. Soc. for Music Info. Retrieval Conf.*, Oct.
- [12] M. Goto. A predominant-F0 estimation method for polyphonic musical audio signals. In *Proc. Int. Cong. Acoustics*, pages 1085–1088, 2004.
- [13] P. Hamel, M. Davies, K. Yoshii, and M. Goto. Transfer learning in mir: Sharing learned latent representations for music audio classification and similarity. 2013.
- [14] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.
- [15] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.
- [16] H. Indefrey, W. Hess, and G. Seeser. Design and evaluation of double-transform pitch determination algorithms with nonlinear distortion in the frequency domain-preliminary results. In *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, pages 415–418, 1985.
- [17] A. Jansson, E. Humphrey, N. Montecchio, R. M. Bittner, A. Kumar, and T. Weyde. Singing voice separation with deep u-net convolutional networks. In *18th Int. Soc. for Music Info. Retrieval Conf.*, Suzhou, China, Oct. 2017.
- [18] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. Widmer. On the potential of simple frame-wise approaches to piano transcription. *arXiv preprint arXiv:1612.05153*, 2016.
- [19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2014.
- [20] A. Klapuri. Multipitch analysis of polyphonic music and speech signals using an auditory model. *IEEE Trans. Audio, Speech, Lang. Proc.*, 16(2):255–266, 2008.
- [21] T. Kobayashi and S. Imai. Spectral analysis using generalized cepstrum. *IEEE Trans. Acoust., Speech, Signal Proc.*, 32(5):1087–1089, 1984.
- [22] S. Kum, C. Oh, and J. Nam. Melody extraction on vocal segments using multi-column deep neural networks. In *Proc. ISMIR*, pages 819–825, 2016.
- [23] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *eprint arXiv:1708.02002*, 2017.
- [24] J.-Y. Liu and Y.-H. Yang. Event localization in music auto-tagging. In *Proc. ACM Multimedia*, pages 1048–1057. ACM, 2016.
- [25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
- [26] G. Peeters. Music pitch representation by periodicity measures based on combined temporal and spectral representations. In *Proc. IEEE ICASSP*, 2006.
- [27] F. Rigaud and M. Radenen. Singing voice melody transcription using deep neural networks. In *ISMIR*, pages 737–743, 2016.

- [28] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- [29] J. Salamon and E. Gómez. Melody extraction from polyphonic music audio. *Music Information Retrieval Evaluation eXchange (MIREX)*, 2010.
- [30] J. Salamon and E. Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, 2012.
- [31] J. Schlüter. Learning to pinpoint singing voice from weakly labeled examples. In *ISMIR*, pages 44–50, 2016.
- [32] L. Su. Between homomorphic signal processing and deep neural networks: Constructing deep algorithms for polyphonic music transcription. In *Asia Pacific Signal and Infor. Proc. Asso. Annual Summit and Conf. (APSIPA ASC)*, 2017.
- [33] L. Su. Vocal melody extraction using patch-based cnn. In *Proc. ICASSP*, 2018.
- [34] L. Su and Y.-H. Yang. Combining spectral and temporal representations for multipitch estimation of polyphonic music. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 23(10):1600–1612, 2015.
- [35] K. Tokuda, T. Kobayashi, T. Masuko, and S. Imai. Mel-generalized cepstral analysis: a unified approach to speech spectral estimation. In *Proc. Int. Conf. Spoken Language Processing*, 1994.
- [36] T. Tolonen and M. Karjalainen. A computationally efficient multipitch analysis model. *IEEE Speech Audio Processing*, 8(6):708–716, 2000.
- [37] P. Verma and R. W. Schafer. Frequency estimation from waveforms using multi-layered neural networks. In *INTERSPEECH*, pages 2165–2169, 2016.

EMPIRICALLY WEIGHING THE IMPORTANCE OF DECISION FACTORS WHEN SELECTING MUSIC TO SING

Michael Mustaine¹ Karim M. Ibrahim¹
Chitralekha Gupta^{1,2} Ye Wang¹

¹ School of Computing, National University of Singapore, Singapore

² NUS Graduate School for Integrative Sciences and Engineering,
National University of Singapore, Singapore

baronemda@gmail.com, wangye@comp.nus.edu.sg

ABSTRACT

Although music cognition and music information retrieval have many common areas of research interest, relatively little work utilizes a combination of signal- and human-centric approaches when assessing complex cognitive phenomena. This work explores the importance of four cognitive decision-making factors (familiarity, genre preference, ease of vocal reproducibility, and overall preference) influence in the perception of “singability”, how attractive a song is to sing. In Experiment One, we develop a model to validate and empirically determine to what degree these factors are important when evaluating its singability. Results indicate that evaluations of how these four factors impact singability strongly correlate with pairwise evaluations ($\rho = 0.692, p < 0.0001$), supporting the notion that singability is a measurable cognitive process. Experiment Two examines the degree to which timbral and rhythmic features contribute to singability. Regression and random forest analysis find that some selected features are more significant than others. We discuss the method we use to empirically assess the complex decisions, and provide a preliminary exploration regarding what acoustic features may motivate these choices.

1. INTRODUCTION

A fundamental task of MIR is to develop of acoustic feature extractors that capture unique characteristics from a recorded piece of sound. However, some acoustic features may not be wholly represented in the acoustic signal, and MIR has been criticized for failing to model analysis based on psychological research [3]. For example, “danceability” - the perceptual experience of grooviness [23,48] - is a feature available in signal processing pack-

ages [9, 10], and open-access APIs¹ using a combination of beat salience and consistency [36]. However, based solely on these acoustic properties the most danceable song would be closer to a steady, metronomic pulse, which clearly does not capture the perceptual nuances of what makes music danceable [14]. The inclusion of psychological acoustic features using signal-only analysis is surprising, given that music is a dynamic system influenced by cognitive [20], cultural, market, and political forces [8]. Despite this knowledge, research is relatively sparse as to how, or to what degree, specific acoustic features influence musical preference. Part of the scarcity may be due to the relative difficulty in quantifying the influence of important psychological features empirically. This work examines the extent to which a cognitive psychology, signal processing, machine learning, and economic decision-making can be used to investigate a previously unexplored psychological perception of “singability”: the degree to which a song is attractive to sing. To our knowledge, no empirical study has been conducted which explores whether a feature such as singability can be extracted from a piece of music.

Determining a complex psychological process and decision making strategy like singability is a difficult task. To start, it is intuitively difficult to quantify such a subjective multiple criterion choice in a controlled, scientific manner. Because singability will likely not contain a universally agreed upon set of factors, the major challenge is defining a method that can quantify how - and to what degree - these factors should be incorporated into a model for evaluation. We first introduce some background on closely related concepts to our interpretation of singability from psychological experiments and MIR applications.

1.1 Related Work

Perhaps the most historically relevant psychological research relating to singing preference was initially proposed by Berlyne [7]. Berlyne suggests that music exhibits an inverted-U-shaped relationship for preference, influenced by novelty, complexity, and tone. This model has been replicated independently from a variety of perspectives including personality and preference research [29], and flow



© Michael Mustaine, Karim M. Ibrahim, Chitralekha Gupta, Ye Wang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Michael Mustaine, Karim M. Ibrahim, Chitralekha Gupta, Ye Wang. “Empirically Weighing the Importance of Decision Factors when Selecting Music to Sing”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ <https://developer.spotify.com/web-api/get-audio-features/>

states [12].

Prior research on singability has focused on music recommendation systems for digital karaoke applications [25, 26]; they used a competence-based evaluation, and recommended music using an individual's singing proficiency. These systems define singing preference solely on whether one can recreate the original performance [16] and fails to consider other aspects of preference such as familiarity on preference; if the ability to recreate an original version of a song is the sole criteria for determining singable tracks, a naïve extension to improve performance would be to recommend songs based on demographic features such as age, sex, and height through automated assessment of singing voice [46].

2. SINGABILITY FACTORS

We examine singability using a synthesis of multiple criterion decision making processes, acoustic feature extraction, and machine learning founded on a theoretical background of music cognition. Based on the general research discussed above, we expand the interpretation of singability from other research [25, 26] to include more factors than just the ability to reproduce the original rendition of a track. For the purpose of this work, singability is defined as a psychological process which includes how attractive a song is to sing without concern of social consternation for being unable to produce the original vocalizations. Based on this refined definition, we consider four factors which could impact singability and include: i) familiarity, ii) genre, iii) preference to listen (listenability), and iv) producibility.

To maintain a realistic scope for exploratory research, we did not include an exhaustive list of potential singability factors. These factors were selected due to their relative presence in the psychological literature. We also were interested in selecting features that would be less demanding to ask crowdsourced workers; other features we did not explore, such as the importance of lyrics or social factors, could be analysed using methodology specific to their disciplines should compelling evidence for singability be found. Next, we highlight research specific to these factors, then describe a method to quantify the prioritization of them when making a complex, multiple criterion decision.

2.1 Familiarity

Familiarity has important influences on preference formation. The mere exposure effect, a foundational psychological process [28, 49], demonstrates that increased exposure to essentially anything increases your preference for it, even when unaware of it's inclusion in your immediate environment [24]. In [32], the mere exposure effect was also found to impact music preference; multiple repetitions of unfamiliar music [28], and random tone sequences [47] increased preferences for them. A possible reason for why familiarity increases preference is because it improves ease of processing [30], impacting the complexity component

of Berlyne's optimal complexity model described in Section 1.1. The relationship between familiarity through mere exposure appears to occur early in cognitive processing - Korsakoff amnesics demonstrate increased liking to musical stimuli through increased exposure [19].

However, it is important to consider that increased familiarity does not increase preference in all cases; most people do not actively listen to extremely familiar songs such as *Twinkle, Twinkle, Little Star*. This still makes sense when considering Berlyne's optimal complexity model (Section 1.1) - extremely familiar music is too simple or not novel enough to engage. Therefore, it is hypothesized that although familiar music is important for singability, music that is too familiar will not be preferred.

2.2 Genre

Genre preference describes a specific aspect of the mere exposure effect through common acoustic features which are hallmark in the genres you typically listen to. For example, Rap music has a high degree of speech, and Metal music generally is high tempo, and with negative valence [4]. This form of familiarity is more active and personal, aligning more closely to the role that individual preference plays in exposure. Neurological evidence for an active mere exposure effect through genre has been demonstrated in brain imaging studies. Using electroencephalography, Mismatch Negativity Responses (MMNs; a spike in brainwave polarization when expectations are violated) can be elicited with tone sequences in the first few trials regardless of formal musical training [38]. In a subsequent study, authors of [39] found that MMN responses, were stronger when genre conventions were defied in a participants preferred musical style. In a study containing 17 million users from over 30 countries, users download tracks of secondary genres acoustic features similar to those of their most preferred genre [4]. For example, users who had clear preferences for Rap music preferentially downloaded tracks from other genres that contained more speech sounds. We therefore hypothesize that genre plays an important role in the selection of a preferred song to sing.

2.3 Listenability

The definition of listenability used for this work refers to how attractive a song is to listen to. Although it may be appealing to suggest that songs that are listenable are by extension singable, they must be considered mutually exclusive. Rap or Metal music for example may fit this category as the vocalizations required are not conducive for singing, but are still highly popular and can be very listenable. Furthermore, listenability is distinct from familiarity, but can be influenced by it. As suggested in Section 1.1, nursery rhymes are highly familiar, but are likely not considered highly listenable or singable by most. Highly listenable songs may also not be familiar because older tracks are played significantly less than newly released songs. Listenability may be best differentiated from familiarity in

that it can be an immediate process, requiring only a single exposure in order to be evaluated as attractive. Cognitive processing of various complex musical features such as genre [21] can happen at millisecond timescales. Listenability is considered an important factor for singability because it increases the likelihood that a song will be selected to or attended by users (thus directly influence the likelihood it will be sung in the first place) and because they are more salient in memory.

2.4 Producibility

Music cognition research has examined the distinction of singing quality; the perceptual or acoustic features that make trained singers sound better than amateurs. Quality of singing voice has been assessed with respect to full upper resonance in a singer's formant range (known as the singer's formant, a prominent spectral envelope of 3kHz) as of singing voice quality [5]. Professional singers have higher formant intensity than untrained voices; relative amplitudes of singer's formants grew as vocal intensity increased and diminished as pitch rose [35], trained voices have more energy in the formant range but not for all pitches, and males in general have higher formant intensity than females [35]. The singer's formant appears to be a particularly important property for classical operatic singers to project above the orchestra [37].

Although measures regarding whether an individual has vocal training can be assessed through the singer's formant, producibility is not contingent on these features. For example, untrained singers with self-expressed singing talent have identical pitch matching accuracies when compared to trained singers [45]. Producibility based on vocal features which indicate professional training may also not be appropriate because the correlation between genre preference and training does not align with what is popularly sung; individuals with more musical training show increased preferences for "serious" genres such as Classical and Jazz, but not other genres such as Pop [17].

3. EXPERIMENT ONE: VALIDATING SINGABILITY

To our knowledge, there is no prior work that examines whether what people think makes a song singable correlates with what they actually select in natural settings. For example, [41] instructed professional musicians to evaluate recordings of top-three placing performances from piano competitions under three conditions, recordings with: i) video only, ii) audio only, or iii) audio and video. Participants accurately ranked the video-only condition more consistently with who won the competition than in any other condition; the audio-only condition was the least consistent. This work establishes that it is possible that our impressions of what features are important in our musical preferences may not be internally consistent.

We combine a series of psychological analysis methods to establish whether singability can be consistently assessed among individuals using a set of 50 popular song

excerpts. To establish a bottom-up ground truth, a forced alternative choice (FAC) experiment is conducted with pairs of songs; a complex decision-making model known as Analytic Hierarchical Process (AHP) [33] is used to determine top-down impressions. We then rank songs based on their assessed singability using both methods (FAC and AHP) to determine whether there is consistency between what we think is singable, and what our decisions end up inevitably being. An additional benefit of using AHP is that it can weigh the degree to which each of the four features described above contributes to an individual's choice to sing a song. Because AHP is less commonly used, we briefly describe AHP and how it is conducted prior to reporting experimental structure.

3.1 Analytic Hierarchical Process

AHP is a technique to quantify how, and to what degree, subjective criteria influence a complex decision making task. The validity of the AHP has been examined extensively [44], and has been used within government, business, and healthcare [42]. Figure 1 illustrates the final importance values for each factor and are now described. Determining singability using AHP involves breaking down the decision problem into a set of global priorities (green boxes). Global priorities are a set of general factors that are suspected to influence the decision-making process. After global priorities are determined, levels within each priority (local priorities; blue boxes) are established. Once priorities have been established, the importance of each factor can be systematically evaluated to determine their contribution to the final decision. Decision makers weigh the importance of each of these priorities using multiple pairwise comparisons, and require the decision maker to evaluate every priority relative to another. For instance, a worker is asked "how important was it that the vocals were easy to reproduce, as opposed to moderately difficult". Because more than one worker answered the same question multiple times, we take the average importance value from all comparisons as the final importance value. Priorities are calculated by dividing the importance of the first comparison over the other. A pairwise comparison matrix is generated after all evaluations are made by multiplying the entries of each row and taking the n th root of the product. The roots are then summed and normalized to produce an eigenvector representing the priority importance.²

3.2 Methods

The dataset contains excerpts of 50 songs (ten songs from five genres) from the top 50 Billboard chart songs between the years of 2011-2015. Selected songs had equal numbers of male and female singers (five per sex per genre). In order to reduce high degrees of familiarity, songs from the bottom of the list were selected. 15-seconds of audio was extracted from each artist's official YouTube channel. Audio was extracted from the video as mp3 files.

²For in-depth example, see: http://rad.ihu.edu.gr/fileadmin/labsfiles/decision_support_systems/lessons/ahp/AHP_Lesson_1.pdf

A two-part online survey was crowdsourced using Amazon's Mechanical Turk.³ Although some research suggests that the quality of crowdsourced data is more diverse and at times better than data collected in traditional laboratory settings [6], additional metrics which validate or refine analysis highlighted in Section 1.1 should be considered. The first part of the experiment consisted of a series of FACs. Workers were instructed to listen to excerpts of two songs. They were asked to determine which song was more singable, listenable, and whether either of the songs were familiar. Workers repeated this paradigm for five pairs of songs in total.

After completing the FAC section, workers were then instructed to complete an AHP after briefly reflecting on the choices they made when selected between pairs of songs. Different levels of local priorities are established for each global priority. Five levels for genre were selected; Rock, Pop, Alternative, Country, and Rap music were selected; two levels for familiarity (low and high); three levels for producibility (easy, medium, hard) and; three levels for preference to listen (low, medium, high). In order to keep the task as simple as possible for workers, we reduced the number of local priorities to as little as possible - unlike producibility and listenability, only two levels were selected for familiarity because we wanted to know whether any prior knowledge of a piece would influence their choice.

Importance values were calculated by taking the average response for each priority across all respondents. Lastly, we requested workers to report only their sex - we did not collect information regarding worker age, socioeconomic status, or ethnicity. The reasoning for this was two-fold: i) Mechanical Turk demographic variability is in general more diverse than traditional laboratory data collection [6], and; ii) we were interested in establishing the general existence of a psychological perception from a rarified set of possible influencers before examining how dynamic anthropological and sociological factors modulate the preference. A benefit of using AHP is that it is a simple process to add or remove global priorities and replicate the experiment easily with new variables and interactions.

3.3 Analysis

Pairwise comparisons were conducted for all 50 songs (1225 pairs), each job instructed users to evaluate 5 pairs (245 jobs), and each job was assessed 3 times (735 surveys conducted). 88 submissions (11%) were rejected for incorrectly answering a confirmatory test question.⁴ A worker was compensated \$0.07 USD per job and could perform up to five surveys. 245 unique respondents (44% male, 56% female) completed the survey. On average, workers agreed with each other that one song was more singable than another 77.8% of the time. We examined whether individuals selected a song as more singable based on the sex of the

artist. A binomial test indicated that individuals selected same-sex singers slightly more often (53%; $p < 0.001$), though the difference was marginal.

Once AHP priorities were calculated, songs were segmented into bins for the familiarity (high and low), and listenability (high, medium, and low) categories based on the survey responses. Figure 1 represents the global and local priority values generated through the Mechanical Turk survey. Song rankings for AHP were derived for each song by producing a rank-order based off the product of local priority values for genre, listenability, and familiarity. For example, a Rock song which was in the top 50th percentile for familiarity, and the bottom 33rd percentile for listenability would receive a singability value of $0.227 * 0.613 * 0.299 = 0.0416$. Producibility was not included in the calculation because this feature is relative to an individual's skill at singing and can only be evaluated for each user, as opposed to each song. Ranks for the FAC portion of the experiment were generated by ordering the amount of times any given song within a pairwise comparison was selected by the user as more attractive to sing.

Once ranks were generated for the bottom-up (FAC), and top-down (AHP) processes, we conducted a Spearman- ρ rank correlation. Ranks derived from FAC are highly correlated with ranks derived from the AHP ($r_s = 0.691, p < 0.0001$). 47.61% of the variance in rank could be accounted for across ranked derived from FAC and AHP. Figure 2 plots the ranks derived for each song excerpt. Each song's coordinates represent the FAC derived rank (x-axis) to the AHP derived rank (y-axis). Significant Spearman- ρ correlations were also found comparing Billboard ranks to FAC ($r_s = 0.518, p < 0.001$) and AHP ($r_s = 0.540, p < 0.0001$).

3.4 Discussion

The purpose of experiment one was to derive a method that can determine whether people's heuristic impressions of preference reliably predicts their actual decisions. The highly significant correlation ($p < 0.0001$) and large effect size ($r^2 = 0.4761$), supports the hypothesis that people's top-down assessments of singability are features they actually use when making the decision. This finding is significant because it supports the notion that a less labour intensive process is needed for determining a music-cognitive process; you do not need to conduct a bottom-up comparison for the entire corpus of music to determine general preference. The results suggest that listenability is the most important feature followed by: familiarity, genre, and producibility. The importance values for most local priorities are generally intuitive; easily produced, familiar music we like to listen to are important factors we use when deciding to sing something. Rock was the most important genre (22.7% importance), followed by Pop (21.3%), Alternative (19.6%), Country (19.3%), and Rap (16.8%). The significant binomial correlation also indicates that user demographic information such as sex should be considered when recommending music to sing. Although the preference for same-sex singers (3%) does not account for a

³ <https://www.mturk.com/>

⁴ Workers were instructed to select whether an excerpt from Michael Jackson's Billie Jean was more familiar than an unreleased composition from one of the authors

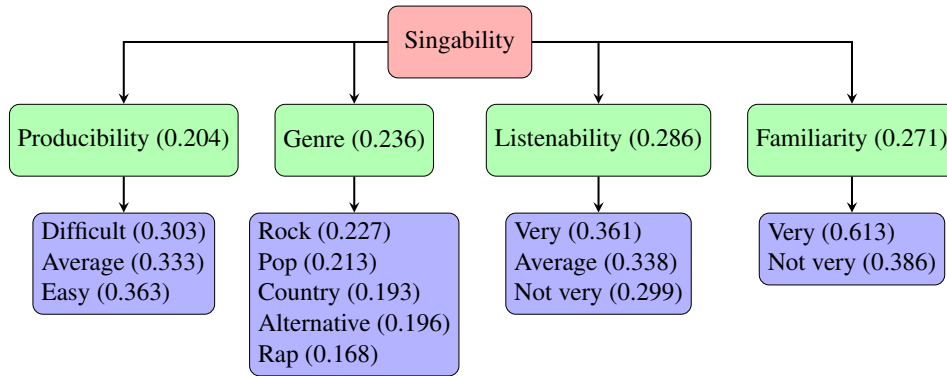


Figure 1. Analytic Hierarchy Process for Singability derived from Mechanical Turk experiment. The most important global priority was familiarity, followed by preference to listen, genre, and producibility.

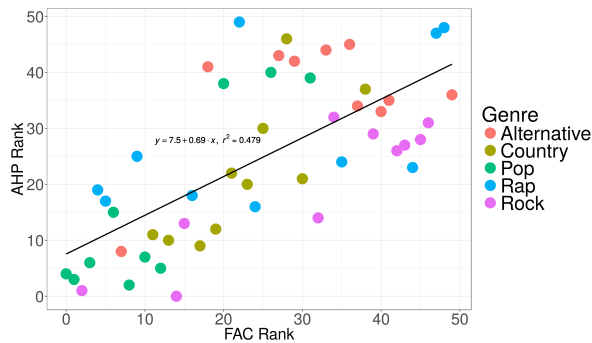


Figure 2. FAC-to-AHP Rank Scatterplot. X-axis represents ranks derived from the AHP analysis for a given track. Y-axis represents ranks derived from FAC analysis. Linear regression line for this data is plotted ($\hat{y} = 0.6922x + 7.8490$)

high degree of difference, recommendation systems based on human behaviours are relatively rare and can improve user satisfaction in generally unexplored ways.

A downside of this current investigation is that the variation of importance of global and local priorities was quite low, ranging between 2-3% across most factors. A more pronounced effect may be achievable using a more controlled, laboratory recruited participant pool. Producibility was also not a factor used to generate AHP ranking. The rationale for this is that there is no clear or simple way to evaluate vocalization difficulty of an excerpt relative to an MTurk worker's actual skill, whereas measures of familiarity and preference have a high degree of comorbidity with qualitative assessments [49].

An important component missing from this analysis is determining whether specific acoustic features influence ranking in meaningful way; is a song that is more singable one that generally has more pronounced vocals, or a faster tempo? Experiment two is a preliminary exploration into assessing whether some acoustic features are more important than others for determining singability based on the ranks generated through the AHP.

4. EXPERIMENT TWO: FEATURE IMPORTANCE EXPLORATION

After establishing that singability is a measurable cognitive process, the natural next step is analysis of acoustic features. Evaluating the importance of acoustic features related to singability may enable us to establish whether, or which, specific auditory signals contribute to this complex decision-making task. Experiment two provides preliminary, exploratory analysis into the importance of a specific set of acoustic features when evaluating singability.

4.1 Methods

Similar to [43], we extract perceptually-relevant features for singability under two categories: timbral and rhythmic. Signal processing is conducted using a combination of LibRosa [27], MIRToolbox [22], and vocal analysis work in [18]. 24 features (4 rhythmic and 20 timbral) in total were assessed. An averaged value for each feature was extracted for each song every 15-seconds. Timbral features include: Vocal-to-Accompaniment Ratio (VAR) [40], High Frequency Energy (HFE) [11], Mel-band Frequency Cepstral Coefficients (MFCC) 1-5 [13], spectral centroid mean and deviation [34], spectral roll off mean and deviation, and root mean squared (RMS) of energy mean and deviation [31]; rhythmic features include: tempo, zero-crossing mean and deviation [15], event density [2], and syllabic rate [18]. These features were selected for exploratory purposes due to their ubiquity in signal processing toolkits and MIR research.

4.2 Analysis

To determine whether specific features are more common in singable songs, we first conduct multiple linear regression comparing the AHP generated numeric values to the 24 extracted acoustic features. The multiple-comparisons F-Test was marginally significant ($F(23, 36) = 1.779, p = 0.05915, r^2 = 0.2328$), independent regressions yielded significant two features (Deviation of RMS and MFCC 5) and six marginally significant features (Deviations of spectral roll off, MFCCs 1 and 3, and means of RMS, spectral

Feature	t-value	p-value
Deviation of MFCC 5	-2.919	0.00604**
Deviation of RMS	-2.539	0.01558*
Deviation MFCC 1	1.994	0.05372.
Deviation spectral roll off	1.934	0.06099.
Mean of zero crossings	-1.864	0.0702.
Mean spectral centroid	1.738	0.09072.
Deviation MFCC 3	1.713	0.09525.
Mean of RMS	1.703	0.09713.

Table 1. Individual Linear Regression Significance Table. Multiple comparisons F-Test was marginally significant ($F(23, 36) = 1.779, p = 0.05915, r^2 = 0.2328$). $\cdot p < 0.1, * p < 0.01, ** p < 0.05$

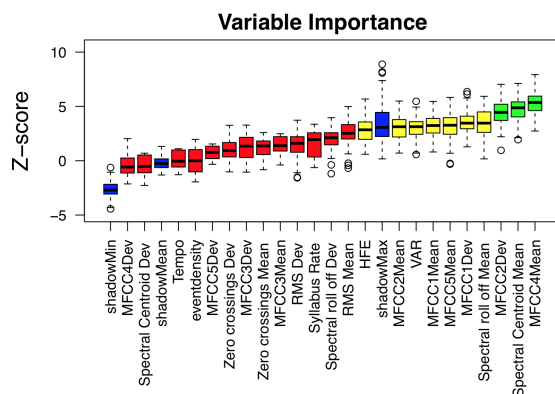


Figure 3. Random forest with regression model. Z-scores represent the relative importance of a feature in the determination of AHP-generated values for songs. Colours represent significance values: significant (green), marginally significant (yellow), not significant (red), and anchor values (blue).

centroid and zero-crossing). Table 1 provides a summary of analysis for all marginally significant features.

A statistical disadvantage of relying on standard linear regression analysis only is that multiple comparisons increasingly introduces type-I error with each added feature. We employ a random forest for regression and compare significant features across both models. An added benefit of using random forest is that it can assess the relative importance of each feature in the evaluation of singability. Three features significantly influenced AHP-generated singability scores (Mean of spectral centroid and MFCC 4, and deviation of MFCC 2), and six marginally influenced AHP-generated singability scores (Syllabic rate, VAR, HFE, mean of RMS, and deviation of MFCC). Figure 3 presents the relative importance of each feature (x-axis) as a Z-score (y-axis). Features that were at least marginally significant across both models included: mean of spectral centroid and RMS, and deviation of MFCC 1. Features that were at least marginally significant in the random forest model that were not significant using independent linear regressions included: mean of MFCC 4, spectral roll off, VAR, HFE, and syllabic rate.

4.3 Discussion

Both sets of analyses suggest that acoustic features may influence perceptions of singability. However, the models disagree on which features are maximally important in this decision. The three significant features that were shared across models (mean of RMS, spectral centroid, and deviation of MFCC 1) suggest that more singable songs are in general louder, brighter, and timbral fluctuations in high frequency energy may be particularly important when selecting music to sing to. Features where there was a disagreement in singability across models include zero-crossings, spectral roll off, VAR, HFE, and syllabic rate. This suggests that types of percussive sounds, pronounced vocals, and higher than average frequency in vocalizations and syllabic rate, may also contribute to evaluations of singability. The marginal significance of the multiple-comparisons F-test indicate that acoustic features may influence judgements of singability, however additional analysis needs to be conducted in order to demonstrate the validity of this assertion (see Section 5). Future work should investigate whether less common features, such as chorusness [1], are more relevant to singability.

Compared to Experiment One, the results from Experiment Two are less interpretable. It may be that the our corpus size, or that extracting high-level acoustic features from 15-second excerpts is insufficient sampling for this kind of analysis.

5. CONCLUSIONS

The methods utilized in both experiments may be useful for others in the refinement of psychologically-based music features such as danceability, or enable the exploration of other previously unexamined features.

Experiment One establishes a method for measuring complex cognitive decision making processes like singability in an operationalized manner. A major limitation of this operationalization is that it did not consider social and contextual features influencing singing preference. As described in Section 3.2, a benefit of using AHP is that including or removing global priorities is simple; future work should consider the role that other factors (such as social context and song lyrics) may play in the evaluation of singability.

Experiment Two provides a preliminary exploration of the extent acoustic features influence singability scores generated in experiment one. Two statistical models, one simple and the other more complex, were used to determine what features may be contributing most to the evaluation of singability. Significant features in common across the two models suggests that further signal analysis will be important future work.

This exploratory work does not definitively establish singability as a core feature of the music. Rather we suggest that it provides compelling evidence to support a perceptual process of singability, and a refinable methodology to explore or support other properties involving cognition.

6. ACKNOWLEDGEMENTS

This project is funded by Smule Inc. We would also like to thank our reviewers for their insightful comments and feedback.

7. REFERENCES

- [1] Hooked: a Game for Discovering what Makes Music Catchy. (Proceedings of the 12th International Society for Music Information Retrieval Conference):245–250, 2013.
- [2] Samer A Abdallah and Mark D Plumbley. Probability as metadata: event detection in music using ica as a conditional density model. In *Proc. 4th Int. Symp. Independent Component Analysis and Signal Separation (ICA2003)*, pages 233–238. Citeseer, 2003.
- [3] Jean-Julien Aucouturier and Emmanuel Bigand. Seven problems that keep mir from attracting the interest of cognition and neuroscience. *Journal of Intelligent Information Systems*, 41(3):483–497, 2013.
- [4] Michael D Barone, Jotthi Bansal, and Matthew H Woolhouse. Acoustic features influence musical choices across multiple genres. *Frontiers in psychology*, 8:931, 2017.
- [5] Wilmer T Bartholomew. A physical definition of good voice-quality in the male voice. *the Journal of the Acoustical Society of America*, 5(3):224–224, 1934.
- [6] Tara S Behrend, David J Sharek, Adam W Meade, and Eric N Wiebe. The viability of crowdsourcing for survey research. *Behavior research methods*, 43(3):800, 2011.
- [7] Daniel E Berlyne. Novelty, complexity, and hedonic value. *Attention, Perception, & Psychophysics*, 8(5):279–286, 1970.
- [8] Denise D Bielby and C Lee Harrington. Managing culture matters: Genre, aesthetic elements, and the international market for exported television. *Poetics*, 32(1):73–98, 2004.
- [9] Dmitry Bogdanov, Joan Serra, Nicolas Wack, and Perfecto Herrera. From low-level to high-level: Comparative study of music similarity measures. In *Multimedia, 2009. ISM'09. 11th IEEE International Symposium on*, pages 453–458. IEEE, 2009.
- [10] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José R Zapata, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. pages 493–498. Citeseer, 2013.
- [11] Lauren B Collister and David Huron. Comparison of word intelligibility in spoken and sung phrases. *Empirical Musicology Review*, 3(3):109–122, 2008.
- [12] Mihaly Csikszentmihalyi. *Flow and the psychology of discovery and invention*. New York: Harper Collins, 1996.
- [13] Jeremiah D Deng, Christian Simmermacher, and Stephen Crane. A study on feature analysis for musical instrument classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(2):429–438, 2008.
- [14] Anders Friberg, Erwin Schoonderwaldt, Anton Hedblad, Marco Fabiani, and Anders Elowsson. Using perceptually defined music features in music information retrieval. *arXiv preprint arXiv:1403.7923*, 2014.
- [15] Fabien Gouyon, François Pachet, Olivier Delerue, et al. On the use of zero-crossing rate for an application of classification of percussive sounds. In *Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00)*, Verona, Italy, 2000.
- [16] Chu Guan, Yanjie Fu, Xinjiang Lu, Enhong Chen, Xiaolin Li, and Hui Xiong. Efficient karaoke song recommendation via multiple kernel learning approximation. *Neurocomputing*, 2017.
- [17] David J Hargreaves, Chris Comber, and Ann Colley. Effects of age, gender, and training on musical preferences of british secondary school students. *Journal of Research in Music Education*, 43(3):242–250, 1995.
- [18] K.M. Ibrahim, D. Grunberg, K. Agres, C. Gupta, and Y. Wang. Intelligibility of sung lyrics: A pilot study. Suzhou, China, 2017.
- [19] Marcia K Johnson, Jung K Kim, and Gail Risse. Do alcoholic korsakoff's syndrome patients acquire affective reactions? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 11(1):22, 1985.
- [20] Stefan Koelsch and Walter A Siebel. Towards a neural basis of music perception. *Trends in cognitive sciences*, 9(12):578–584, 2005.
- [21] Carol L Krumhansl. Plink:" thin slices" of music. *Music Perception: An Interdisciplinary Journal*, 27(5):337–354, 2010.
- [22] Olivier Lartillot, Petri Toivainen, and Tuomas Eerola. A matlab toolbox for music information retrieval. *Data analysis, machine learning and applications*, pages 261–268, 2008.
- [23] Guy Madison. Experiencing groove induced by music: consistency and phenomenology. *Music Perception: An Interdisciplinary Journal*, 24(2):201–208, 2006.
- [24] George Mandler, Yoshio Nakamura, and Billie J Van Zandt. Nonspecific effects of exposure on stimuli that cannot be recognized. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13(4):646, 1987.

- [25] Kuang Mao, Ju Fan, Lidan Shou, Gang Chen, and Mohan Kankanhalli. Song recommendation for social singing community. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 127–136. ACM, 2014.
- [26] Kuang Mao, Lidan Shou, Ju Fan, Gang Chen, and Mohan S Kankanhalli. Competence-based song recommendation: Matching songs to ones singing skill. *IEEE Transactions on Multimedia*, 17(3):396–408, 2015.
- [27] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.
- [28] Max Meyer. Experimental studies in the psychology of music. *The American Journal of Psychology*, 14(3/4):192–214, 1903.
- [29] Adrian C North and David J Hargreaves. Subjective complexity, familiarity, and liking for popular music. *Psychomusicology: A Journal of Research in Music Cognition*, 14(1-2):77, 1995.
- [30] Nathan Novemsky, Ravi Dhar, Norbert Schwarz, and Itamar Simonson. Preference fluency in choice. *Journal of Marketing Research*, 44(3):347–356, 2007.
- [31] Costas Panagiotakis and Georgios Tziritas. A speech/music discriminator based on rms and zero-crossings. *IEEE Transactions on multimedia*, 7(1):155–166, 2005.
- [32] Isabelle Peretz, Danielle Gaudreau, and Anne-Marie Bonnel. Exposure effects on music preference and recognition. *Memory & Cognition*, 26(5):884–902, 1998.
- [33] Thomas L Saaty. How to make a decision: the analytic hierarchy process. *European journal of operational research*, 48(1):9–26, 1990.
- [34] Emery Schubert, Joe Wolfe, and Alex Tarnopolsky. Spectral centroid and timbre in complex, multiple instrumental textures. In *Proceedings of the international conference on music perception and cognition, North Western University, Illinois*, pages 112–116. sn, 2004.
- [35] H-J Schultz-Coulon, R-D Battmer, and H Riechers. Der 3-khz-formant—ein mass für die tragfähigkeit der stimme? *Folia Phoniatrica et Logopaedica*, 31(4):302–313, 1979.
- [36] Sebastian Streich and Perfecto Herrera. Detrended fluctuation analysis of music signals: Danceability estimation and further semantic characterization. In *Proceedings of the 118th AES Convention*, 2005.
- [37] Johan Sundberg. Level and center frequency of the singer’s formant. *Journal of voice*, 15(2):176–186, 2001.
- [38] Mari Tervaniemi, Minna Huottilainen, and Elvira Brattico. Melodic multi-feature paradigm reveals auditory profiles in music-sound encoding. *Frontiers in human neuroscience*, 8(July):496, 2014.
- [39] Mari Tervaniemi, Lauri Janhunen, Stefanie Kruck, Vesa Putkinen, and Minna Huottilainen. Auditory profiles of classical, jazz, and rock musicians: Genre-specific sensitivity to musical sound features. *Frontiers in psychology*, 6:1900, 2016.
- [40] Wei-Ho Tsai, Dwight Rodgers, and Hsin-Min Wang. Blind clustering of popular music recordings based on singer voice characteristics. *Computer Music Journal*, 28(3):68–78, 2004.
- [41] Chia-Jung Tsay. Sight over sound in the judgment of music performance. *Proceedings of the National Academy of Sciences*, 110(36):14580–14585, 2013.
- [42] Suppawong Tuarob and Conrad S Tucker. Quantifying product favorability and extracting notable product features using large scale social media data. *Journal of Computing and Information Science in Engineering*, 15(3):031003, 2015.
- [43] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.
- [44] Omkarprasad S Vaidya and Sushil Kumar. Analytic hierarchy process: An overview of applications. *European Journal of operational research*, 169(1):1–29, 2006.
- [45] Christopher Watts, Jessica Murphy, and Kathryn Barnes-Burroughs. Pitch matching accuracy of trained singers, untrained subjects with talented singing voices, and untrained subjects with nontalented singing voices in conditions of varying feedback. *Journal of Voice*, 17(2):185–194, 2003.
- [46] Felix Weninger, Martin Wöllmer, and Björn Schuller. Automatic assessment of singer traits in popular music: Gender, age, height and race. *Proceedings of the 10th International Society for Music Information Retrieval Conference*, pages 37–42, 2011.
- [47] William R Wilson. Feeling more than we can know: Exposure effects without learning. *Journal of personality and social psychology*, 37(6):811, 1979.
- [48] Maria AG Witek, Eric F Clarke, Mikkel Wallentin, Morten L Kringelbach, and Peter Vuust. Syncopation, body-movement and pleasure in groove music. *PloS one*, 9(4):e94446, 2014.
- [49] Robert B Zajonc. Attitudinal effects of mere exposure. *Journal of personality and social psychology*, 9(2p2):1, 1968.

Session E

Timbre, tagging, similarity, patterns
and alignment

ANALYSIS BY CLASSIFICATION: A COMPARATIVE STUDY OF ANNOTATED AND ALGORITHMICALLY EXTRACTED PATTERNS IN SYMBOLIC MUSIC DATA

Iris Yuping Ren
Utrecht University
y.ren@uu.nl

Anja Volk
Utrecht University
a.volk@uu.nl

Wouter Swierstra
Utrecht University

w.s.swierstra@uu.nl

Remco C. Veltkamp
Utrecht University
r.c.veltkamp@uu.nl

ABSTRACT

Musical patterns are salient passages that repeatedly appear in music. Such passages are vital for compression, classification and prediction tasks in MIR, and algorithms employing different techniques have been proposed to find musical patterns automatically. Human-annotated patterns have been collected and used to evaluate pattern discovery algorithms, e.g., in the Discovery of Repeated Themes & Sections MIREX task. However, state-of-the-art algorithms are not yet able to reproduce human-annotated patterns. To understand what gives rise to the discrepancy between algorithmically extracted patterns and human-annotated patterns, we use jsymbolic2 to extract features from patterns, visualise the feature space using PCA and perform a comparative analysis using classification techniques. We show that it is possible to classify algorithmically extracted patterns, human-annotated patterns and randomly sampled passages. This implies: (a) Algorithmically extracted patterns possess different properties than human-annotated patterns (b) Algorithmically extracted patterns have different structures than randomly sampled passages (c) Human-annotated patterns contain more information than randomly sampled passages despite subjectivity involved in the annotation process. We further discover that rhythmic features are of high importance in the classification process, which should influence future research on automatic pattern discovery.

1. INTRODUCTION

Patterns occur in many dimensions of life: we constantly look for patterns to classify and predict based on our experience [40]. In music, composers employ patterns to induce structures to their music [14]; listeners look for patterns while they listen attentively [16, 19]; performers learn patterns to better memorise, perform and improvise [39]; musicologists use patterns as evidence for categorisation and theorisation [1, 23]. In this paper, we work mainly with repeated patterns which characterise and categorise folk songs.

Because of the many potential applications of musical patterns, algorithms that can automatically identify patterns are useful in many contexts. Automatic pattern discovery is an active research area in which many different methods have been developed, such as string-based approaches [5, 8, 17, 21, 22, 32], geometric approaches [4, 7, 29, 41], data mining approaches [6, 36], and machine learning approaches [34, 46].

One open question is how one should evaluate the quality of algorithmically extracted patterns. One common approach is to compare the extracted patterns with human-annotated patterns [2, 11, 15]. However, because of the aforementioned versatile application possibilities and diverse definitions of musical patterns, we face several challenges using human-annotated patterns to evaluate the algorithms. First, there is a lack of human-annotated pattern datasets in general [37]. Second, subjectivity and irreducible human errors could be introduced in the annotation process [27]. Third, it is not straightforward to see what metrics one should compute to compare the human-annotated patterns with automatically extracted patterns.

Previous research has addressed these challenges to a certain extent. Historically, algorithms have been tested on unassociated datasets with disparate metrics [15]. One attempt to standardise the evaluation of algorithms is the MIREX Discovery of Repeated Themes & Sections task initiated in 2014. In the task, a pattern is defined as a set of time-pitch pairs that occurs at least twice in a piece of music and the JKU-PDD dataset was introduced [11]. According to the evaluation metrics in this task, the state-of-the-art algorithms perform acceptably well in precision, recall, and F1-scores, although they cannot reproduce the human-annotated patterns yet. Another pattern annotation dataset which has been used for evaluating the algorithms is the MTC-ANN Dutch Folk Song dataset [43]: human-annotations have been compared with algorithmically extracted patterns by their performance in a classification task [2] showing the annotated patterns perform better. Furthermore, a large disagreement between annotated and computationally extracted patterns has been shown in both the JKU-PDD and MTC-ANN dataset in [37].

The aim of this paper is to identify and analyse the discrepancy between human annotations and algorithmically extracted patterns. To achieve this goal, we extract characteristic features from human-annotated and automatically extracted patterns, and conduct a comparative study on the



© Iris Yuping Ren, Anja Volk, Wouter Swierstra, Remco C. Veltkamp. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Iris Yuping Ren, Anja Volk, Wouter Swierstra, Remco C. Veltkamp. "Analysis by classification: A comparative study of annotated and algorithmically extracted patterns in symbolic music data", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

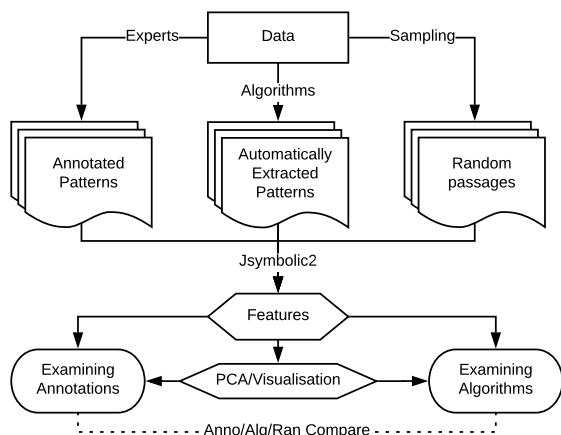


Figure 1. Pipeline of our experiments. Given the music data, experts annotate patterns, algorithms extract patterns, and we randomly sample passages in the corpus. Tasks are shown in rounded boxes. Diamond boxes are transformed data/features. Section 2 gives a detailed description.

pattern features using classification methods. To establish a baseline, we randomly sample passages that have the same lengths as human annotations. By performing a ternary classification task amongst the human-annotated patterns, algorithmically extracted patterns and random passages, we provide evidence that they are separable by classifiers. Despite taking musical patterns out of context and only considering the local structures annotated by humans and extracted by algorithms, the result of the experiment shows preliminary implications for the future design and evaluation of pattern discovery algorithms.

Contribution Using the monophonic MTC-ANN Dutch Folk Song dataset [43], our main contributions are: (a) By calculating features of human-annotated, automatically extracted and sampled passages, we summarise and visualise the distributions of patterns in the feature space using Principal Component Analysis (PCA) (b) Our classifiers successfully discriminate between human-annotated patterns and algorithmically extracted patterns above random chance level, which enables us to analyse what characterises the differences between human-annotated patterns, automatically extracted patterns and random passages (c) Based on the analysis of features and classification results, we propose several ways to improve pattern discovery algorithms.

Figure 1 is the pipeline of our experiments to be detailed in the next section. Abbreviations such as **Anno** (Annotations), **Ran** (Random passages), and **Alg** (Algorithmically extracted patterns) are used in tables and figures.

2. DATA PREPARATION

We use the MTC-ANN Dutch Folk Song dataset [43], which contains an exceptionally large number of annotated patterns and is therefore suitable for a classification experiment. In this section, we examine groups of patterns, random passages, and their features in this dataset.

Algorithm	#Pattern	#Occurrences	Incl.
(Annotation)	153	1657	✓
SIAR	893	5576	✓
SIAP	250	3650	✓
SIAF1	822	5308	✓
VM	182	25679	✓
VM2	159	4658	✓
SC	126	355	✓
SCFP	200	724	✓
PatMinr	105663	182306	✗
ME	3339951	5651956	✗
MDGP	3543940	5457210	✗
COSIATEC	61499	99501	✗

Table 1. Algorithms and the count of extracted patterns. Abbreviation correspondence and details are given in Section 2.1. The counts of PatMinr, ME, MDGP, and COSIATEC are larger by several magnitude because we include a parameter sweep, while other algorithms use a parameter setting preset by authors of the algorithms. A comprehensive investigation into parameter settings of algorithms is not conducted in this paper.

2.1 Pattern groups in MTC-ANN

Annotated patterns During the making of MTC-ANN, three experts have been asked to annotate the prominent patterns in each song which best classify the song into one of 26 tune families. *Tune family* is a concept in ethnomusicology that groups together tunes sharing the same ancestor in the process of oral transmission [9]. The dataset consists of 360 Dutch folk songs with 1657 annotated pattern occurrences. In an annotation study on what influences human judgements when categorising melodies belonging to the same tune family, repeated patterns turned out to play the most important role [45]. It is, therefore, reasonable to use repeated pattern discovery algorithms on this dataset.

Patterns from algorithms Table 1 shows the number of extracted patterns from state-of-the-art musical pattern discovery algorithms that have been used and compared in previous research [2, 37]. The count numbers for PatMinr [22], MotivesExtractor (ME) [32], MDGP [8], and COSIATEC [26] include different parameter settings of the algorithm and are therefore several magnitudes larger than other entries. We do not include these patterns because a comprehensive parameter search of the algorithms would be out of the scope of this paper. For the same reason, although algorithms such as SIATECCompress - TLP (SIAP), SIATECCompress - TLF1 (SIAF1), SIATECCompress - TLR (SIAR) are not optimised for MTC-ANN, a parameter search is not conducted.

We use the seven pattern discovery algorithms and extract the patterns from the MTC-ANN dataset using the same setup as in [2, 37]. The extracted patterns from each algorithm form a subgroup under the umbrella of the extracted pattern group. The seven algorithms were submitted to the MIREX task during 2014-2017: SIATECCompress - TLP (SIAP), SIATECCompress - TLF1 (SIAF1), SIATECCompress - TLR (SIAR) [28], VM & VM2 [44], SYMCHM (SC) [35], and SIARCT-CFP (SIACFP) [7].

Sampling random passages We compare annotated and extracted patterns with randomly sampled passages as a baseline in order to potentially support or refuse the significance of musical patterns. In more detail, taking the annotated patterns from MTC-ANN, random passages are sampled with the following procedures: for each annotated pattern, we find the corresponding song where the annotation appears. We then find a random starting point and take an excerpt of the same length as the pattern to construct a candidate excerpt. Finally, we repeat the sampling procedures five times to prevent accidental results.

2.2 Compute features

Much work of research exists on how to design and compute musical features. As we are concerned with repeated patterns, and there are many possibilities as to what features make a pattern repetitive [42], we hence adopt a standardised feature extraction process as described below.

Feature Calculation We calculate features from the patterns by using a common feature extraction tool: the `jsymbolic2` toolbox in the `jMIR` toolset [25]. `jsymbolic2` takes MIDI files as input and computes 155 musically meaningful features in six categories: texture, rhythm, dynamics, pitch, melody and chords. After computing all the features for all the patterns, we have a feature vector of 155 dimensions associated with each pattern. Another well-known feature extraction package, the `FANTASTIC` toolbox [30] is not used because it cannot process input of short length, which excludes valuable annotated patterns from contributing to subsequent classification tasks.

Feature Selection We perform a feature selection step and retain 63 features by first eliminating the features which are constant across all patterns, such as *Vibrato Prevalence*, *Average Range of Glissandos*, and so forth. Next, we eliminate the features which are not relevant to the music content of time and pitch, such as the dynamics features and artefacts introduced by MIDI conversion.

PCA and Visualisation PCA is known to be a practical preprocessing step and visualisation tool for classification problems. PCA produces linear combinations of features which maximise variances in a given dataset and are suitable for visualising differences in data.

In Figure 2, we plot different groups and subgroups of patterns in a two-dimensional¹ PCA embedding of the feature space. We make four cross-group comparisons to show typical cases of how musical patterns distribute in the feature space spanned by the first two components of the PCA decomposition. The visualisation is generated by using the annotated patterns as training data to obtain the PCA embedding, then project random passages and patterns from different algorithms onto this PCA embedding space.

From the four snapshots we take from the musical pattern PCA feature space as shown in Figure 2, we make several observations: (1) Annotated patterns and random passages have an extensive area of overlap, which makes it impossible to find a linear classifier using the first two principal components of the annotated patterns, which in turn makes

it nontrivial to differentiate the two groups of patterns as shown in the upper left figure. (2) SIAR patterns exhibit very different distribution from the annotated patterns and random passages as shown in the top right subfigure. Notice the annotated patterns concentrate at the top left corner. In this case, it is relatively easy to separate the long-tail area of the extracted patterns from the annotation area. By applying this observation and designing a filtering process, it could substantially improve the performance of the SIAR algorithm on MTC-ANN. (3) The overlap between the annotated patterns and extracted patterns is small in the bottom left figure. A linear classifier can be devised to separate the two groups of data using the first two principal dimensions of the annotated patterns. The extracted patterns of the SC algorithm have different features than the annotated patterns. (4) In the bottom right figure, we show all the heterogeneous patterns as extracted by algorithms, annotated by humans or randomly sampled in the same PCA embedding. Patterns extracted by algorithms of the same family, namely *SIATECCompress - TLP (SIAP)*, *SIATECCompress - TLF1 (SIAF1)*, *SIATECCompress - TLR (SIAR)*, and *SIACFP* tend to share the same long-tail property, and therefore their performance on MTC-ANN can be improved by an extra filtering step as described above.

In summary, setting out from the visual examination and our observations above, it is promising to apply classification techniques to discriminate the features of different groups of patterns. We commence on the classification task and conduct a comparative analysis using the classification results in the next section.

3. METHOD CONFIGURATION

In this section, we introduce the classifiers and evaluation metrics we use for the classification task.

3.1 Classification

Supervised classification methods have been used extensively in MIR tasks such as genre classification and classifying geographically different corpora. In addition, comparative analyses using classification methods have been performed in many areas of research [10, 33]. To the best of our knowledge, using supervised classification for conducting comparative analyses have not been used with symbolic musical patterns. In this paper, we use supervised classification methods to differentiate human-annotated, algorithmically extracted and randomly sampled passages in MTC-ANN. By putting patterns into groups (the group of algorithmically extracted patterns, the group of annotations, and the group of random passages) and observing whether there are systematic differences on the group level, we gain a different perspective than using the metrics based on individual patterns, such as the precision, recall, and F1-score used in MIREX.

To prevent the results to be classifier-specific, we use a mixture of simple and more sophisticated, linear and non-linear classifiers to perform the ternary classification task. We also use standard machine learning techniques to train and test classifiers: first, scaling and centering preprocessing steps are performed on all the features and PCA input;

¹ More visualisations can be found at <https://goo.gl/qmyxdh>

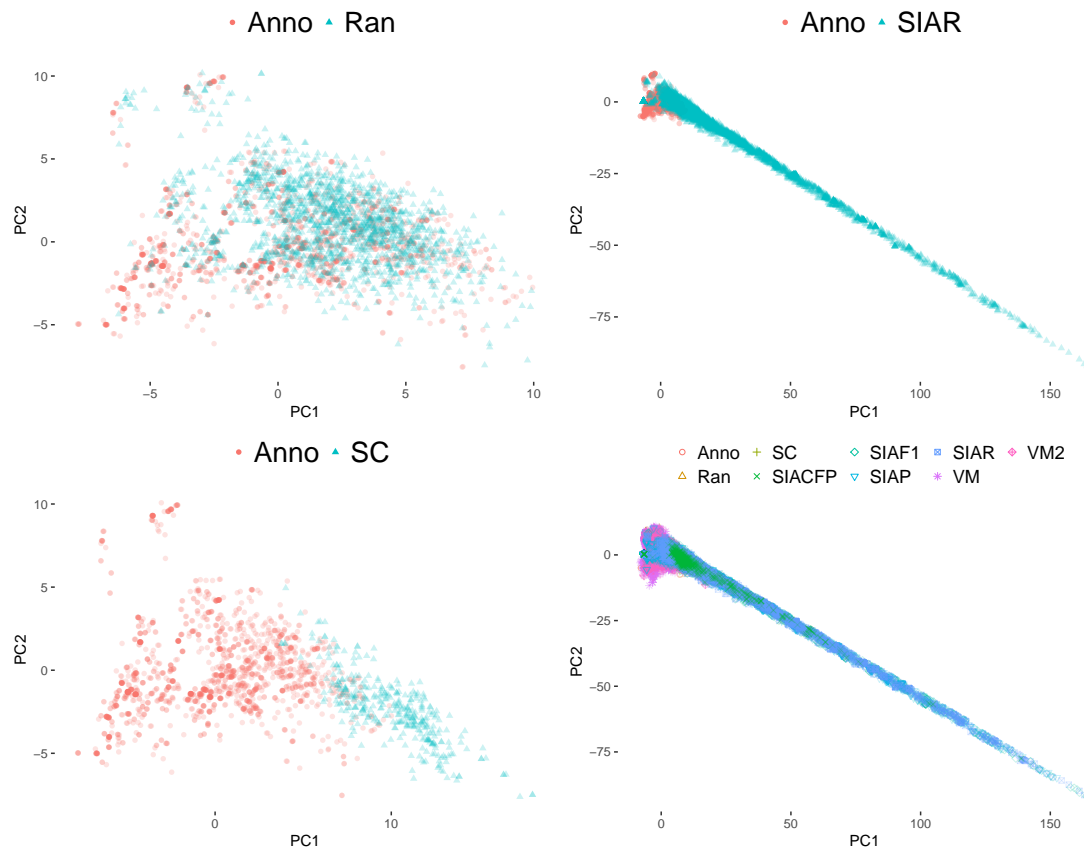


Figure 2. Visualisation of different groups of patterns using the space spanned by the first two principal components of the annotated pattern features. The legend denoting the colour correspondence with algorithms/annotated patterns/random passages is on the top of each subfigure. Notice that the scopes of figures in the left column are subregions of figures on the right. Notes on each subfigure: (1) Upper left: Random passages and annotated patterns. The overlap between the two groups is large, and it is nontrivial to separate them in this two-dimensional PCA embedding. (2) Upper right: SIAR patterns and the annotated patterns. SIAR patterns exhibit a long-tail behaviour which is not shared by the annotated patterns. (3) Bottom left: SC patterns and the annotated patterns. The overlap of the data points is small, which makes it easier to separate the two groups in this embedding. (4) Bottom right: Random passages, annotated patterns and patterns from all algorithms. We see some of the algorithmically extracted patterns are very different from the annotated patterns, and the algorithms belonging to the same family exhibit the same long-tail behaviour.

additionally, to avoid overfitting, for all experiments, we use a 10-fold cross-validation 3-times repetition scheme. The PCA projection and parameter search of each classifier are performed separately on each fold. The six statistical classifiers we use are:

GBM [13] (Gradient Boosting Machine) produces a prediction model consisting of an ensemble of decision trees. The parameters we search through are the learning rate, the complexity of trees, the minimum number of samples to commence splitting and the number of iterations.

LVQ [18] (Linear Vector Quantisation) applies a winner-takes-all Hebbian learning-based approach. We search through two parameters in this classifier: the codebook size and the number of prototypes.

LDA [38] (Linear Discriminant Analysis) produces a linear classifier which finds a linear combination of features that best separates different classes in datasets. This classifier does not contain parameters.

NB [31] (Naive Bayes) computes the conditional a-

posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule. Three parameters are tuned for this classifier: the Laplace smoothing, kernel bandwidth and distribution type.

RF [3] (Random Forest) operates by constructing a multitude of decision tree. The parameter we consider is the number of variables per level.

SVM [12] (Support Vector Machine) calculates a map from data to a new representation so that the data points of the separate categories are divided by a gap that is as wide as possible. We use the radial basis function kernel and consider two parameters: the smoothing factor and the weight of training examples.

The experiments have been performed using R. The task takes about 2 hours on an i7 CPU with a maximum memory usage of 2Gb. For reproducibility, the data and code to replicate the experiments can be downloaded².

²<https://github.com/irisyupingren/patdisISMIR2018>

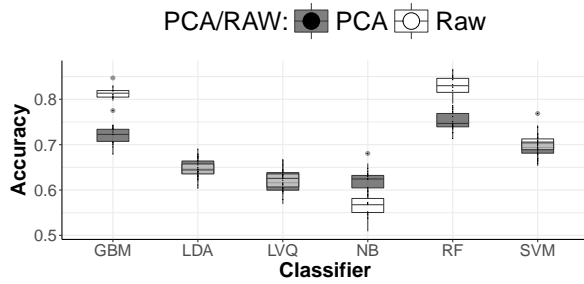


Figure 3. Accuracy values of classifiers in thirty experiments (10-fold cross-validation repeated three times) using six classifiers with jsymbolic2 features and features after PCA decomposition.

Other schemes with different parameters and with a new test set split were used, too, but are omitted because they give similar results to our analysis.

3.2 Evaluation

We mainly use accuracy and its variance as a measure of the performance of classifiers. To further interpret the results of the classification task, we compute confusion matrices and feature importance measures. Ten other metrics for each classifier are provided for further inspection³. In the next section, we report the most relevant results.

4. RESULTS AND DISCUSSION

In this section, we first report the model metrics of classifiers. By comparing the metrics, we identify Random Forest as the best classifier. Then we interpret the performance of the Random Forest classifier using the confusion matrix. Last, we examine important features in our best model.

4.1 Model metrics

In Figure 3, we show the accuracy and variance of different classifiers using two groups of features: the raw features and features after PCA decomposition. The baseline accuracy is $\frac{1}{\#group} \sim 33\%$ because the number of patterns in each group is the same = 1657, as ensured by uniform sampling.

We see that all the classifiers give a result higher than the baseline accuracy. PCA improves the performance of the classifier NB; for three classifiers, LVQ, SVM and LDA, using PCA or raw input does not make a significant difference on the performance; the performance of other classifiers is worse when using the PCA input. PCA has different influences on the performance of classifiers because there are different internal feature transformation mechanisms in each classifier. Overall, the random forest classifier gives the best results with the raw feature input and the parameter $\#variables = 32$.

The high accuracy and the fact that we can construct a classifier to differentiate the three groups of data imply that: first, algorithmically extracted patterns possess different properties than human-annotated patterns, which suggests an extra consideration to features of patterns when trying to discover patterns automatically; second, algorithmically

³ <https://goo.gl/ezuTCT>

Original → Classified ↓	Alg	Ran	Anno
Alg	1595(±7.4)	17.2(±4.6)	24.8(±8.4)
Ran	8.3(±2.7)	1597(±2.8)	5.0(±2.2)
Anno	54.1(±9.9)	42.6(±2.7)	1627(±10.0)

Table 2. Confusion matrix results from the ternary classification experiment using the Random Forest classifier: mean and variance (in parenthesis) of ten experiments. The row names indicate the patterns are classified into the group of this name by the classifier; the column names indicate the patterns are originally from the group of this name. Three groups of data are classified with high accuracies and significant p-values $\ll 0.05$.

extracted patterns have different structures than random passages, which means the extracted patterns cannot be replaced by sampled passages and could be more useful than sampled passages for various applications that employ musical patterns; last, human-annotated patterns contain more information than randomness despite subjectivity involved in the annotation process, which is in agreement with the carefully designed annotation acquiring process [43] and the previous findings that the annotations are useful for classifying tune families [2].

4.2 Confusion Matrix

In Table 2, we give the confusion matrix results calculated from the classifier which has the best classification results: Random Forest. We perform the repeated cross-validation experiment ten times and take the average and variance of the resulting ten confusion matrices. The results show us on the individual patterns level how different groups of data are separable to one another. The sum of each column is roughly 1657, which is the group size of our data. The row sums do not have this constraint because we do not put restrictions on the group size as determined by the classifier. To read the table, for example, the number 24.8 in the right top corner of the table is the mean number of patterns classified as algorithmically extracted patterns but are actually annotations.

We see the classifier can differentiate the three groups with few misclassified instances. Although it would indicate a good performance of the algorithms if the count in the confusion matrix is larger in the algorithm pattern group and the annotation pattern group, we come to the conclusion that the algorithmically extracted patterns, annotated patterns and random passages all possess their own traits and are not similar enough for the classifier to fail. This is in accordance with previous research that the extracted patterns are not yet indistinguishable from the human annotations [2, 37]. On the positive side, we establish that neither annotated patterns nor extracted patterns are as meaningless as random data.

4.3 Feature Importance

In Figure 4, we show the individual importance value of the features in the classification process by using the Boruta algorithm [20]. The Boruta algorithm randomly duplicates

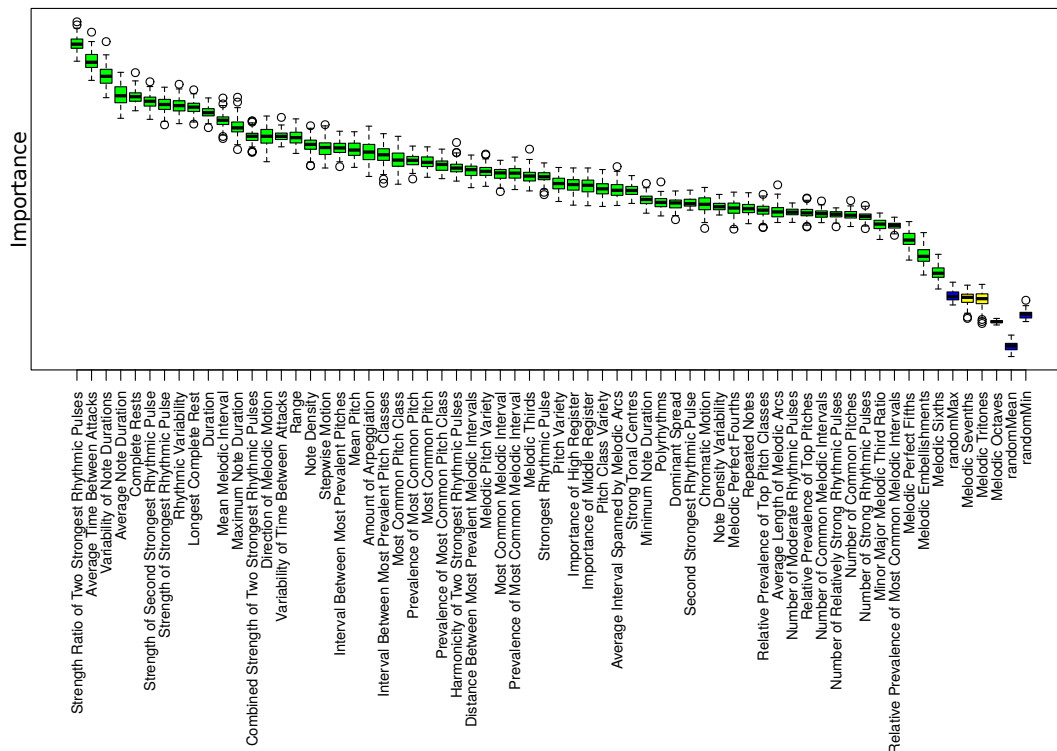


Figure 4. Feature importance in classifying annotated patterns, extracted patterns and randomly sampled passages using a random forest classifier. The boxplot shows the mean and variance (interquartile ranges) of the feature importance value [20]. The features are ranked by their importance. We omit the y-axis label because the absolute importance values are not relevant for our analysis. The colour green indicates features that are more important than the random features and are therefore confirmed to be important; blue entries show the performance of the random features; red and yellow indicate unimportant and tentative features respectively.

and shuffles the values in the original features. The algorithm then employs the random features together with the original features in classification tasks. During the classification process, the algorithm calculates and compares the Mean Decrease Impurity importance value [24].

Although we have 23 rhythmic features out of 63 features in total, all top ten most important features are rhythmic features. This suggests that these rhythmic features are relatively more important than other features in constructing the random forest classifier. The prominent features give hints on potential improvements to current existing pattern discovery algorithms. String-based and data mining algorithms translate pitch and duration pairs into a list of symbols and do not take into account metric structures imposed by musical punctuations such as bar lines and measures. Other known algorithms also seldom explicitly consider metric features in patterns. The feature importance values send the message that, in designing and evaluating pattern discovery algorithms, at least for the MTC-ANN dataset, we should take metric structures into considerations as well as the repetitions and pitch related features in the patterns.

In addition, the importance of other jsymbolic2 features is confirmed with the exception of three features which performed worse or at the same level as random features, as shown in Figure 4. For example, the Melodic Octaves feature is confirmed to be unimportant and the Melodic

Sevenths and Melodic Tritones feature are marked to be a tentative attribute. They are unessential features because such intervals rarely happen in the MTC-ANN dataset.

5. CONCLUSIONS AND FUTURE WORK

We visualised and successfully classified human-annotated patterns, algorithmically extracted patterns and random passages in MTC-ANN. An analysis of the classification results suggests that the automatically extracted patterns are not yet indistinguishable from the human-annotated patterns, and both extracted and annotated patterns show different traits than randomly sampled passages. Using classification methods for comparative analysis of pattern groups provides a new perspective on examining the output of pattern discovery algorithms than the comparison of individual patterns in the MIREX task. In this way, we discover that rhythmic features play an important role in distinguishing the groups of patterns in MTC-ANN.

Future research needs to consider different contexts of patterns, such as within a melody, within a tune family and within the corpus, in order to investigate the influence of the context on what establishes a musical pattern. Expanding our research to other datasets once pattern annotations become available will allow us to verify whether the importance of rhythmic features is specific to MTC-ANN.

6. REFERENCES

- [1] Kofi Agawu. *Music as discourse: Semiotic adventures in romantic music*. Oxford University Press, 2014.
- [2] Peter Boot, Anja Volk, and W. Bas de Haas. Evaluating the role of repeated patterns in folk song classification and compression. *Journal of New Music Research*, 45(3):223–238, 2016.
- [3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [4] Chantal Buteau and Guerino Mazzola. Motivic analysis according to Rudolph Reti: Formalization by a topological model. *Journal of Mathematics and Music*, 2(3):117–134, 2008.
- [5] Emiliós Cambouropoulos. Musical parallelism and melodic segmentation. *Music Perception*, 23(3):249–268, 2006.
- [6] Tsung-Ping Chen and Li Su. Discovery of repeated themes and sections with pattern clustering. *Music Information Retrieval Evaluation eXchange (MIREX 2017)*, 2017.
- [7] Tom Collins, Andreas Arzt, Sebastian Flossmann, and Gerhard Widmer. SIARCT-CFP: Improving precision and the discovery of inexact musical patterns in point-set representations. *Proceedings of the 14th International Society for Music Information Retrieval Conference*, pages 549–554, 2013.
- [8] Darrell Conklin. Discovery of distinctive patterns in music. *Intelligent Data Analysis*, 14(5):547–554, 2010.
- [9] James R. Cowdery. A fresh look at the concept of tune family. *Ethnomusicology*, 28(3):495–504, 1984.
- [10] Luciano da Fontoura Costa and Roberto Marcondes Cesar Jr. *Shape classification and analysis: theory and practice*. CRC Press, Inc., 2009.
- [11] Music Information Retrieval Evaluation eXchange (MIREX) 2013. Discovery of Repeated Themes & Sections. <http://www.musicir.org/mirex/wiki/2013>.
- [12] Vojtech Franc and Václav Hlaváč. Multi-class support vector machine. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 2, pages 236–239. IEEE, 2002.
- [13] Jerome H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
- [14] Jia-Lien Hsu, Arbee LP Chen, and C-C Liu. Efficient repeating pattern finding in music databases. In *Proceedings of the 7th International Conference on Information and Knowledge Management*, pages 281–288. ACM, 1998.
- [15] Berit Janssen, W. Bas de Haas, Anja Volk, and Peter van Kranenburg. Finding repeated patterns in music: State of knowledge, challenges, perspectives. *Proceedings of the 10th International Symposium on Computer Music Modeling and Retrieval*, pages 277–297, 2013.
- [16] Mari Riess Jones. Dynamic pattern structure in music: Recent theory and research. *Perception & psychophysics*, 41(6):621–634, 1987.
- [17] Ian Knopke and Frauke Jürgensen. A system for identifying common melodic phrases in the masses of Palestrina. *Journal of New Music Research*, 38(2):171–181, 2009.
- [18] Teuvo Kohonen. Improved versions of learning vector quantization. In *Proceedings of the 3rd International Joint Conference on Neural Networks*, pages 545–550. IEEE, 1990.
- [19] Gerhard Kubik. Pattern perception and recognition in african music. *The performing arts: music and dance*, pages 221–49, 1979.
- [20] Miron B. Kursa and Witold R. Rudnicki. Feature selection with the boruta package. *Journal of Statistical Software*, 36(11):1–13, 2010.
- [21] Olivier Lartillot. Multi-dimensional motivic pattern extraction founded on adaptive redundancy filtering. *Journal of New Music Research*, 34(4):375–393, 2005.
- [22] Olivier Lartillot. PatMinr: In-depth motivic analysis of symbolic monophonic sequences. *Music Information Retrieval Evaluation eXchange (MIREX 2014)*, 2014.
- [23] Fred Lerdahl and Ray S. Jackendoff. *A generative theory of tonal music*. MIT press, 1985.
- [24] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems*, pages 431–439, 2013.
- [25] Cory McKay. *Automatic Music Classification with jMIR*. PhD thesis, McGill University, 2010.
- [26] David Meredith. COSIATEC and SIATECCompress: Pattern discovery by geometric compression. *Music Information Retrieval Evaluation Exchange (MIREX 2013)*, 2013.
- [27] David Meredith. Music analysis and point-set compression. *Journal of New Music Research*, 44(3):245–270, 2015.
- [28] David Meredith. Using SIATECCompress to discover repeated themes and sections in polyphonic music. *Music Information Retrieval Evaluation Exchange (MIREX 2016)*, 2016.
- [29] David Meredith, Kjell Lemström, and Geraint A. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4):321–345, 2002.

- [30] Daniel Müllensiefen. Fantastic: Feature analysis technology accessing statistics (in a corpus): Technical report. *Goldsmiths University of London*, 2009.
- [31] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Proceedings of the 15th Advances in Neural Information Processing Systems*, pages 841–848, 2002.
- [32] Oriol Nieto and Morwaread M. Farbood. Identifying polyphonic patterns from audio recordings using music segmentation techniques. *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pages 411–416, 2014.
- [33] Sandeep Patel and A James Barkovich. Analysis and classification of cerebellar malformations. *American Journal of Neuroradiology*, 23(7):1074–1087, 2002.
- [34] Matevž Pesek, Aleš Leonardis, and Matija Marolt. Symchman unsupervised approach for pattern discovery in symbolic music with a compositional hierarchical model. *Applied Sciences*, 7(11):1135, 2017.
- [35] Matevž Pesek, Urša Medvešek, Aleš Leonardis, and Matija Marolt. Symchm: a compositional hierarchical model for pattern discovery in symbolic music representations. *11th Annual Music Information Retrieval eXchange (MIREX15). Malaga*, pages 1–3, 2015.
- [36] Iris Yuping Ren. Closed patterns in folk music and other genres. *Proceedings of the 6th International Workshop on Folk Music Analysis*, pages 56–58, 2016.
- [37] Iris Yuping Ren, Hendrik Vincent Kooops, Anja Volk, and Wouter Swierstra. In search of the consensus among musical pattern discovery algorithms. *Proceedings of the 18th International Society for Music Information Retrieval*, pages 671–680, 2017.
- [38] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.
- [39] Grace Rubin-Rabson. Studies in the psychology of memorizing piano music: A comparison of the whole and the part approach. *Journal of Educational Psychology*, 31(6):460, 1940.
- [40] Herbert A. Simon and Richard K. Sumner. Machine models of music. chapter Pattern in Music, pages 83–110. MIT Press, Cambridge, MA, USA, 1992.
- [41] Wai Man Szeto and Man Hon Wong. A graph-theoretical approach for pattern matching in post-tonal music analysis. *Journal of New Music Research*, 35(4):307–321, 2006.
- [42] Jan Van Balen. *Audio description and corpus analysis of popular music*. PhD thesis, Utrecht University, 2016.
- [43] Peter van Kranenburg, Berit Janssen, and Anja Volk. The Meertens Tune Collections: The Annotated Corpus (MTC-ANN) versions 1.1 and 2.0.1. *Meertens Online Reports*, 2016(1), 2016.
- [44] Gissel Velarde and David Meredith. A wavelet-based approach to the discovery of themes and sections in monophonic melodies. *Music Information Retrieval Evaluation Exchange (MIREX 2014)*, 2014.
- [45] Anja Volk and Peter Van Kranenburg. Melodic similarity among folk songs: An annotation study on similarity-based categorization in music. *Musicae Scientiae*, 16(3):317–339, 2012.
- [46] Cheng-i Wang, Jennifer Hsu, and Shlomo Dubnov. Music pattern discovery with variable markov oracle: A unified approach to symbolic and audio representations. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pages 176–182, 2015.

GENERALIZED SKIPGRAMS FOR PATTERN DISCOVERY IN POLYPHONIC STREAMS

Christoph Finkensiep

Markus Neuwirth

Martin Rohrmeier

École Polytechnique Fédérale de Lausanne

{christoph.finkensiep, markus.neuwirth, martin.rohrmeier}@epfl.ch

ABSTRACT

The discovery of patterns using a minimal set of assumptions constitutes a central challenge in the modeling of polyphonic music and complex streams in general. Skipgrams have been found to be a powerful model for capturing semi-local dependencies in sequences of entities when dependencies may not be directly adjacent (see, for instance, the problems of modeling sequences of words or letters in computational linguistics). Since common skipgrams define locality based on indices, they can only be applied to a single stream of non-overlapping entities. This paper proposes a generalized skipgram model that allows arbitrary cost functions (defining locality), efficient filtering, recursive application (skipgrams over skipgrams), and memory efficient streaming. Further, a sampling mechanism is proposed that flexibly controls runtime and output size. These generalizations and optimizations make it possible to employ skipgrams for the discovery of repeated patterns of close, nonsimultaneous events or notes. The extensions to the skipgram model provided here do not only apply to musical notes but to any list of entities that is monotonic with respect to a given cost function.

1. INTRODUCTION

Discovering relevant patterns in a given corpus of musical pieces is a central problem for music modeling and music information retrieval (MIR) and is crucial for a range of applications from search to stylistic modeling. While there exist many approaches for modeling monophonic melodies [8, 7], polyphony constitutes a persistent challenge due to the vast amount of latent structural patterns that occur on multiple levels. These patterns involve surface ornaments and accompaniment figurations, contrapuntal configurations, latent polyphony comprising multiple interleaved voices, and harmonic and voice-leading schemata.

While many of the underlying patterns are themselves relatively simple, identifying these patterns is challenging, because it involves distinguishing the relevant notes while

ignoring others. In addition, many patterns do not specify notes exactly but leave some flexibility when being instantiated, especially concerning timing. Finally, multiple patterns may co-occur simultaneously or in an interleaved manner.

When modeling the latent structure of polyphony, it is important to find the characteristic properties of the structure to be modeled. Therefore, it is advantageous to start from a model with minimal assumptions about the target structure and add assumptions to the basic model until the desired patterns are found. This way, the properties of the modeled structure are always clear and well-separated from the assumptions inherent in the underlying representation.

There are a variety of methods for modeling sequential data with minimal assumptions, such as those developed in computer linguistics, that treat the data as a single stream of events. However, these cannot be straightforwardly applied to polyphonic data without adding further implicit assumptions or removing information contained in the original data. Therefore, a generalization of skipgrams [4] is developed in this paper that is applicable to a stream of polyphonic notes that need not be explicitly presented as separate voices. This model is applied to a musical corpus for the discovery of polyphonic patterns.

In the remainder of this paper, we first discuss related work in more detail (Section 2); we then describe our approach for generalized skipgrams (Section 3); finally, we describe and discuss our empirical evaluations (Sections 4 and 5).

2. RELATED WORK

Among polyphonic structures, *voice-leading schemata* are particularly prominent in recent research [3]. Schemata can be understood as structural building blocks that can be elaborated in multiple ways. They are defined as fixed patterns of two to four voices where the soprano and bass constitute specific patterns relative to the key of the piece (or a segment within that piece) and may be supplemented with one or two middle voices. The core challenge from an MIR perspective is that the structural elements in each stage of the sequence need not occur simultaneously, owing to highly flexible note elaborations. Thus, instances of schemata in the music are “semi-local”. An example of this problem can be seen in Figure 1. The underlying schema consists of four stages and is elaborated by neighbor and



© Christoph Finkensiep, Markus Neuwirth, Martin Rohrmeier. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Christoph Finkensiep, Markus Neuwirth, Martin Rohrmeier. “Generalized Skipgrams for Pattern Discovery in Polyphonic Streams”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

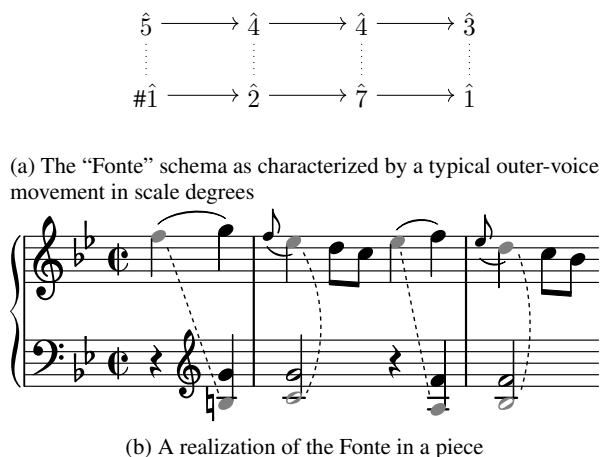


Figure 1: An example of a voice-leading schema

passing notes. The first stage consists of non-overlapping notes, so there is no point in time where both notes sound together. The same applies for the third stage.

This semi-locality property of schema patterns can be met by formalizing an extension of the *skipgram* formalism, which has been successfully applied in linguistics to sequences of words or letters (for a review see [4]). Skipgrams in the original version can only be applied to "monophonic" streams like text, melodies or slices of polyphonic music, as has been done in [10]. The generalized version of skipgrams as proposed in the present paper allows not only the application to truly polyphonic streams but also recursive application, which can be used to build nested structures like schema patterns.

Previously, polyphonic music was mainly modeled using slicing techniques, i.e., cutting the piece vertically at each note onset or offset. In [10], common, index-based skipgrams as well as an onset-time-based variant are applied to slices reduced to a "voice-leading type" representation, similar to the representation used here (for more details see Section 4.2). The approach presented in this paper takes the idea two steps further by generalizing the cost function (allowing non-slice representations as input) and by building even the vertical structure with this generalized skipgram method, in addition to the horizontal structure.

Multiple viewpoint systems (e.g., [1, 2, 12, 11]) take a sequence of slices and derive sequence features, or "viewpoints", from it. Polyphonic structure is modeled by including information about the continuation of notes across slices. For prediction, n -grams of all lengths are combined by comparing all suffixes of a given gram to other grams of the corresponding length. However, slicing techniques are generally problematic when grouping non-overlapping notes, as these are not contained in any single slice.

An alternative to slices is suggested in [6] where polyphonic music is encoded as a set of data points in a multidimensional space. Accordingly, patterns are orthogonal projections (i.e., considering only some features, not all) of subsets of the data points that can be translated to a different position (in both pitch and time). This translation oper-

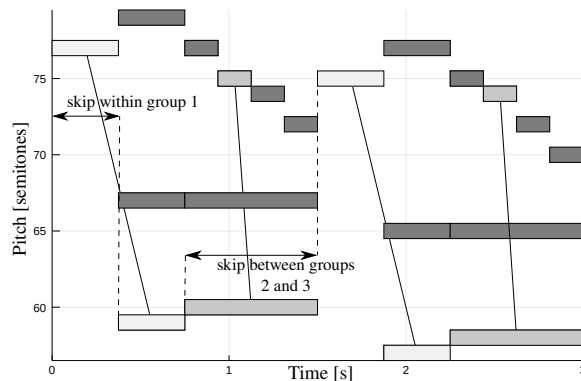


Figure 2: An example of applying skipgrams to polyphonic music displayed in a piano-roll visualization. The highlighted notes are members of the skipgram, the stages are indicated by solid lines between notes belonging to the same stage. The skip cost within and between stages is given by inter-onset intervals. This 2×4 skipgram represents the same pattern as the polyphonic schema in Figure 1.

ation, however, permits only exact matches in the selected dimensions and cannot account for temporally varied patterns.

3. GRAM-BASED METHODS AND GENERALIZED SKIPGRAMS

Gram-based methods extract short sequences of entities from a longer stream of entities (e.g., words, letters, or notes). The most basic gram model, the n -gram, is just a consecutive subsequence in the input stream that has n elements. Skipgrams extend the n -gram idea by allowing non-adjacent subsequences that "skip" up to k elements [4].

Both n -grams and skipgrams assume that the distance between entities is determined by their *position in the stream*. While this assumption might be reasonable for text, it is problematic for other applications that involve general temporal streams, in particular streams of musical events such as notes. Therefore, it is desirable to measure the distance between events (notes) based on their timing information, i.e., onset, offset, and duration. Second, while notes might be simultaneous in a score, they occur sequentially in a stream or list-of-notes representation, which becomes problematic if distance is measured by index.

Sears et al. [10] avoid the latter problem by operating on a slice representation of a piece, in which slices have a unique onset and do not overlap. They partially solve the first problem by replacing the maximal number of skips with a maximal inter-onset-interval, i.e., a time-based distance measure. Our paper presents a generalization of skipgrams to arbitrary pairwise cost functions for streams that are monotonic with respect to the cost function, which allows both efficient implementation and streams of overlapping entities.

Consider Figure 2, a piano-roll representation of a poly-

phonic piece of music. The notes that make up a single stage of a voice-leading schema might not be simultaneous, but should be close together. Given the notes of the piece as a list of triples (*onset, offset, pitch*), candidates for a schema stage can be found by considering all groups of notes (pairs, in the case of two voices) that lie within a certain distance. In the traditional skipgram approach, this distance would be measured by the index of each note in the list. However, in the case of voice-leading schemata, it is more meaningful to measure this distance with respect to the timing of the notes, e.g., as the distance between onsets or the distance between the offset of the earlier and the onset of the latter note.

Since a voice-leading schema consists of several consecutive stages, it is natural to apply this more general idea of skipgrams again, now to the list possible stages. As with notes, the distance between two stages can be defined in several ways, e.g., as the distance from the beginning of the first stage to the beginning of the second, or the amount of time between the stages. In the following section, an algorithm that enumerates skipgrams over streams of arbitrary objects for arbitrary definitions of distance is presented along with some useful extensions.

3.1 The Generalized Skipgram Algorithm

The basic algorithm for generalized skipgrams is shown in Algorithm 1. It takes a stream of objects (e.g., notes or schema stages), an upper bound on the allowed “skip” k , the length of the generated skipgrams n , and a cost function c . The cost function is used to represent the distance between two objects: the combined cost across a skipgram must not be greater than k . While it traverses the input stream, a set of prefixes (incomplete skipgrams) is maintained. For each prefix that can be extended by the current element without increasing the total cost beyond k , the extended version is added to the prefix set. Prefixes of length n are added to the output and removed from the set of prefixes. Finally, the current element starts a new prefix.

In order to keep the set of prefixes small, a prefix is removed as soon as extending it increases the cost beyond k . As long as the stream is sorted in a way that every subsequent element would increase the cost of the prefix even further, this optimization does not discard valid skipgrams. That means the input stream *input* must satisfy

$$\forall x < y < z \in \text{input} : c(x, y) \leq c(x, z),$$

where $x < y$ denotes that x appears before y in *input*.

The cost function can handle the distance in several ways. For example, if the distance between the first and the last note in a skipgram should be limited, then the cost equals the distance between the onsets of two neighboring notes in the skipgram. On the other hand, if the distance between two neighboring notes is to be limited, the cost can be defined non-continuously as 0 if the notes are within the allowed distance and $k + 1$. This way, the combined cost is 0, except when a single neighbor pair of notes is too far apart, in which case it exceeds k .

Algorithm 1 The basic algorithm for enumerating generalized skipgrams.

```

1: function SKIPGRAMS(input,  $k$ ,  $n$ ,  $c$ )
2:    $prefs \leftarrow \{\}$ 
3:    $output \leftarrow []$ 
4:    $cost(p, x) = \sum_{i=1}^{l-1} c(p_i, p_{i+1}) + c(p_l, x)$ 
5:   for  $x \leftarrow \text{input}$  do
6:      $open \leftarrow \{p \mid p \in prefs, cost(p, x) \leq k\}$ 
7:      $ext \leftarrow \{p \circ x \mid p \in open\}$ 
8:      $append(output, [p \mid p \in ext, |p| = n])$ 
9:      $prefs \leftarrow open \cup \{p \mid p \in ext, |p| < n\} \cup \{x\}$ 
10:  end for
11:  return  $output$ 
12: end function

```

Note that the skipgram algorithm traverses the input stream exactly once, so streaming it is straightforward. Similarly, the output can be streamed instead of collected, either using some form of concurrency and a channel between the output of the skipgram generator and some consumer, or non-concurrently using an iterator pattern.

3.2 Early Filtering

If the list of generated skipgrams will be filtered for some property (e.g., only selecting notes that do not overlap or that match a given schema prototype), it is desirable to filter out prefixes that cannot be completed to satisfy the predicate as early as possible, instead of generating all of its completions first. Therefore, an extension of Algorithm 1 additionally takes a predicate *pred*, which it applies to every generated prefix. Every prefix that does not satisfy *pred* is removed. As with the cost function, this predicate can be defined freely.

3.3 Stable Ordering

Algorithm 1 adds its output in the order in which prefixes are completed. As a consequence, the output stream does not retain the order of the first element in each skipgram with respect to the input order. Instead the input order is reflected in the last element of each skipgram.

For some applications, it might be desirable to keep the order of the first elements intact. For example, when computing skipgrams of skipgrams (e.g., first groups of notes, then sequences of groups), the second skipgram pass expects its input to be monotonic with respect to the cost function. In the case of note groups, this will likely depend on the earliest onset in the group. The first skipgram pass takes a list of notes ordered by onset, but the note groups it returns will not be ordered by the onset of their first (and therefore earliest) note but by the onset of their last note.

In general, the appropriate order of skipgrams can be obtained by generating the whole list of skipgrams and sorting it, but this would destroy the streaming property of the algorithm. As the number of skipgrams grows quickly, it might not even be feasible to keep all skipgrams in memory. Furthermore, as k is intended to limit the range of skipgrams, a skipgram can only be displaced as much as k ,

so the array will almost be sorted, and sorting can be done on the fly.

Stable ordering can be ensured efficiently by holding back completed skipgrams (instead of adding them to the output as soon as they are generated) until no new skipgrams can be generated that should precede them. This can be achieved by using a priority queue to hold the finished but not yet released skipgrams in the correct order. In each iteration, the open prefixes are searched for the “oldest” first element. Then, all skipgrams in the output queue starting with an element not younger than this oldest initial prefix element are released. If the output needs to be ordered lexicographically, the queue content must be compared not to the oldest initial element but to the complete lexicographically oldest open prefix. The queue can be updated efficiently by sorting the newly generated prefixes and merging the resulting list with the existing queue as in a merge sort.

3.4 Sampling Skipgrams

The number of generated skipgrams as well as the asymptotic runtime of the algorithm are difficult to estimate, as they depend on the number of elements within a range of k or the number of currently open prefixes, respectively, at any point in the stream. This, in turn, depends on the combination of input and cost function, so no general statement about runtime and space complexity can be made without knowing both. In the worst case, generalized skipgrams consist of all subsets of length n from the list of L entities, generating $\binom{L}{n}$ skipgrams.

Because this amount of skipgrams is costly to enumerate and process, an alternative is to uniformly draw samples from the list of all skipgrams. For a given probability p , a biased coin is tossed for each skipgram, which determines whether the skipgram is selected or not. If this is done after the skipgram is completely generated, all skipgrams must be enumerated once, so computation time is saved only during consumption but not production. Conversely, one could toss the coin for each new prefix of length 1. This way, all extensions of a discarded prefix need not be computed, which saves computation time but also removes a whole family of related skipgrams from the output.

A third approach combines the other two by tossing a coin each time a prefix is extended. As this happens $n - 1$ times for a prefix of length n (not counting the creation of the initial length 1 prefix), so the coin is biased not with p but with $p' = \sqrt[n]{p}$. This way, a skipgram is only included if all of its prefix extensions succeed, i.e., with probability $(p')^{n-1} = p$. Furthermore, computation time can be saved by discarding short prefixes, while variety is preserved by also discarding prefixes in later stages.

With this method, it is not possible to uniformly draw a fixed number of samples. However, the expected number of samples can be estimated by choosing a small p and extrapolating the resulting amount of sampled skipgrams. The total number of skipgrams N can be estimated similarly, as the number of sampled skipgrams is expected to

be Np .

4. SKIPGRAMS ON POLYPHONIC MUSIC

4.1 Dataset

The described method is applied to 17 piano sonatas by Wolfgang Amadeus Mozart in MIDI format.¹ A schema has 2 or 3 voices and consists of 2 to 4 stages. These *dimensions* of a schema are notated as voices \times stages or $n_v \times n_s$, and the sampling parameters p_v and p_s are adapted to these dimensions. The skip limit within a stage k_v is one bar in total, the limit between the stages k_s is also one bar, but per pair of stages.

4.2 Method

For the discovery of musical schemata, three assumptions are made. First, schemata are semi-local structures, that is, they extend over a limited range of time. This property is inherent in the skipgram formalism with an appropriate cost function. Second, they consist of a horizontal sequence of pseudo-vertical structures, which consist of a fixed number of possibly non-simultaneous or even non-overlapping notes. Third, patterns are characterized by their pitch content, not by their temporal properties. This pitch content is subject to certain equivalences, e.g., regarding transposition of the whole pattern or the exact octave of each pitch.

In order to find vertical structures in polyphonic pieces, skipgrams can be applied to a stream of notes. Each note has a pitch, an onset and an offset, and the notes are sorted by their onsets. For this purpose, the cost function is the difference between the onsets of two notes in the skipgram

$$c_v(n_1, n_2) = \text{onset}(n_2) - \text{onset}(n_1),$$

which in sum amounts to the onset difference between the earliest and the latest note in the group. This allows notes to overlap but restricts their temporal distance. The stable variant of the skipgram algorithm is used to ensure that the output stream is again ordered by (earliest) onset. The number of notes n_v in the first pass can be regarded as the number of voices in the vertical structure.

A second skipgram pass builds length n_s sequences of vertical structures by taking the output of the first pass as the new input. Since these structures should be horizontally organized, temporal overlap between their stages is forbidden (by defining an appropriate *pred*). The amount of time between the onsets of slices is restricted by a step function that admits a skip up to k_s for each pair of neighbors:

$$c_s(s_1, s_2) = \begin{cases} 0 & \text{if } \text{onset}(s_{2,1}) - \text{onset}(s_{1,1}) \leq k_s \\ k_s + 1 & \text{otherwise.} \end{cases}$$

Due to the large amount of skipgrams generated, the sampling parameters are adapted to the number of voices (p_v)

¹Encoded by Craig Sapp in Kern format and converted to MIDI. Available at <https://github.com/craigsapp/mozart-piano-sonatas>.

n_v	n_s	p_s	p_v	sampled	total	cov
2	2	1.0	1.0	$3.30 \cdot 10^8$	10^8	1.0
2	3	1.0	0.001	$9.92 \cdot 10^7$	10^{11}	0.99998
2	4	1.0	10^{-6}	$3.77 \cdot 10^7$	10^{13}	0.30
3	2	0.1	1.0	$3.53 \cdot 10^8$	10^{10}	0.9997

Table 1: The combinations of parameters used to generate the results. For each combination, the sampled and the rough estimated total number of skipgrams, as well as the coverage are given. Coverage is the ratio of the number of skipgram classes encountered and the number of possible classes for the given dimensions ($12^{n_v n_s - 1}$).²

and stages (p_s). An example of such a nested skipgram is shown in Figure 2.

The pitch content of the resulting structures is summarized by summing the occurrences of skipgrams with similar pitch content (“skipgram classes”). Pitch combinations are grouped by sorting the pitches in each stage in ascending order, removing octave information (pitch classes), and transposing every pitch class relative to the lowest note of the first stage. For example, the sequence $(f, c', a') \rightarrow (e, c', g')$ would be encoded as $(0, 7, 4) \rightarrow (11, 7, 2)$. This is similar but not identical to the voice-leading type representation used in [10], which additionally removes the order of the pitches and the magnitude of each pitch class. Since the focus here is on polyphonic voice-leading schemata, both order and magnitude are retained. For n_v voices and n_s stages, there are $12^{n_v n_s - 1}$ possible skipgram classes, as the transposition step always sets the first bass note to 0.

5. RESULTS AND DISCUSSION

5.1 Results

Table 1 gives an overview of the used parameter combinations. For each set of parameters, it shows the number of generated skipgrams, the estimated number of total skipgrams, as well as the coverage, which is the ratio of the number of encountered skipgram classes and the number of possible classes for the given dimensions. Good coverage is important for prediction tasks, where the *zero-frequency problem* occurs (i.e., where a prediction context has not been encountered at least once, see [10]).

As the outer-voice movement is most characteristic of voice-leading schemata, the most general insight can be provided by two-voiced skipgram classes, which also have a good coverage for reasonable computational effort. Additionally, the 3×2 skipgrams were generated to observe the effect of increasing the number of voices on the found patterns. Larger dimensions did not produce a sufficient coverage due to computational constraints. The total number of skipgrams without sampling can be estimated by

² Runtimes ranged from a few minutes to several hours. While parallelization was straightforward by splitting the dataset, memory usage was problematic and prevented generating skipgrams with larger dimensions.

	class	abs	rel
1	$(0, 3) \rightarrow (0, 2) \rightarrow (10, 2) \rightarrow (10, 0)$	1190	0.32
2	$(0, 3) \rightarrow (3, 3) \rightarrow (0, 3) \rightarrow (3, 3)$	1029	0.27
3	$(0, 1) \rightarrow (10, 1) \rightarrow (8, 0) \rightarrow (8, 10)$	1009	0.27
4	$(0, 0) \rightarrow (5, 0) \rightarrow (0, 0) \rightarrow (5, 0)$	976	0.26
5	$(0, 0) \rightarrow (9, 0) \rightarrow (5, 0) \rightarrow (0, 0)$	964	0.26
6	$(0, 3) \rightarrow (3, 3) \rightarrow (8, 3) \rightarrow (3, 3)$	937	0.25
7	$(0, 1) \rightarrow (10, 1) \rightarrow (0, 0) \rightarrow (8, 10)$	934	0.25
8	$(0, 0) \rightarrow (9, 0) \rightarrow (0, 0) \rightarrow (9, 0)$	921	0.24
9	$(0, 0) \rightarrow (10, 1) \rightarrow (10, 0) \rightarrow (8, 10)$	902	0.24
10	$(0, 3) \rightarrow (0, 1) \rightarrow (10, 1) \rightarrow (10, 0)$	897	0.24

Table 2: The 10 most frequent 2×4 skipgram classes that have no repeating stages

multiplying the sampled skipgrams with $p_s(p_v^{n_s})$. p_v needs to be exponentiated, as it factors in the output probability of a skipgram on each of its stages.

Table 3 shows the 10 most frequent skipgram classes found for each set of parameters. As the most frequent patterns mainly indicate arpeggiation patterns (see Section 5.2), an additional filter is applied to the 2×4 skipgrams, which forbids the repetition of a stage. The resulting 10 most frequent patterns are shown in Table 2.

In addition to enumerating (or sampling from) all skipgrams in the corpus, the generalized skipgram algorithm can be used as a pattern matcher or schema finder. Figure 2 shows the first skipgram matching a two voiced pattern $(0, 6) \rightarrow (1, 4) \rightarrow (10, 4) \rightarrow (11, 3)$ in the third movement of Mozart’s third piano sonata (from which the example in Figure 1 is taken), found by enumerating the pieces 2×4 skipgrams. Pattern matching can be performed efficiently by using the early filtering mechanism described in Section 3.2 to remove prefixes that cannot match.

5.2 Discussion

As Table 1 shows, even with moderate sampling the coverage is excellent for smaller dimensions. As the dimensions increase and the sampling probability decreases, the coverage rapidly decreases as well, because the large number of possible pitch combinations conflicts with the increased need for reducing the computational effort via sampling. For example, for 2 voices and 4 stages, there are $12^7 \approx 3.58 \cdot 10^7$ possible skipgram classes, but the total number of sampled skipgrams was only $3.77 \cdot 10^7$. In principle, however, the flexibility of nested skipgrams provides a good coverage, confirming the results in [10] for flat skipgrams. For larger problem instances of up to 4×4 skipgrams, the computational problem can be solved by good parallelization and sufficient computation resources.

Based on frequency, the patterns found in the corpus are dominated by interval combinations as they appear in major and minor triads, followed by simple step-wise relations. This cannot be explained by the fact that the music of Mozart is mostly triadic to a large extent, as the stages of a single skipgram are strictly non-simultaneous.

In music that is triadic but consists of sequences of non-arpeggiated, non-repeating chords, the stages of a skipgram will be taken from different chords. Thus, the found patterns reveal the usage of harmonic *surface patterns* such as Alberti bass. This becomes especially clear from variants of $(0,0) \rightarrow (5,0)$ and $(0,0) \rightarrow (9,0)$, which are consistently ranked very high and indicate a prevalence of the fifth in combination with the third or the root of a major triad, resembling the Alberti bass pattern.

In contrast, the filtered patterns shown in Table 2 mainly consist of instances of the typical voice-leading pattern of descending 3-2 suspension sequences. This pattern is used as a typical elaboration procedure in several voice-leading schemata. This finding shows that nested skipgrams are very well capable of representing polyphonic structures. The remaining question is how to automatically distinguish surface patterns from patterns on higher structural levels in the generated skipgrams.

6. CONCLUSION

The results clearly show that the generalized skipgram formalism is capable of modeling streams of events that have a *non-flat* shape, such as streams of notes in polyphonic music. The monotonicity property explained in Section 3.1 is a general criterion for the applicability of the presented algorithm. Thus, generalized skipgrams can prove useful for a wide range of problems from various domains other than music that deal with sequential but overlapping data.

With respect to polyphonic music, generalized skipgrams provide a powerful mechanism for accessing polyphonic structure, solving the problem of building vertical structures from non-overlapping notes. Hence, a potentially powerful application of generalized skipgrams is to use them as the basic representation in variety of other methods that provide rich pattern languages, replacing the currently used sequence-of-slices structure. For example, it is possible to apply viewpoint techniques to the skipgrams generated from a polyphonic stream.

Finally, the skipgram approach requires very little assumptions on its own, but can easily be extended to filter for more advanced, theoretically or empirically motivated properties. The discovery of schema-like patterns, for example, will require to add appropriate filters that separate surface from middleground patterns. This helps to advance the understanding of the essential properties of schemata, as the added assumptions can be clearly separated from the ones inherent in the skipgram representation.

7. ACKNOWLEDGEMENTS

The research presented in this paper is generously supported by the Volkswagen Foundation and Claude Latour. We also thank the anonymous reviewers for their helpful comments.

	class	abs	rel
1	$(0,0) \rightarrow (0,0)$	3.3e6	10.0
2	$(0,0) \rightarrow (5,0)$	1.38e6	4.18
3	$(0,0) \rightarrow (0,5)$	1.34e6	4.06
4	$(0,0) \rightarrow (9,0)$	1.34e6	4.06
5	$(0,0) \rightarrow (7,0)$	1.32e6	4.01
6	$(0,7) \rightarrow (7,7)$	1.31e6	3.96
7	$(0,5) \rightarrow (0,0)$	1.28e6	3.89
8	$(0,3) \rightarrow (3,3)$	1.27e6	3.86
9	$(0,0) \rightarrow (0,7)$	1.22e6	3.7
10	$(0,0) \rightarrow (10,0)$	1.21e6	3.68
1	$(0,0) \rightarrow (0,0) \rightarrow (0,0)$	121073	1.22
2	$(0,0) \rightarrow (0,0) \rightarrow (9,0)$	44143	0.44
3	$(0,0) \rightarrow (0,0) \rightarrow (5,0)$	43764	0.44
4	$(0,0) \rightarrow (5,0) \rightarrow (0,0)$	40859	0.41
5	$(0,3) \rightarrow (3,3) \rightarrow (3,3)$	40543	0.40
6	$(0,7) \rightarrow (7,7) \rightarrow (7,7)$	39470	0.39
7	$(0,0) \rightarrow (9,0) \rightarrow (0,0)$	39056	0.39
8	$(0,0) \rightarrow (0,0) \rightarrow (10,0)$	34975	0.35
9	$(0,0) \rightarrow (0,0) \rightarrow (0,5)$	29343	0.29
10	$(0,0) \rightarrow (0,0) \rightarrow (7,0)$	28667	0.28
1	$(0,0) \rightarrow (0,0) \rightarrow (0,0) \rightarrow (0,0)$	6381	0.169
2	$(0,0) \rightarrow (0,0) \rightarrow (0,0) \rightarrow (9,0)$	2436	0.065
3	$(0,0) \rightarrow (0,0) \rightarrow (0,0) \rightarrow (5,0)$	2386	0.063
4	$(0,0) \rightarrow (0,0) \rightarrow (9,0) \rightarrow (0,0)$	2184	0.058
5	$(0,0) \rightarrow (0,0) \rightarrow (5,0) \rightarrow (0,0)$	2173	0.058
6	$(0,0) \rightarrow (5,0) \rightarrow (0,0) \rightarrow (0,0)$	2075	0.055
7	$(0,3) \rightarrow (3,3) \rightarrow (3,3) \rightarrow (3,3)$	2031	0.054
8	$(0,0) \rightarrow (9,0) \rightarrow (0,0) \rightarrow (0,0)$	2001	0.053
9	$(0,7) \rightarrow (7,7) \rightarrow (7,7) \rightarrow (7,7)$	1918	0.051
10	$(0,0) \rightarrow (0,0) \rightarrow (0,0) \rightarrow (10,0)$	1753	0.046
1	$(0,0,0) \rightarrow (0,0,0)$	461238	1.31
2	$(0,0,0) \rightarrow (9,0,0)$	214255	0.61
3	$(0,3,3) \rightarrow (3,3,3)$	208562	0.59
4	$(0,0,0) \rightarrow (5,0,0)$	205212	0.58
5	$(0,7,7) \rightarrow (7,7,7)$	200170	0.57
6	$(0,3,3) \rightarrow (0,3,3)$	172370	0.49
7	$(0,0,0) \rightarrow (10,0,0)$	162212	0.46
8	$(0,2,2) \rightarrow (2,2,2)$	133658	0.38
9	$(0,7,7) \rightarrow (0,7,7)$	131357	0.37
10	$(0,0,2) \rightarrow (0,0,0)$	128846	0.37

Table 3: The 10 most frequent 2×2 , 2×3 , 2×4 , and 3×2 skipgram classes. Relative frequencies have been scaled by 10^3 and rounded appropriately.

8. REFERENCES

- [1] D. Conklin. “Representation and Discovery of Vertical Patterns in Music”. In: *Music and Artificial Intelligence*. Ed. by C. Anagnostopoulou, M. Ferrand, and A. Smaill. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 32–42. ISBN: 978-3-540-45722-0.
- [2] D. Conklin and M. Bergeron. “Discovery of Contrapuntal Patterns”. In: *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010*. Ed. by J. S. Downie and R. C. Veltkamp. International Society for Music Information Retrieval, 2010, pp. 201–206. ISBN: 978-90-393-5381-3.
- [3] R. Gjerdingen. *Music in the Galant Style*. Oxford, New York: Oxford University Press, Oct. 11, 2007. 528 pp. ISBN: 978-0-19-531371-0.
- [4] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks. “A Closer Look at Skip-Gram Modelling”. In: *Proc. of the 5th International Conference on Language Resources and Evaluation (LREC-2006)*. European Language Resources Association, 2006, pp. 1222–1225.
- [5] O. Lartillot. “In-Depth Motivic Analysis Based on Multiparametric Closed Pattern and Cyclic Sequence Mining”. In: *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*. Ed. by H.-M. Wang, Y.-H. Yang, and J. H. Lee. 2014, pp. 361–366.
- [6] D. Meredith, K. Lemström, and G. A. Wiggins. “Algorithms for Discovering Repeated Patterns in Multidimensional Representations of Polyphonic Music”. In: *Journal of New Music Research* 31.4 (Dec. 1, 2002), pp. 321–345. ISSN: 0929-8215. DOI: 10.1076/jnmr.31.4.321.14162.
- [7] M. T. Pearce and G. A. Wiggins. “Expectation in Melody: The Influence of Context and Learning”. In: *Music Perception: An Interdisciplinary Journal* 23.5 (July 1, 2006), pp. 377–405. ISSN: 0730-7829, 1533-8312. DOI: 10.1525/mp.2006.23.5.377.
- [8] M. Pearce and G. Wiggins. “Improved Methods for Statistical Modelling of Monophonic Music”. In: *Journal of New Music Research* 33.4 (Dec. 1, 2004), pp. 367–385. ISSN: 0929-8215. DOI: 10.1080/0929821052000343840.
- [9] P.-Y. Rolland. “Discovering Patterns in Musical Sequences”. In: *Journal of New Music Research* 28.4 (Dec. 1, 1999), pp. 334–350. ISSN: 0929-8215. DOI: 10.1076/0929-8215(199912)28:04;1-O;FT334.
- [10] D. R. W. Sears, A. Arzt, H. Frostel, R. Sonnleitner, and G. Widmer. “Modeling Harmony with Skip-Grams”. In: *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*. Ed. by S. J. Cunningham, Z. Duan, X. Hu, and D. Turnbull. 2017, pp. 332–338. ISBN: 978-981-11-5179-8.
- [11] R. P. Whorley, C. Rhodes, G. Wiggins, and M. T. Pearce. “Harmonising Melodies: Why Do We Add the Bass Line First?” In: *International Conference on Computational Creativity*. Sydney, 2013, pp. 79–86.
- [12] R. P. Whorley, G. Wiggins, C. Rhodes, and M. T. Pearce. “Development of Techniques for the Computational Modelling of Harmony”. In: *International Conference on Computational Creativity*. Coimbra, Portugal, 2010, pp. 11–15.

COMPARISON OF AUDIO FEATURES FOR RECOGNITION OF WESTERN AND ETHNIC INSTRUMENTS IN POLYPHONIC MIXTURES

Igor Vatulkin Günter Rudolph

TU Dortmund, Department of Computer Science

{igor.vatulkin;guenter.rudolph}@tu-dortmund.de

ABSTRACT

Studies on instrument recognition are almost always restricted to either Western or ethnic music. Only little work has been done to compare both musical worlds. In this paper, we analyse the performance of various audio features for recognition of Western and ethnic instruments in chords. The feature selection is done with the help of a minimum redundancy - maximum relevance strategy and a multi-objective evolutionary algorithm. We compare the features found to be the best for individual categories and propose a novel strategy based on non-dominated sorting to evaluate and select trade-off features which may contribute as best as possible to the recognition of individual and all instruments.

1. INTRODUCTION

Instrument recognition in polyphonic audio signals, when acoustic properties of multiple simultaneously played sources contribute together to the spectrum, corresponds to a very challenging problem in music information retrieval. An unknown number of sound sources, instrument bodies with different characteristics, dissimilarities of overtone distribution across pitches, various playing styles, applied effects, etc. hinder the robust identification of playing instruments. Earlier works started with recognition of individual tones [6, 15]. Several years later first studies on polyphonic instrument recognition were published [5, 7]. In further works, many different and complex methods were proposed, like source separation [14], complex feature engineering [27], or deep neural networks [12].

However, most studies concentrate on the detection of Western instruments in Western classical or popular music. Recently, more attention was paid to analyse also ethnic/world music, for example for onset detection in Carnatic music [22] or rhythm analysis in Indian music [23], but only little work was reported on recognition of ethnic instruments, in particular in polyphonic recordings, or in both Western and ethnic recordings. [10] presented a

study on recognition of 10 Indian stringed, wind, and percussive instruments (sitar, edakkai, indian flute, etc.), however only two instruments were mixed together at the same time. Classification of solo recordings into three families (string, woodwind, percussion) of 9 Pakistani instruments (benju, bainsuri, tabla, etc.) was done in [17]. [2] examined properties of various acoustic features for recognition of five Hindustani instruments. In [25], the performance of models trained for recognition of Western instruments in Western mixtures was validated when applied for polyphonic mixtures with ethnic samples.

The goal of our study was to propose a strategy how audio features can be automatically validated for their ability to detect Western and/or ethnic instruments. We adopt the general experiment setup from [25], starting with a large feature set and applying feature selection for identification of the most relevant features. However, in contrast to [25], we aim at the recognition of not only Western, but also ethnic instruments, and incorporate datasets created with both Western and ethnic samples, making recognition tasks harder, but also allowing the classification models to be more robust and not restricted to data sets created with more similar instruments. Furthermore, we propose a novel strategy based on non-dominated sorting to identify features which are particularly well suited to classify either Western, ethnic, or both categories of instruments.

The remainder of this paper is organised as follows. In Section 2, we introduce the backgrounds of multi-objective feature selection. Section 3 describes the study setup. In Section 4, we discuss the results, compare the features, and present our strategy how the most relevant features for the recognition of Western, ethnic, and both groups of instruments can be identified. We conclude with Section 5.

2. MULTI-OBJECTIVE FEATURE SELECTION

The goal of feature selection (FS) is to identify relevant features and to remove irrelevant and redundant ones. Relevant features contribute to the “best” classification models, so that their removal would decrease the classification quality. Irrelevant features do not capture any important properties of classification categories; if too many of such features are contained in the data set, some of them may be identified by chance as relevant, leading to decreased generalisation ability of models. Redundant features can be removed from the feature set without decrease of classification quality for models trained with this set, because



© Igor Vatulkin, Günter Rudolph. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Igor Vatulkin, Günter Rudolph. “Comparison of Audio Features for Recognition of Western and Ethnic Instruments in Polyphonic Mixtures”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

other features already describe the same properties. For a good introduction into feature selection, we refer to [11].

To evaluate feature sets, some criterion is needed, like classification accuracy, or correlation with the target. In the multi-objective feature selection (MO-FS), several of such criteria are optimised simultaneously:

$$\mathbf{q}^* = \arg \min_{\mathbf{q}} \{m_i(\mathbf{y}, \hat{\mathbf{y}}, \Phi(\mathcal{F}, \mathbf{q})) : i = 1, \dots, K\}, \quad (1)$$

where \mathcal{F} is the complete feature set, $\Phi(\mathcal{F}, \mathbf{q})$ is the selected feature set, \mathbf{q} is the binary vector which indicates features to be selected (zero entry at position i means that i -th feature is not selected), \mathbf{y} are correct labels (categories to predict), $\hat{\mathbf{y}}$ are predicted labels, and m_1, \dots, m_K are K evaluation or *objective* functions, which may measure classification performance (accuracy, precision, recall, etc.) but also other relevant criteria (number of selected features, degree of internal redundancies across features, etc.)

In [25], it is proposed to minimise two criteria: the number of selected features and the balanced classification error, which is defined as follows:

$$e = \frac{1}{2} \left(\frac{FN}{TP + FN} + \frac{FP}{TN + FP} \right), \quad (2)$$

where TP is the number of true positives, TN true negatives, FP false positives, and FN false negatives.

When MO-FS is applied, some feature sets cannot be compared: consider a smaller feature set, which leads to a higher classification error, and a larger feature set, which leads to a lower error; none of these sets can be described as superior to another one. However, some sets may be worse with regard to both criteria, and can be identified with the help of non-dominance relation: feature set \mathbf{q}_1 dominates feature set \mathbf{q}_2 ($\mathbf{q}_1 \prec \mathbf{q}_2$), if and only if

$$\begin{aligned} \forall i \in \{1, \dots, K\} : m_i(\mathbf{q}_1) \leq m_i(\mathbf{q}_2) \text{ and} \\ \exists j \in \{1, \dots, K\} : m_j(\mathbf{q}_1) < m_j(\mathbf{q}_2). \end{aligned} \quad (3)$$

In other words, \mathbf{q}_1 dominates \mathbf{q}_2 , when it is not worse than \mathbf{q}_2 with regard to all criteria, and is better with regard to at least one criterium. Here, we restrict us to minimisation of all criteria; criteria to be maximised can be simply redefined for minimisation (e.g., multiplying them with -1). The goal of MO-FS is to find a *non-dominated front* of incomparable feature sets, which are not dominated by any other feature set.

3. EXPERIMENTAL SETUP

In the experimental setup, we mostly follow [25]. Table 1 lists all instruments used in this study. The instruments which were recognised in classification experiments, are written in normal font. Further instruments, which were present in audio mixtures, but were not considered as classes to be identified, are written in *italic font*. The Western instrument samples are taken from MUMS [4],

RWC [9], and University of Iowa¹ databases, and ethnic from Ethno World 5 Professional & Voices².

The chords are randomly mixed from individual samples as described in [25], however, in contrast to previous work, we have created heterogeneous data sets, so that in each mixture of three to four tones at least one Western and at least one ethnic sample is contained. The *experiment* set consists of 3000 chords. During feature selection, it is divided into the training and optimisation set by means of 5-fold cross-validation. *Training* set is used to train classification models, and *optimisation* set to estimate the balanced classification error e . Other independently mixed 3000 chords are used as *holdout* set to measure the effect of overfitting towards the experiment set.

The audio features comprise acoustic characteristics available for extraction with open-source AMUSE framework [26], including mel frequency cepstral coefficients (MFCCs) [21], root mean square (RMS) of the time signal, various spectral characteristics (centroid, bandwidth, kurtosis, skewness, flux, etc.), chroma [8] and chroma energy normalized statistics (CENS) [20], but also other less frequently used features like characteristics of ERB bands and Bark scale domain [19] or phase domain [18]. Before the extraction, the audio was downsampled to 22,1 kHz mono signal, and the most short-framed features were extracted from 512 samples without overlap; for exact details see [24]. For each feature, three dimensions are stored separately: a value from the middle of the attack interval, from the onset frame (the end of the attack interval equal to the beginning of the release interval), and from the middle of the release interval, where attack and release intervals were previously extracted with MIR Toolbox [16], leading to the complete set of 795 feature dimensions. Classification is done with random forest classifier [13].

The first FS strategy was minimum redundancy-maximum relevance (MRMR) [3], which aims at the minimisation of redundancy between selected features and maximisation of relevance to the target category. The second FS strategy was evolutionary multi-objective feature selection (EMO-FS) with \mathcal{S} -metric selection evolutionary multi-objective algorithm (SMS-EMOA) [1], for further details please see [25].

4. DISCUSSION OF RESULTS

4.1 Performance Analysis

Table 2 provides the summary of results after feature selection. $e_H(\Phi)$ denotes the baseline classification error using the complete feature set. $|\hat{\Phi}_O|$ denotes the cardinality of the feature set with the smallest optimisation error $e_O(\hat{\Phi}_O)$. $e_H(\hat{\Phi}_O)$ denotes the holdout error for that feature set, and $e_H(\hat{\Phi}_H)$ the best holdout error among all output feature sets after feature selection.

Both MRMR and EMO-FS significantly outperform the baseline method which trains models with all features. This means, that FS explicitly makes sense. As it can be

¹ <http://theremin.music.uiowa.edu/MIS.html>

² <http://www.bestservice.de>

Category	Instruments
WESTERN	
Bowed	Cello, viola, violin
Key	Piano
Stringed	Acoustic guitar, electric guitar
Woodwind/brass	Flute, trumpet
ETHNIC	
Bowed	Dilruba, egyptian fiddle, erhu, <i>jinghu opera violin</i> , <i>morin khuur violin</i>
Key	Hohner melodica, scale changer harmonium
Stringed	Balalaika, bandura, banjolin, banjo framus, <i>bouzouki</i> , <i>ceylon guitar</i> , <i>cümbüs</i> , <i>domra</i> , <i>kantele</i> , <i>oud</i> , <i>sitar</i> , <i>tampura</i> , <i>tanbur</i> , <i>saz</i> , <i>ukulele</i>
Woodwind/brass	Bawu, dung dkar trumpet, fujara, <i>pan flute</i> , <i>pinkillo</i> , <i>pivana</i> , <i>shakuhachi</i>

Table 1: Instruments used in this study.

	No FS	MRMR				EMO-FS			
Task	$e_H(\Phi)$	$ \widehat{\Phi}_O $	$e_O(\widehat{\Phi}_O)$	$e_H(\widehat{\Phi}_O)$	$e_H(\widehat{\Phi}_H)$	$ \widehat{\Phi}_O $	$e_O(\widehat{\Phi}_O)$	$e_H(\widehat{\Phi}_O)$	$e_H(\widehat{\Phi}_H)$
WESTERN									
Acoustic guitar	0.4395	10	0.3962	0.3906	0.3906	46	0.3809	0.3885	0.3751
Cello	0.4696	12	0.4295	0.4382	0.4382	37	0.4358	0.4574	0.4404
Electric guitar	0.1704	309	0.1532	0.1424	0.1369	44	0.1238	0.1158	0.1084
Flute	0.4907	3	0.4485	0.4651	0.4516	45	0.4513	0.4675	0.4547
Piano	0.2531	7	0.2148	0.2411	0.2260	54	0.2112	0.2320	0.2237
Trumpet	0.3119	20	0.2516	0.2538	0.2506	64	0.2706	0.2604	0.2488
Viola	0.4968	13	0.4735	0.4857	0.4687	36	0.4417	0.4561	0.4406
Violin	0.4791	8	0.4518	0.4639	0.4632	55	0.4538	0.4605	0.4504
ETHNIC									
Balalaika	0.3976	34	0.3070	0.2987	0.2821	48	0.3113	0.3226	0.2931
Bandura	0.5000	2	0.4653	0.4893	0.4587	50	0.4713	0.4809	0.4689
Banjo framus	0.4909	11	0.3915	0.4252	0.4151	39	0.4184	0.4139	0.3994
Banjolin	0.4827	18	0.3504	0.3895	0.3895	32	0.3661	0.4012	0.3937
Bawu	0.3776	12	0.1814	0.2150	0.1836	66	0.2028	0.2138	0.1963
Dilruba	0.4492	32	0.3769	0.3987	0.3974	59	0.3297	0.3761	0.3503
Dung dkar	0.4213	47	0.3971	0.3487	0.3487	49	0.3478	0.3373	0.2983
Egyptian fiddle	0.3533	79	0.2387	0.2750	0.2420	57	0.1763	0.1984	0.1692
Erhu	0.4507	12	0.3719	0.3766	0.3672	42	0.3609	0.3489	0.3293
Fujara	0.3061	28	0.1945	0.1887	0.1840	51	0.1994	0.2236	0.1959
Melodica	0.3889	33	0.2721	0.3041	0.2926	48	0.2638	0.2898	0.2633
Scale changer harmonium	0.3192	22	0.2342	0.2590	0.2445	43	0.2263	0.2578	0.2149

Table 2: Results after feature selection, details are explained in Section 4.1.

expected, $e_H(\hat{\Phi}_O)$ is often higher than $e_O(\hat{\Phi}_O)$. However, this difference is significant only for ethnic instruments and EMO-FS. The null hypothesis that both errors come from the same distribution is rejected by means of Wilcoxon signed rank test for paired observations (MATLAB function SIGNRANK) only for ethnic instruments/EMO-FS with p-value of 0.0210. For combination Western/EMO-FS, $p = 0.1484$, for Western/MRMR $p = 0.1094$ and ethnic/MRMR $p = 0.0922$. Both MRMR and EMO-FS are comparable: each method leads to a smaller $e_O(\hat{\Phi}_O)$ for exactly a half of Western and a half of ethnic categories, and there is no significant difference between these errors after the application of Wilcoxon rank sum test for unpaired observations (MATLAB function RANKSUM).

4.2 Best Features for Individual Categories

To identify the most relevant features for each category, we may estimate feature ranks as follows. Let N_c be the number of solutions (feature sets) in the non-dominated front after the multi-objective optimisation for category c (recall that the non-dominated front contains the best incomparable solutions, cf. Section 2). Let $q_{k,i}$ be 1 when feature k in non-dominated solution i is selected, and 0 when this feature is not selected. We may count the number of occur-

rences of feature k in the front and normalise this number by the size of the front:

$$r(c, k) = \frac{1}{N_c} \cdot \sum_{i=1}^{N_c} q_{k,i}. \quad (4)$$

Table 3 lists two most relevant features for individual classification tasks using either MRMR (columns 2-5) or EMO-FS (columns 6-9). As MRMR starts with the most relevant feature and only adds further features during the iteration process, $r = 1$ for all 1st best features in that case. Because of the differences in operating methods (EMO-FS explores a significantly larger number of feature sets, but is slower), the two most relevant features are usually not the same. However, for cello, flute, and erhu the best feature is exactly the same for MRMR and EMO-FS. For scale changer harmonium and bawu the 1st best feature for MRMR is the same as the 2nd best feature for EMO-FS.

One observation is that MFCCs seem to play a more important role for the recognition of Western instruments: although all processed MFCC dimensions together account for appr. 17% of the complete feature set, they correspond to 56.25% of all 16 entries for 1st best features (8 entries for each MRMR and EMO-FS) and 18.75% for 2nd best features for Western instruments, but only to 20.83% for

Task	MRMR				EMO-FS			
	1st best feature	r	2nd best feature	r	1st best feature	r	2nd best feature	r
WESTERN								
Ac. guitar	A(MFCC 2)	1	O(Bark scale magn. 6)	0.83	A(Phase domain angles)	0.80	A(Chroma 11)	0.60
Cello	O(MFCC 4)	1	A(Spectral slope)	0.89	O(MFCC 4)	0.50	A(RMS)	0.42
El. guitar	A(1st period. ampl. peak)	1	A(RMS ERB band 2)	0.95	R(Bark scale magn. 19)	0.58	R(Delta MFCC 3)	0.42
Flute	A(MFCC 3)	1	R(MFCC 6)	0.67	A(MFCC 3)	0.55	O(LPC 2)	0.36
Piano	R(MFCC 1)	1	O(RMS ERB band 2)	0.83	O(RMS ERB band 1)	0.67	A(Sum corr. components)	0.56
Trumpet	R(Ampl. 4th spectr. peak)	1	R(Inharmonicity)	0.92	R(Ampl. 5th spectr. peak)	0.73	R(MFCC 2)	0.64
Viola	O(Low energy)	1	A(CENS chroma 11)	0.80	O(MFCC 9)	0.82	R(RMS ERB band 1)	0.55
Violin	A(MFCC 1)	1	A(Var. aver. dist. betw. ZC)	0.80	O(MFCC 5)	0.60	A(RMS ERB band 2)	0.50
ETHNIC								
Balalaika	O(Bark scale magn. 21)	1	A(LPC 3)	0.89	O(Bark scale magn. 21)	0.73	O(LPC 3)	0.53
Bandura	A(Sub-band energy rat. 4)	1	R(MFCC 1)	0.50	A(MFCC 3)	0.86	A(Spectral kurtosis)	0.71
Banjo framus	O(Spectral flux)	1	A(LPC 4)	0.83	A(LPC 4)	0.89	A(ZC rate ERB band 6)	0.67
Banjolin	O(Bark scale magn. 23)	1	O(MFCC 8)	0.92	A(LPC 2)	0.80	O(Sum corr. components)	0.80
Bawu	A(RMS peak num. above mean ampl.)	1	O(Bark scale magn. 6)	0.83	O(RMS peak number)	0.56	A(RMS peak num. above mean ampl.)	0.56
Dilruba	O(MFCC 3)	1	O(MFCC 1)	0.88	A(Spectral extent)	0.88	O(RMS ERB band 2)	0.75
Dung dkar	O(Spectral kurtosis)	1	R(ZC rate ERB band 2)	0.89	R(Ampl. 1st spectr. peak)	0.88	R(LPC 1)	0.63
Egypt. fiddle	A(Phase domain angles)	1	A(MFCC 7)	0.94	A(Spectral flux))	0.69	O(Bark scale magn. 23)	0.62
Erhu	R(MFCC 4)	1	A(RMS peak number)	0.80	R(MFCC 4)	0.70	R(RMS)	0.60
Fujara	A(Max. ampl. chroma)	1	O(Ampl. 4th spectr. peak)	0.91	R(RMS)	0.90	O(Spectral flatness 4)	0.70
Melodica	R(RMS ERB band 8)	1	A(Spectral bandwidth)	0.92	R(MFCC 2)	0.56	A(ZC rate ERB band 6)	0.44
Scale changer harmonium	R(LPC 5)	1	A(MFCC 1)	0.91	R(Phase domain angles)	0.60	R(LPC 5)	0.50

Table 3: Ranks of features for categorisation of individual instruments. A(·): features from middles of attack intervals; O(·): from onset frames; R(·): from middles of release intervals. LPC: linear prediction coefficient; ZC: zero-crossings.

1st and 20.83% for 2nd best ethnic features. Among ethnic instruments, the half of all MFCC occurrences belongs to bowed instruments (dilruba, egyptian fiddle, erhu), and other two belong to key instruments (melodica and scale changer harmonium). This leads to a careful suggestion that the mel spectrum is probably not the best feature domain for ethnic stringed and brass instruments, which deserves further investigations. Among two most relevant features for ethnic instruments, particularly LPCs occur rather frequently (8 times / 16.7% of all entries against 1 time / 3.13% for Western instruments).

With regard to attack/onset/release-envelope, we may observe, that all three extraction frame categories appear frequently in Table 3. However, the attack phase seems to be generally more relevant for all instruments (for Western instruments, 43.75% of all entries belong to features stored from middles of attack intervals, for ethnic, 39.58%). Onset features correspond to 28.13% of Western and 33.33% of ethnic entries, and release features to 28.13% of Western and 27.08% of ethnic entries.

4.3 Best Features for Western and Ethnic Instruments

To compare the significance of features for all Western instruments against all ethnic instruments, we may estimate mean feature ranks across all categories of the same “world”. Let M_W be the number of Western instruments ($M_W = 8$) and M_E of ethnic instruments ($M_E = 12$). Let $q_{c,k,i}$ be 1, when feature k is selected in i -th non-dominated solution of the category c , and 0, when it is not selected. Then, the accumulated rank of feature k for all Western instruments is calculated as:

$$r(W, k) = \frac{1}{M_W} \cdot \sum_{c=1}^{M_W} \left(\frac{1}{N_c} \cdot \sum_{i=1}^{N_c} q_{c,k,i} \right), \quad (5)$$

and, similarly, the accumulated rank of feature k for all ethnic instruments as:

$$r(E, k) = \frac{1}{M_E} \cdot \sum_{c=1}^{M_E} \left(\frac{1}{N_c} \cdot \sum_{i=1}^{N_c} q_{c,k,i} \right), \quad (6)$$

The accumulated rank corresponds to the relative share of selections of a feature k among all non-dominated solutions for all categories of the same world.

Table 4 lists top 10 features for Western and ethnic categories. Additionally to non-dominated fronts from the optimisation set (columns 1,2,5,6), we analyse the importance of features for the independent holdout set (columns 3,4,7,8). As we can observe, the best features for the optimisation set are often the same as for the holdout set, which supports the suggestion, that those features are well suitable for different data sets. Again, we see, that with regard to accumulated ranks, MFCCs appear rather often for Western categories (8 entries) than for ethnic categories (6 entries), and LPCs rather often for ethnic categories (6 entries vs. no entry). EMO-FS selected often RMS for ERB bands among top 10 Western features (9 of 20 corresponding entries vs. no entry for ethnic categories).

To measure the statistical difference between importances of features for both worlds, we validate the following statistical hypothesis H_0 : given the accumulated ranks of top 20 features of one world, we assume that the ranks of the same features but for another world belong to the same distribution. As Western and ethnic categorisation tasks are independent, we validate this hypothesis by means of Wilcoxon rank sum test (RANKSUM function in

MRMR				EMO-FS			
Optimisation set	r	Holdout set	r	Optimisation set	r	Holdout set	r
TOP 10 FEATURES FOR WESTERN CATEGORIES							
A(1st period. ampl. peak)	0.228	R(1st period. ampl. peak)	0.259	R(RMS ERB band 1)	0.182	A(RMS ERB band 2)	0.258
A(MFCC 1)	0.211	A(1st period. ampl. peak)	0.251	A(RMS ERB band 2)	0.177	R(RMS ERB band 1)	0.221
O(Low energy)	0.178	A(MFCC 1)	0.227	A(Max. ampl. chroma)	0.172	A(Max. ampl. chroma)	0.185
R(1st period. ampl. peak)	0.177	O(Low energy)	0.195	A(Phase domain angles)	0.169	R(Var. aver. dist. betw. ZC)	0.176
R(ZC rate ERB band 1)	0.169	A(MFCC 3)	0.191	O(Bark scale magnitude 1)	0.156	R(RMS ERB band 2)	0.173
A(MFCC 3)	0.164	A(Spectral slope)	0.178	A(MFCC 3)	0.152	A(Ampl. 1st spectral peak)	0.167
A(Spectral slope)	0.151	R(MFCC 3)	0.159	A(Sum corr. components)	0.147	O(RMS ERB band 1)	0.157
R(Inharmonicity)	0.135	R(Spectr. centroid ERB 8)	0.150	O(RMS ERB band 1)	0.148	A(Phase domain angles)	0.144
A(ZC rate ERB band 10)	0.135	R(Low energy)	0.144	A(Ampl. 1st spectr. peak)	0.143	O(RMS ERB band 2)	0.143
R(MFCC 1)	0.132	R(MFCC 1)	0.141	R(RMS ERB band 2)	0.141	O(CENS chroma 6)	0.142
ETHNIC							
A(Low energy)	0.203	A(Low energy)	0.155	R(Ampl. 1st spectral peak)	0.199	R(Ampl. 1st spectral peak)	0.203
R(LPC 6)	0.158	A(Sub-band energy ratio 4)	0.148	O(Bark scale magnitude 21)	0.195	O(Bark scale magnitude 21)	0.178
A(Sub-band energy ratio 4)	0.146	R(LPC 6)	0.137	O(RMS peak number)	0.170	O(Spectral extent)	0.161
O(LPC 7)	0.130	A(Phase domain angles)	0.137	R(RMS)	0.170	O(LPC 4)	0.157
A(Phase domain angles)	0.129	O(MFCC 3)	0.135	A(Spectral bandwidth)	0.159	R(Bark scale magnitude 3)	0.154
O(Bark scale magnitudes 6)	0.125	O(Bark scale magnitude 6)	0.128	O(Spectral flux))	0.143	R(RMS)	0.153
O(MFCC 3)	0.121	R(ZC rate for ERB band 2)	0.119	R(RMS peak num. above mean ampl.)	0.141	A(Strength of 6.major key)	0.145
O(Bark scale magnitude 21)	0.104	O(Bark scale magnitude 21)	0.111	R(MFCC 2)	0.137	A(Inharmonicity)	0.135
A(LPC 3)	0.097	O(Spectr. centroid ERB 10)	0.109	R(MFCC 4)	0.137	A(MFCC 6)	0.130
O(Spectr. centroid ERB 10)	0.095	A(LPC 4)	0.108	A(Bark scale magnitude 17)	0.136	O(MFCC 1)	0.130

Table 4: Accumulated ranks of features for categorisation of Western and ethnic instruments. A(-): features from middles of attack intervals; O(-): from onset frames; R(-): from middles of release intervals. LPC: linear prediction coefficient; ZC: zero-crossings.

MATLAB). H_0 is rejected in all cases for both feature selection strategies and both sets (optimisation/holdout). Table 5 contains p-values. This means that top 20 features which are particularly good for recognition of Western instruments are not similarly good for the recognition of ethnic instruments, and vice versa. However, please note that H_0 is rejected only for a limited set of 8 Western and 12 ethnic instruments, even if they were carefully chosen to represent different instrument categories. Further studies with a significantly larger number of instruments may support or weaken this statement.

4.4 Best Features for All Categories

To provide generic recommendations on features which are particularly useful for the recognition of both Western and ethnic instruments, Figure 1 plots the accumulated ranks $r(W, k)$ and $r(E, k)$ of all features. Upper subfigures contain results for MRMR, bottom subfigures for EMO-FS, left subfigures correspond to optimisation set, and right subfigures to holdout set. Dashed lines divide the rank space in three regions. For features in the bottom right region, $r(W, k)$ is at least twice as large as $r(E, k)$. For features in the top left region, $r(E, k)$ is at least twice as large as $r(W, k)$. The ranks of features in the middle region are comparable for Western and ethnic instruments. As we are interested to identify features which are best suited for the classification of all instruments, we marked the first non-dominated front with large filled circles and the second non-dominated front with small filled circles, supported with feature IDs. The mapping of IDs to feature names is provided in Table 6. For MRMR, the features belonging to first non-dominated fronts are low energy, MFCC 1, and 1st periodicity amplitude peak. For

EMO-FS, these features are RMS, Bark scale magnitude 3, RMS for ERB bands 1 and 2, maximal amplitude in the chromagram, and amplitude of the 1st spectral peak.

It is worth to mention that even if our feature vector contains almost 800 dimensions, the features can be extracted from various frame lengths or with varying parameters, and further signal descriptors can be added. Further work is necessary to identify better features for instrument recognition, and our framework provides an automatic strategy to evaluate the suitability of features or their extraction parameters to classify instruments of different categories by means of non-dominance relation.

5. CONCLUSIONS

In this paper, we have applied two feature selection methods for recognition of Western and ethnic instruments in polyphonic audio mixtures. Both methods lead to a significant reduction of the classification error compared to models trained with all features. To measure the relevance of features for individual categories as well as for a set of 8 Western and 12 ethnic categories, we proposed a simple rank measure based on feature occurrence in non-dominated fronts, with the aim to simultaneously minimise the number of features and the classification error. Even if larger feature sets with a smaller error are usually preferable for classification scenarios, also small feature sets with higher errors give valuable insights into relevance of individual features. The statistical comparison of features best suited for recognition of Western instruments against features best suited for recognition of ethnic instruments showed that their performance is significantly different. This empirically supports the suggestion, that many acoustic descriptors developed and optimised for music instru-

H0	MRMR		EMO-FS	
	Optimisation set	Holdout set	Optimisation set	Holdout set
Top 20 Western features are similarly good for ethnic categories	5.69e-08	5.69e-08	2.21e-07	6.70e-08
Top 20 ethnic features are similarly good for Western categories	1.99e-04	1.11e-04	6.67e-06	2.66e-06

Table 5: p-values for comparison of top 20 Western and top 20 ethnic features represented by their accumulated ranks.

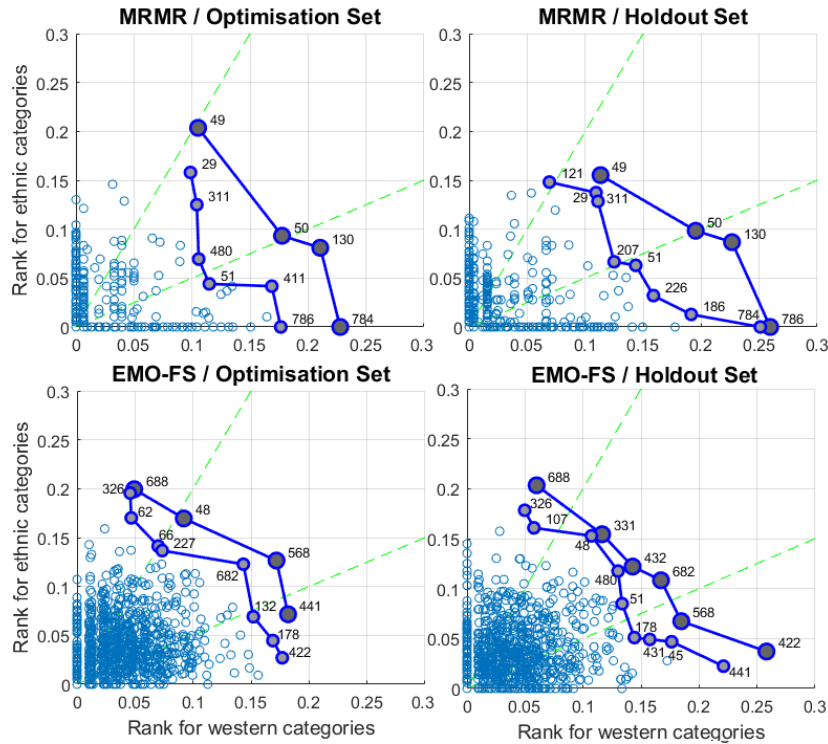


Figure 1: Best (large circles) and 2nd best (small circles) non-dominated features for both Western and ethnic categories. The fronts were estimated for the maximisation of accumulated ranks.

No.	Name	No.	Name	No.	Name
29	R(LPC 6)	132	A(MFCC 3)	422	A(RMS for ERB band 2)
45	R(Var. of aver. dist. between ZC)	178	A(Phase domain angles)	431	O(RMS for ERB band 1)
48	R(RMS)	186	A(MFCC 3)	432	O(RMS for ERB band 2)
49	A(Low energy)	207	O(MFCC 4)	441	R(RMS for ERB band 1)
50	O(Low energy)	226	R(MFCC 3)	480	R(Spectral centroid ERB band 10)
51	R(Low energy)	227	R(MFCC 4)	568	A(Max. ampl. chroma)
62	O(RMS peak number)	311	O(Bark scale magnitude 6)	682	A(Ampl. 1st spectral peak)
66	R(RMS peak number above mean ampl.)	326	O(Bark scale magnitude 21)	688	R(Ampl. 1st spectral peak)
107	O(Spectral extent)	331	R(Bark scale magnitude 3)	784	A(1st periodicity ampl. peak)
121	A(Sub-band energy ratio 4)	411	R(ZC rate for ERB band 1)	786	R(1st periodicity ampl. peak)
130	A(MFCC 1)				

Table 6: Names of features from two best fronts of Figure 1. A(·): features from middles of attack intervals; O(·): from onset frames; R(·): from middles of release intervals. LPC: linear prediction coefficient; ZC: zero-crossings.

ment recognition in Western music are not best suited for the recognition of ethnic instruments.

Another focus of our investigation was to identify those features which are particularly well suited for the recognition of both Western and ethnic instruments. This can be done by means of non-dominated sorting in the two-dimensional rank space. Even if the goal of identifying the best “compromise” features is somewhat contrary to the identification of the best specific features for Western and ethnic instruments, both approaches make sense. Keep-

ing a nearly unlimited number of possible combinations of many world instruments with different effects and playing styles in mind, a good strategy is to start with a sufficiently large set of audio descriptors. In the second time-consuming optimisation step, more efforts can be spent for refining the extraction parameters of these features and development of further ones, which are particularly relevant for a concrete instrument class. With the help of our framework, both tasks can be executed and analysed automatically.

6. REFERENCES

- [1] N. Beume, B. Naujoks, and M. Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.
- [2] A. K. Datta, S. S. Solanki, R. Sengupta, S. Chakraborty, K. Mahto, and A. Patranabis. *Automatic Musical Instrument Recognition*, pages 167–232. Springer Singapore, Singapore, 2017.
- [3] C. H. Q. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal on Bioinformatics and Computational Biology*, 3(2):185–206, 2005.
- [4] T. Eerola and R. Ferrer. Instrument library (MUMS) revised. *Music Perception*, 25(3):253–255, 2008.
- [5] J. Eggink and G. J. Brown. A missing feature approach to instrument identification in polyphonic music. In *Proc. of 2003 IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 553–556. IEEE, 2003.
- [6] A. J. Eronen and A. Klapuri. Musical instrument recognition using cepstral coefficients and temporal features. In *Proc. of IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 753–756. IEEE, 2000.
- [7] S. Essid, G. Richard, and B. David. Instrument recognition in polyphonic music. In *Proc. of 2005 IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 245–248. IEEE, 2005.
- [8] T. Fujishima. Realtime chord recognition of musical sound: a system using common lisp music. In *Proc. of the International Computer Music Conference (ICMC)*, pages 464–467, 1999.
- [9] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Music genre database and musical instrument sound database. In *Proc. of the 4th Int'l Conf. on Music Information Retrieval (ISMIR)*, pages 229–230, 2003.
- [10] S. Gunasekaran and K. Revathy. Fractal dimension analysis of audio signals for indian musical instrument recognition. In *Proc. of the Int'l Conf. on Audio, Language and Image Processing (ICALIP)*, pages 257–261, 2008.
- [11] I. Guyon, M. Nikravesh, S. Gunn, and L. A. Zadeh, editors. *Feature Extraction. Foundations and Applications*, volume 207 of *Studies in Fuzziness and Soft Computing*. Springer, Berlin Heidelberg, 2006.
- [12] Y. Han, J.-H. Kim, and K. Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech & Language Processing*, 25(1):208–221, 2017.
- [13] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2009.
- [14] T. Heittola, A. Klapuri, and T. Virtanen. Musical instrument recognition in polyphonic audio using source-filter model for sound separation. In *Proc. of the 10th Int'l Society for Music Information Retrieval Conference (ISMIR)*, pages 327–332, 2009.
- [15] I. Kaminsky and A. Materka. Automatic source identification of monophonic musical instrument sounds. In *Proc. of IEEE Int'l Conf. on Neural Networks*, volume 1, pages 189–194 vol.1, 1995.
- [16] O. Lartillot and P. Toivainen. MIR in Matlab (II): A toolbox for musical feature extraction from audio. In *Proc. 8th Int'l Conf. on Music Information Retrieval (ISMIR)*, pages 127–130, 2007.
- [17] S. A. Lashari, R. Ibrahim, and N. Senan. Soft set theory for automatic classification of traditional pakistani musical instruments sounds. In *Proc. of Int'l Conf. on Computer Information Science (ICCIS)*, volume 1, pages 94–99, 2012.
- [18] I. Mierswa and K. Morik. Automatic feature extraction for classifying audio data. *Machine Learning Journal*, 58(2-3):127–149, 2005.
- [19] B. C. J. Moore and B. R. Glasberg. A revision of Zwicker's loudness model. *Acta Acustica united with Acustica*, 82(2):335–345, 1996.
- [20] M. Müller, F. Kurth, and M. Clausen. Audio matching via chroma-based statistical features. In *Proc. of the 6th Int'l Conf. on Music Information Retrieval (ISMIR)*, pages 288–295, 2005.
- [21] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Upper Saddle River, 1993.
- [22] J. Sebastian and H. A. Murthy. Onset detection in composition items of carnatic music. In *Proc. of the 18th Int'l Society for Music Information Retrieval Conf.*, pages 560–567, 2017.
- [23] A. Srinivasamurthy. *A Data-driven Bayesian Approach to Automatic Rhythm Analysis of Indian Art Music*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2016.
- [24] I. Vatulkin. *Improving Supervised Music Classification by Means of Multi-Objective Evolutionary Feature Selection*. PhD thesis, Dep. of Computer Science, TU Dortmund, 2013.
- [25] I. Vatulkin. Generalisation performance of western instrument recognition models in polyphonic mixtures with ethnic samples. In *Proc. of the 6th Int'l Conf. on Computational Intelligence in Music, Sound, Art and Design (EvoMUSART)*, volume 10198 of *Lecture Notes in Computer Science*, pages 304–320, 2017.
- [26] I. Vatulkin, W. Theimer, and M. Botteck. AMUSE (Advanced MUSIC Explorer) - a multitool framework for music data analysis. In J. S. Downie and R. C. Veltkamp, editors, *Proc. of the 11th Int'l Society on Music Information Retrieval Conf. (ISMIR)*, pages 33–38, 2010.
- [27] A. Zlatintsi and P. Maragos. Multiscale fractal analysis of musical instrument signals with application to recognition. *IEEE Transactions on Audio, Speech & Language Processing*, 21(4):737–748, 2013.

INSTRUDIVE: A MUSIC VISUALIZATION SYSTEM BASED ON AUTOMATICALLY RECOGNIZED INSTRUMENTATION

Takumi Takahashi^{1,2}

Satoru Fukayama²

Masataka Goto²

¹ University of Tsukuba, Japan

² National Institute of Advanced Industrial Science and Technology (AIST), Japan

s1720822@s.tsukuba.ac.jp, {s.fukayama, m.goto}@aist.go.jp

ABSTRACT

A music visualization system called *Instrudive* is presented that enables users to interactively browse and listen to musical pieces by focusing on instrumentation. Instrumentation is a key factor in determining musical sound characteristics. For example, a musical piece performed with vocals, electric guitar, electric bass, and drums can generally be associated with *pop/rock* music but not with *classical* or *electronic*. Therefore, visualizing instrumentation can help listeners browse music more efficiently. Instrudive visualizes musical pieces by illustrating instrumentation with multi-colored pie charts and displays them on a map in accordance with the similarity in instrumentation. Users can utilize three functions. First, they can browse musical pieces on a map by referring to the visualized instrumentation. Second, they can interactively edit a playlist that showing the items to be played later. Finally, they can discern the temporal changes in instrumentation and skip to a preferable part of a piece with a multi-colored graph. The instruments are identified using a deep convolutional neural network that has four convolutional layers with different filter shapes. Evaluation of the proposed model against conventional and state-of-the-art methods showed that it has the best performance.

1 INTRODUCTION

Since multiple musical instruments having different timbres are generally used in musical pieces, *instrumentation* (combination or selection of musical instruments) is a key factor in determining musical sound characteristics. For example, a song consisting of vocals, electric guitar, electric bass, and drums may sound like *pop/rock* or *metal* but not *classical* or *electronic*. Consider, for example, a listener who appreciates *gypsy jazz* (featuring violin, acoustic guitar, clarinet, and double bass). How can he/she discover similar-sounding music? Searching by instrumentation can reveal musical pieces played with the same, slightly differ-

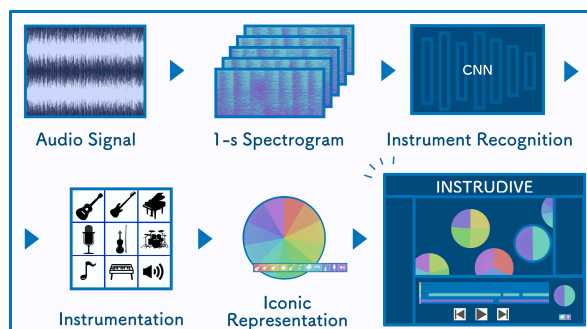


Figure 1: Overview of Instrudive music visualization system.

ent, or completely different instrumentation, corresponding to his/her preferences.

Instrumentation is strongly connected with musical sound and genres but is not restricted to a specific genre. For example, *pop/rock*, *funk*, and *fusion* are sometimes played with similar instrumentation. Therefore, it can be helpful for listeners to overcome the confinements of a genre by focusing on sound characteristics when searching for similar-sounding music.

To let users find musical pieces that they prefer, various methods and interfaces for retrieving and recommending music have been proposed. They are generally categorized into three approaches: bibliographic retrieval based on the metadata of musical pieces, such as artist, album, year of release, genres, and tags [2], music recommendation based on collaborative filtering using playlogs [5, 38], and music recommendation/retrieval based on content-based filtering using music analysis, such as genre classification [14, 30] and auto-tagging [4, 14, 20]. Music interfaces leveraging automatic instrument recognition [22] have received less attention from researchers.

We have developed a music visualization system called *Instrudive* that automatically recognizes the instruments used in each musical piece of a music collection, visualizes the instrumentations of the collection, and enables users to browse for music that they prefer by using the visualized instrumentation as a guide (Figure 1). Instrudive visualizes each musical piece as a pie-chart icon representing the duration ratio of each instrument that appears. This enables a user to see which instruments are used and their relative amount of usage before listening. The icons of



© Takumi Takahashi, Satoru Fukayama, Masataka Goto. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Takumi Takahashi, Satoru Fukayama, Masataka Goto. "Instrudive: A Music Visualization System Based on Automatically Recognized Instrumentation", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

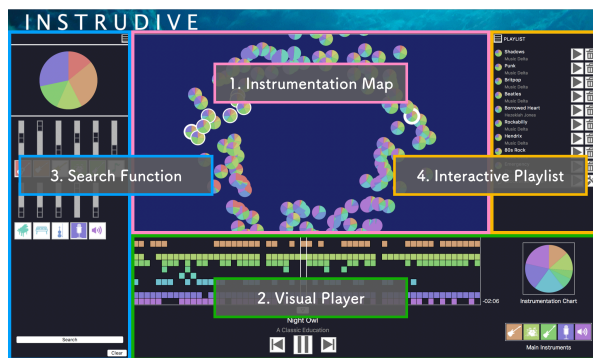


Figure 2: *Instrudive* interface consists of four parts.

all musical pieces in a collection are arranged in a two-dimensional space with similar-instrumentation pieces positioned in close proximity. This helps the user listen to pieces having similar instrumentation. Furthermore, the user can create a playlist by entering a pie-chart query to retrieve pieces having instrumentation similar to the query and listen to a musical piece while looking at a timeline interface representing when each instrument appears in the piece.

In the following section, we describe previous studies on music visualization and instrument recognition. We then introduce the usage and functions of *Instrudive* in Section 3 and explain its implementation in Section 4. Since the main contributions of this work are not only the *Instrudive* interface but also a method for automatically recognizing instruments on the basis of a deep convolutional neural network (CNN), we explain the recognition method and experimental results in Section 5. After discussing the usefulness of the system in Section 6, we summarize the key points and describe future work in Section 7.

2 RELATED WORK

2.1 Music Visualization

Visualization of music by using audio signal processing has been studied by many researchers.

Given a large collection of musical pieces, a commonly used approach is to visualize those pieces to make it easy to gain an overview of the collection [11, 13, 23, 24, 31, 32, 37, 40]. The collection is usually visualized so that similar pieces are closely arranged [13, 23, 24, 31, 32, 37]. The visualization helps listeners to find and listen to musical pieces they may prefer by browsing the collection. *Instrudive* visualizes the instrumentations of the pieces in the collection by displaying pie-chart icons for the pieces in a two-dimensional space as shown in Figure 2.

Given a musical piece, a commonly used approach is to visualize the content of the piece by analyzing the musical elements [3, 9, 10, 12, 18, 29]. For example, a repetitive music structure is often visualized [3, 9, 10, 12, 29]. This enhances the listening experience by making listeners aware of the visualized musical elements. Our *Instrudive* interface also takes this approach. After a user selects a musical

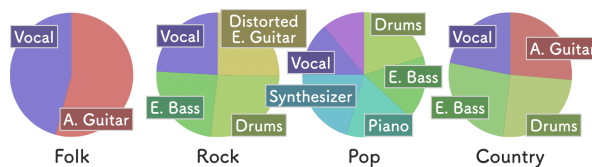


Figure 3: Multi-colored pie charts depict instrumentation.

piece, *Instrudive* displays a timeline interface representing when each musical instrument appears in the piece. This helps the listener focus on the instrumentation while listening to music.

2.2 Instrument Recognition

The difficulty in recognizing instruments depends on the number of instruments used in the piece. The greater the number of instruments, the greater the difficulty. When a single instrument is used in a monophonic recording, many methods achieve good performance [6, 8, 19, 41, 42].

On the other hand, when many instruments are used in a polyphonic recording, which is typical in popular music produced using multitrack recording, it is more difficult to recognize the instruments. Most previous studies [7, 15, 22, 26] used machine learning techniques to overcome this difficulty. In Section 5, we compare our proposed model of instrument recognition with one that uses a support vector machine (SVM).

A more recent approach to recognizing instruments is to use a deep learning method, especially a CNN [16, 27, 28, 34]. Methods using this approach have outperformed conventional and other state-of-the-art methods, but their performances cannot be easily compared due to the use of different databases and instrument labels. Despite their high performance, there is room for improvement in their accuracy. We aim to improve accuracy by proposing and implementing an improved CNN-based method.

3 INSTRUDIVE

Instrudive enables users to browse musical pieces by focusing on instrumentation. The key idea of visualizing the instrumentation is to use a multi-colored pie chart in which different colors denote the different instruments used in a musical piece. The ratios of the colors indicate relative durations in which the corresponding instruments appear. Figure 3 shows example charts created using ground truth annotations from the multitrack MedleyDB dataset [1]. The charts representing different genres have different appearances due to the differences in instrumentation among genres.

These multi-colored pie charts help a user browsing a collection of musical pieces to understand the instrumentations before listening to the pieces. Moreover, during the playing of a musical piece, *Instrudive* displays a multi-colored graph that indicates the temporal changes in instrumentation.

Instrudive can recognize 11 categories of instruments: acoustic guitar, clean electric guitar, distorted electric gui-



Figure 4: Menu appears after right-clicking chart.

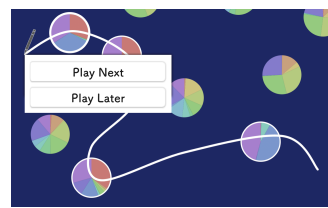


Figure 5: Scattering mode enables playlist to be created by drawing curve.

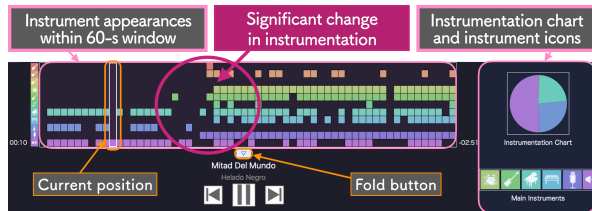


Figure 6: Visual player helps listener understanding instrumentation and its temporal changes.

tar, drums, electric bass, fx/processed sound (sound with effects), piano, synthesizer, violin, voice, and *other* (instruments not included in the 10 categories). The categories depend on this dataset and are defined on the basis of [27].

As shown in Figure 2, the interface of Instrudive consists of four parts: an instrumentation map for browsing musical pieces, a visual player for enhancing the listening experience, a search function for finding musical pieces by using the pie-chart icons as queries, and an interactive playlist for controlling the order of play.

3.1 Instrumentation Map

The instrumentation map visualizes the musical pieces in a collection. Each piece is represented by a multi-colored pie chart. Similar pie charts are closely located in a two-dimensional space. As shown in Figure 9, this map supports visualization modes, *circular* and *scattering*.

When a user right-clicks on a pie chart, a menu appears as shown in Figure 4. The user can play the piece or use the piece as a query for the search function. By using the circular mode, which arranges the pie charts in a circular path, the user can automatically play the pieces with similar instrumentation one after another along the path. By switching to the scattering mode, the user can draw a curve to create a playlist consisting of pieces on the curve as shown in Figure 5.

3.2 Visual Player

The visual player (Figure 6) visualizes the temporal changes in instrumentation in the selected musical piece as it is played. It shows a graph along the timeline interface consisting of a number of colored rectangular tiles, each of which denotes activity (i.e., presence) of the corresponding instrument. As the musical piece is played, this activity graph (covering a 60-s window) is automatically scrolled to continue showing the current play position.

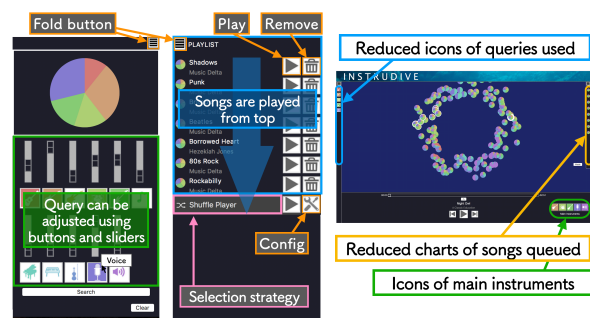


Figure 7: Interfaces for search menu and playlist.

Figure 8: Simplified interface for novice users.

The user can interactively change the play position by left-clicking on another position on the graph. The graph enables the user to anticipate how the instrumentation will change. For example, a significant change in instrumentation can be anticipated, as shown in Figure 6

The pie chart on the right side of Figure 6 represents the instruments currently being played and changes in synchronization with the playing of the piece. The *instrument icons* shown below the chart are consistently shown in the same color, enabling the user to easily distinguish them. By hovering the mouse over an icon, the user can see the name of the instrument.

3.3 Search Function

The search function (left side of Figure 7) enables the user to retrieve pieces by entering a query. Pressing an instrument-icon button intensifies its color, so the selected button is clearly evident. The ratio of instruments in the query can be adjusted by moving the sliders.

When the *search* button is pressed, the system retrieves musical pieces with instrumentation similar to that of the query by using the search algorithm described in Section 4.3. The retrieved pieces are not only highlighted on the map as shown in Figure 10 but also instantly added to the playlist.

3.4 Interactive Playlist

The interactive playlist (right side of Figure 7) shows a list of the retrieved or selected musical pieces along with their pie charts, titles, and artist names. The user can change their order, add or delete a piece, and play a piece.

A musical piece disappears from the playlist after it has been played. If no piece is in the list, the next piece is selected automatically. In circular mode, the available play strategies are *clockwise* (pieces are played in clockwise order), and *shuffle* (pieces are played randomly). In scattering mode, the available play strategies are *shuffle* and *nearest* (pieces nearby are played). The user can thus play pieces having similar or different instrumentation.

3.5 Simplified Interface

We also prepared a simplified interface for novice users who are not familiar with music instrumentation. As shown in Figure 8, the visual player, the search function,

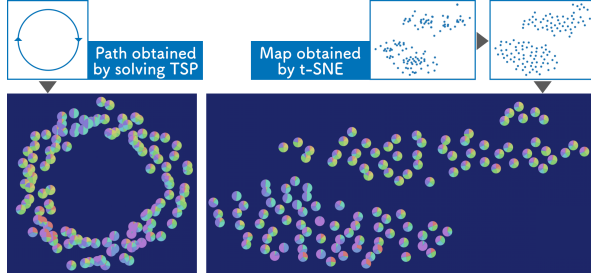


Figure 9: Two algorithms are used to create maps. Map on left is used in *circular* mode; map on right is used in *scattering* mode.

and the interactive playlist can be folded to the side to let the user concentrate on simple interaction using the instrumentation map.

4 IMPLEMENTATION OF INSTRUDIVE

The Instrudive interfaces were mainly programmed using a Python library *Tkinter* and executed on Mac OS X. After the instruments were recognized, as described in Section 5, the results were stored and used for the interfaces.

4.1 Iconic Representation

A multi-colored pie chart of a musical piece with length T s is displayed by computing the *absolute appearance ratio* (AAR) and the *relative appearance ratio* (RAR) for each instrument i ($\in \mathbf{I}$: recognized instrument categories).

The result of recognizing an instrument i is converted into AAR_i :

$$AAR_i = \frac{t_i}{T}, \quad (1)$$

where t_i ($\leq T$) s is the total of all durations in which instrument i is played. AAR represents the ratio of this total time against the length of the musical piece.

$$RAR_i = \frac{AAR_i}{\sum_i AAR_i} \quad (2)$$

represents the ratio of this total time against the total time of the appearances of all instruments. After RAR_i is computed for all instruments, an $|\mathbf{I}|$ -dimensional vector (11-dimensional vector in the current implementation) summarizing the instrumentation of the piece is obtained. The pie chart is a visual representation of this vector: RAR_i is used as an area ratio in the circle for the corresponding instrument.

4.2 Mapping Algorithms

To visualize musical pieces in *circular* mode (Figure 9), we use an $|\mathbf{I}|$ -dimensional vector (11-dimensional vector in the current implementation) of AAR. The AAR vectors for all the pieces are arranged on a circular path obtained by solving the traveling salesman problem (TSP) [25] to find the shortest route for visiting all pieces. After assigning all the pieces on the path, we scatter them randomly towards and away from the center of the circle so that the pie charts are not located too close together.

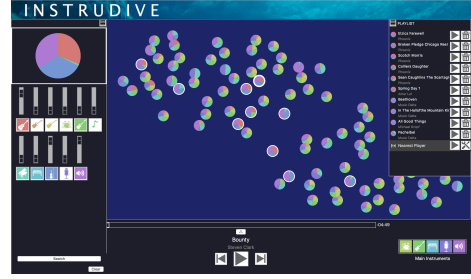


Figure 10: Top ten search results are highlighted and added to playlist. Users can check contents of results before listening.

Layer	Output size
Magnitude spectrogram	$1024 \times 87 \times 1$
Conv (4×1)	$1024 \times 87 \times 32$
Pool (5×3)	$204 \times 29 \times 32$
Conv (16×1)	$204 \times 29 \times 64$
Pool (4×3)	$51 \times 9 \times 64$
Conv (1×4)	$51 \times 9 \times 64$
Pool (3×3)	$17 \times 3 \times 64$
Conv (1×16)	$17 \times 3 \times 128$
Pool (2×2)	$8 \times 1 \times 128$
Dropout (0.5)	1024
Dense	1024
Dense	121
Dense	11

Table 1: Proposed CNN architecture.

To visualize musical pieces in *scattering* mode, the 11-dimensional AAR vectors are projected onto a two-dimensional space by using t-distributed stochastic neighbor embedding (t-SNE) [39], which is an algorithm for dimensionality reduction frequently used to visualize high-dimensional data. Since similar pie charts are often located too close together, we slightly adjust their positions one by one by randomly moving them until all the charts have a certain distance from each other.

4.3 Search Algorithms

Since both a query and a musical piece can be represented as 11-dimensional AAR vectors, we can simply compute the cosine similarity between the query and each musical piece in the collection. In Figure 10, for example, given a query containing acoustic guitar, violin, and others, the retrieved pieces ranked higher have similar pie charts. As the rank gets lower, the charts gradually becomes less similar.

5 INSTRUMENT RECOGNITION

5.1 Pre-processing

Each musical piece was converted into a monaural audio signal with a sampling rate of 44100 Hz and then divided into one-second fragments. To obtain a one-second magnitude spectrogram, we applied short-time Fourier transform (STFT) with a window length of 2048 and a hop size of 512. We then standardized each spectrogram to have zero mean and unit variance. As a result, each one-second spectrogram had 1024 frequency bins and 87 time frames.

5.2 CNN Architecture

We compared several CNN models; the one that showed the best performance is summarized in Table 1. The model mainly consists of four convolutional layers with max-pooling and ReLU activation. A spectrogram represents the structure of frequencies with one axis and its temporal changes against the other axis, which is unlike an image that represents spatial information with both axes. We set the shape of each layer to have length along only one axis (frequency or time). For convolutions, feature maps were padded with zeros so that dimensionality reduction was done only by using max-pooling layers. By doing this, we could use various shapes of layers and their combinations without modifying the shapes of other layers. After a 50% dropout was applied to prevent overfitting, two dense layers with ReLU and an output dense layer with a sigmoid function were used to output an 11-dimensional vector. Batch normalization [17] was applied to each of the convolutional and dense layers. In training, we used the Adam algorithm [21] as the optimizer and binary cross-entropy as the loss function. The mini-batch size was 128, and the number of epochs was 1000.

This proposed CNN model outputs 1-s instrument labels as a vector. By gathering the vectors corresponding to each musical piece, we can represent each musical piece as a sequence of 11-dimensional vectors (instrument labels/activations), which are used to calculate the instrumentation described in Section 4.

5.3 Dataset

To evaluate the proposed CNN model and apply it to *Instrudiver*, we used the MedleyDB dataset [1]. This dataset has 122 multitrack recordings of various genres and instrument activations representing the sound energy for each stem (a group of audio sources mixed together), individually calculated along with time frames with a hop size of 46.4 ms.

We generated instrument labels and split the data on the basis of the source code published online [27]. We used the 11 categories listed in Section 3 based on the ground truth annotations from the multitrack MedleyDB dataset [1]. Since our system does not depend on these categories, it can be generalized to any set of categories given any dataset.

The 122 musical pieces were divided into five groups by using the algorithm in [35] so that the instrument labels were evenly distributed among the five groups. Four of the groups were used for training, and the fifth was used for evaluation. All the musical pieces that appear in *Instrudiver* were included in the data used for evaluation, and their instrumentations were predicted using cross validation.

5.4 Baseline

For comparison with our model, we used a conventional bag-of-features method, a state-of-the-art deep learning method with mel-spectrogram input, and a state-of-the-art deep learning method with raw wave input.

Layer	Output size
Mel-spectrogram	$128 \times 43 \times 1$
Conv (3×3)	$130 \times 45 \times 32$
Conv (3×3)	$132 \times 47 \times 32$
Pool (2×2)	$44 \times 15 \times 32$
Dropout (0.25)	$44 \times 15 \times 32$
Conv (3×3)	$46 \times 17 \times 64$
Conv (3×3)	$48 \times 19 \times 64$
Pool (2×2)	$16 \times 6 \times 64$
Dropout (0.25)	$16 \times 6 \times 64$
Conv (3×3)	$18 \times 8 \times 128$
Conv (3×3)	$20 \times 10 \times 128$
Pool (2×2)	$6 \times 3 \times 128$
Dropout (0.25)	$6 \times 3 \times 128$
Conv (3×3)	$8 \times 5 \times 256$
Conv (3×3)	$10 \times 7 \times 256$
Global pool	$1 \times 1 \times 256$
Dense	1024
Dropout (0.5)	1024
Dense	11

Table 2: Han’s architecture.

5.4.1 Bag-of-features

For the bag-of-features method, we used the features described by [15], consisting of 120 features obtained by computing the mel-frequency cepstral coefficients and 16 spectral features [33]. We trained an SVM with a radial basis function (RBF) kernel by feeding it these 136 features.

5.4.2 Mel-spectrogram (Han’s CNN model)

For the deep learning method with mel-spectrogram input, we used Han’s CNN architecture [16] (Table 2). This architecture is based on VGGNet [36], a commonly used model in the image processing field. Each one-second fragment of the audio signal was resampled into 22050 Hz, converted into a mel-spectrogram, and standardized. Every activation function was LReLU ($\alpha = 0.33$) except the output sigmoid.

In preliminary experiments, training this model failed in almost 700 epochs due to a gradient loss. Therefore, we applied batch normalization to each of the convolutional and dense layers, enabling us to successfully complete 1000 epochs of training. We also used 500 epochs, but the performance was worse than for 1000.

5.4.3 Raw Waveform (Li’s CNN model)

For the deep learning method with raw wave input, we used Li’s CNN model in [27] (Table 3). This model performs end-to-end learning using a raw waveform. We standardized each one-second fragment of the monaural audio signal obtained in pre-processing. Every activation function was ReLU except the output sigmoid. Batch normalization was again applied to each layer. We trained the model with 1000 epochs.

5.5 Metrics

We evaluated each model using four metrics: *accuracy*, *F-micro*, *F-macro*, and *AUC*.

Accuracy was defined as the ratio of predicted labels that exactly matched the ground truth. Each label predicted by the CNN at every one-second fragment in all pieces was

Layer	Output size
Raw wave	44100×1
Conv (3101)	41000×256
Pool (40)	2049×256
Conv (300)	1750×384
Pool (30)	87×384
Conv (20)	68×384
Pool (8)	16×384
Dropout (0.5)	16×384
Dense	400
Dense	11

Table 3: Li’s architecture.

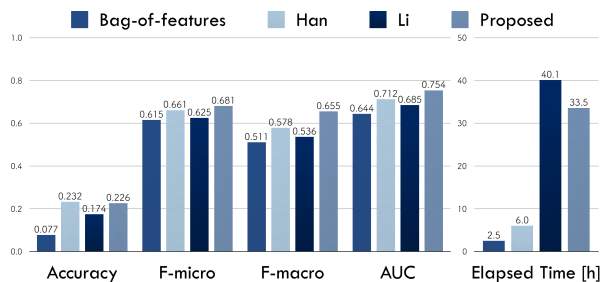


Figure 11: Proposed model showed best performance for F-micro, F-macro, and AUC but took five times longer to complete training than Han’s model, which showed second-best performance.

an 11-dimensional vector of likelihoods. Since each likelihood ranged between 0 and 1, we rounded it to an integer (0 or 1) before matching.

The F-micro was defined as the micro average of the F1 measure for all predicted labels over the 11 categories. The F1 measure is defined as the harmonic mean of recall and precision and is widely used in multi-label classification tasks. Since it is calculated immediately without considering the categories, if some instruments frequently appear, their predicted labels considerably affect the F-micro.

The F-macro was defined as the macro average with each instrument equally considered. For each of the 11 categories, the F1 measure of the predicted labels was first calculated. Then, the average of the resulting 11 values was calculated as the F-macro.

The area under the curve (AUC) of the receiver operating characteristic was first calculated for each category. Then, the macro average of the resulting 11 values was used as the AUC in our multi-label task.

5.6 Results

As shown in Figure 11, the proposed model outperformed the other models in terms of AUC, F-micro, and especially F-macro, which was about 8% better than the next-best model (Han’s model). This indicates that our model has higher generic performance and is more powerful in dealing with various kinds of instruments.

Interestingly, all of the deep learning methods showed significantly higher accuracy than the bag-of-features method. Since the accuracy cannot be increased with predictions made through guesswork, such as predicting classes that frequently appear, the deep learning methods are more capable of capturing the sound characteristics of instruments in sound mixtures.

The proposed model took five times longer to complete training than Han’s model. This is because Han’s model took advantage of using a more compact mel-spectrogram (128×87) than the raw spectrogram (1024×87) used for the proposed model. Since using a mel-spectrogram results in losing more information, the performance was worse.

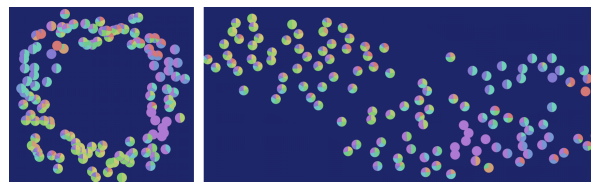


Figure 12: Maps created using ground truth data.

6 DISCUSSION

6.1 Smoothing Transitions Between Listening States

Our observations during testing showed that the use of *Instrudive* helped smooth the transition between listening states. Although the music was often passively listened to, the listeners sometimes suddenly became active when the time came to choose the next piece. In the *circular* mode of *Instrudive*, for example, the *clockwise player* played a piece that had instrumentation similar to the previous one. Since the sound characteristics were changing gradually, a user was able to listen to various genres in a passive state. If non-preferred music started playing, the user skipped to a different type of music by using the *shuffle player*. In addition, the user actively used the search function to access pieces with similar instrumentation and enjoyed looking at the temporal changes in the activity graph.

6.2 Studies from Ground Truth Data

We compared maps created using the automatically recognized (predicted) data (Figure 9) with maps created using the ground truth data (Figure 12). Although they are similar to some extent, the contrast of the color distributions is much more vivid for the ground truth data, suggesting that the performance of our CNN model still has room for improvement. Since the proposed *Instrudive* interface is independent of the method used for instrument recognition, we can simply incorporate an improved model in the future.

7 CONCLUSION

Our *Instrudive* system visualizes the instrumentations of the musical pieces in a collection for music discovery and active music listening. The first main contribution of this work is showing how instrumentation can be effectively used in browsing musical pieces and in enhancing the listening experience during playing of a musical piece. The second main contribution is proposing a CNN model for recognizing instruments appearing in polyphonic sound mixtures that achieves better performance than other state-of-the-art models.

We plan to conduct user studies of *Instrudive* to analyze its nature in more detail and to test different shapes of filters to analyze the reasons for the superior performance of our CNN model. We are also interested in investigating the scalability of our approach by increasing the number of musical pieces and allowing a greater variety of instruments.

8 ACKNOWLEDGMENTS

This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan.

9 REFERENCES

- [1] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 155–160, 2014.
- [2] Dmitry Bogdanov and Perfecto Herrera. How much metadata do we need in music recommendation? A subjective evaluation using preference sets. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, pages 97–102, 2011.
- [3] Matthew Cooper and Jonathan Foote. Automatic music summarization via similarity analysis. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, 2002.
- [4] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (IEEE ICASSP 2014)*, pages 6964–6968, 2014.
- [5] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173, 2010.
- [6] Antti Eronen and Auisi Klapuri. Musical instrument recognition using cepstral coefficients and temporal features. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (IEEE ICASSP 2000)*, volume 2, pages 753–756, 2000.
- [7] Slim Essid, Gaël Richard, and Bertrand David. Instrument recognition in polyphonic music based on automatic taxonomies. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):68–80, 2006.
- [8] Slim Essid, Gaël Richard, and Bertrand David. Musical instrument recognition by pairwise classification strategies. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1401–1412, 2006.
- [9] Jonathan Foote. Visualizing music and audio using self-similarity. In *Proceedings of the Seventh ACM International Conference on Multimedia (ACM Multimedia 1999)*, pages 77–80, 1999.
- [10] Masataka Goto. A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1783–1794, 2006.
- [11] Masataka Goto and Takayuki Goto. Musicream: Integrated music-listening interface for active, flexible, and unexpected encounters with musical pieces. *IPSS Journal*, 50(12):2923–2936, 2009.
- [12] Masataka Goto, Kazuyoshi Yoshii, Hiromasa Fujihara, Matthias Mauch, and Tomoyasu Nakano. Songle: A web service for active music listening improved by user contributions. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, pages 311–316, 2011.
- [13] Masahiro Hamasaki and Masataka Goto. Songrium: A music browsing assistance service based on visualization of massive open collaboration within music content creation community. In *Proceedings of the 9th International Symposium on Open Collaboration (ACM WikiSym + OpenSym 2013)*, pages 1–10, 2013.
- [14] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 339–344, 2010.
- [15] Philippe Hamel, Sean Wood, and Douglas Eck. Automatic identification of instrument classes in polyphonic and poly-instrument audio. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 399–404, 2009.
- [16] Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):208–221, 2017.
- [17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [18] Dasaem Jeong and Juhan Nam. Visualizing music in its entirety using acoustic features: Music flowgram. In *Proceedings of the International Conference on Technologies for Music Notation and Representation*, pages 25–32, 2016.
- [19] Ian Kaminskyj and Tadeusz Czaszejko. Automatic recognition of isolated monophonic musical instrument sounds using kNNC. *Journal of Intelligent Information Systems*, 24(2):199–221, 2005.
- [20] Taejun Kim, Jongpil Lee, and Juhan Nam. Sample-level cnn architectures for music auto-tagging using raw waveforms. In *Proceedings of the 14th Sound and Music Computing Conference (SMC 2017)*, 2017.
- [21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

- [22] Tetsuro Kitahara, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Instrogram: Probabilistic representation of instrument existence for polyphonic music. *IPSSJ Journal*, 2(1):279–291, 2007.
- [23] Peter Knees, Markus Schedl, Tim Pohle, and Gerhard Widmer. An innovative three-dimensional user interface for exploring music collections enriched with meta-information from the web. In *Proceedings of the 14th ACM International Conference on Multimedia (ACM Multimedia 2006)*, pages 17–24, 2006.
- [24] Paul Lamere and Douglas Eck. Using 3D visualizations to explore and discover music. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, pages 173–174, 2007.
- [25] Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, 1992.
- [26] Pierre Leveau, David Soderoy, and Laurent Daudet. Automatic instrument recognition in a polyphonic mixture using sparse representations. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, pages 233–236, 2007.
- [27] Peter Li, Jiyuan Qian, and Tian Wang. Automatic instrument recognition in polyphonic music using convolutional neural networks. *arXiv preprint arXiv:1511.05520*, 2015.
- [28] Vincent Lostanlen and Carmine-Emanuele Cella. Deep convolutional networks on the pitch spiral for music instrument recognition. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, pages 612–618, 2016.
- [29] Meinard Müller and Nanzhu Jiang. A scape plot representation for visualizing repetitive structures of music recordings. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, pages 97–102, 2012.
- [30] Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra. Multi-label music genre classification from audio, text and images using deep features. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, pages 23–30, 2017.
- [31] Elias Pampalk, Simon Dixon, and Gerhard Widmer. Exploring music collections by browsing different views. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, 2003.
- [32] Elias Pampalk and Masataka Goto. MusicRainbow: A new user interface to discover artists using audio-based similarity and web-based labeling. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, pages 367–370, 2006.
- [33] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the CUIDADO project. Technical report, IRCAM, 2004.
- [34] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra. Timbre analysis of music audio signals with convolutional neural networks. In *Proceedings of the 25th European Signal Processing Conference (EUSIPCO 2017)*, pages 2744–2748, 2017.
- [35] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahava. On the stratification of multi-label data. In *Machine Learning and Knowledge Discovery in Databases*, pages 145–158, 2011.
- [36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [37] Marc Torrens, Patrick Hertzog, and Josep-Lluís Arcos. Visualizing and exploring personal music libraries. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, 2004.
- [38] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS 2013)*, pages 2643–2651, 2013.
- [39] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.
- [40] Kazuyoshi Yoshii and Masataka Goto. Music Thumbnailer: Visualizing musical pieces in thumbnail images based on acoustic features. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, pages 211–216, 2008.
- [41] Guoshen Yu and Jean-Jacques Slotine. Audio classification from time-frequency texture. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (IEEE ICASSP 2014)*, pages 1677–1680, 2009.
- [42] Xin Zhang and Zbigniew W. Ras. Differentiated harmonic feature analysis on music information retrieval for instrument recognition. In *Proceedings of the IEEE International Conference on Granular Computing*, pages 578–581, 2006.

INSTRUMENT ACTIVITY DETECTION IN POLYPHONIC MUSIC USING DEEP NEURAL NETWORKS

Siddharth Gururani¹

Cameron Summers²

Alexander Lerch¹

¹ Center for Music Technology, Georgia Institute of Technology, USA

² Gracenote, Emeryville, USA

{siddgururani, alexander.lerch}@gatech.edu, cameron.summers@nielsen.com

ABSTRACT

Although instrument recognition has been thoroughly research, recognition in polyphonic music still faces challenges. While most research in polyphonic instrument recognition focuses on predicting the predominant instruments in a given audio recording, instrument activity detection represents a generalized problem of detecting the presence or activity of instruments in a track on a fine-grained temporal scale. We present an approach for instrument activity detection in polyphonic music with temporal resolution ranging from one second to the track level. This system allows, for instance, to retrieve specific areas of interest such as guitar solos. Three classes of deep neural networks are trained to detect up to 18 instruments. The architectures investigated in this paper are: multi-layer perceptrons, convolutional neural networks, and convolutional-recurrent neural networks. An in-depth evaluation on publicly available multi-track datasets using methods such as AUC-ROC and Label Ranking Average Precision highlights different aspects of the model performance and indicates the importance of using multiple evaluation metrics. Furthermore, we propose a new visualization to discuss instrument confusion in a multi-label scenario.

1. INTRODUCTION

Music is an acoustic rendition of musical ideas. In most cases, one or more instruments are used for this acoustic rendition. As humans, we are easily able to identify the instruments being played in a song after exposure to their sound. However, the same cannot be said for computer algorithms. The task of recognizing musical instruments in an audio signal has been an active area of research in the field of Music Information Retrieval (MIR). While instrument recognition in monophonic audio (only one instrument is present in a signal) is reasonably successful [13], the task is much harder in a polyphonic setting. The challenges

include, among others, the large variance in timbre and performance style within an instrument class combined with perceptual similarity of some instruments and the superposition of multiple instruments in time and frequency. Last but not least, the lack of data with relevant annotations for data-driven approaches is also a problem.

The identification of instruments and their activity in a song is important for music browsing and discovery, such as searching for songs with specific instruments or identifying the position of lead vocals or a saxophone solo. Instrument recognition can also inform other MIR tasks. For example, music recommendation systems can benefit from modeling a user's affinity towards certain instruments and music genre recognition systems could improve with genre-dependent instrument information. It can also be useful for tasks such as automatic music transcription, playing technique detection, and source separation in polyphonic music, where pre-conditioning a model on specific instruments present could possibly boost its performance.

An Instrument Activity Detection (IAD) system takes an audio track as input and outputs continuous instrument activity levels along the entire track. These activities may be binary (on/off) or on a continuous scale as likelihood. IAD systems may have varying time-resolutions for the instrument activity depending on the use case. For example, a solo detection use case would have a finer time-resolution than an instrument tagging system which would work on the track level. This paper proposes a deep neural network-based IAD system trained using multi-track datasets. We also address the problem of evaluation of an IAD system.

The following section reviews literature in instrument recognition and other related tasks. Section 3 describes the proposed IAD system starting with pre-processing the data, the model architectures and post-processing steps. Next, Section 4 describes the dataset used, the various experiments, the evaluation metrics and the proposed method to visualize confusion. We report the results for the experiments in terms of the evaluation metrics and discuss these results in Section 5. Finally, in Section 6 we conclude the paper enumerating a few possible future directions for research on IAD.



© Siddharth Gururani, Cameron Summers, Alexander Lerch. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Siddharth Gururani, Cameron Summers, Alexander Lerch. "Instrument Activity Detection in Polyphonic Music using Deep Neural Networks", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

2. RELATED WORK

The task of ‘instrument recognition’ can be divided into two distinct research problems based on the type of data being analyzed: (i) instrument recognition in monophonic audio and (ii) instrument recognition in polyphonic audio. This section presents an overview of past literature on instrument recognition as well as related topics such as automatic music tagging and sound event detection (SED).

2.1 Instrument Recognition in Monophonic Music

In monophonic music, instrument recognition may be performed on sounds at the note-level or on continuous audio signals of solo instrument performances. An extensive review of traditional feature extraction and classification approaches for note-level instrument recognition has been published by Herrera et al. [15]. For solo phrases, Es-sid et al. utilize MFCCs as features, Principal Component Analysis (PCA) for dimensionality reduction, and Gaussian mixture models (GMM) for classifying solo phrases of 5 instruments [8]. Krishna and Sreenivas propose the so-called Line Spectral Features (LSF). LSFs are used with a GMM and evaluated for instrument family classification and 14-class instrument classification [19].

In addition to extracting established pre-defined features, learned features have also been applied to this task. Yu et al. utilize sparse spectral codes and a support vector machine (SVM) for classifying single-source and multi-source (polyphonic) audio [31]. Han et al. propose to use sparse coding for learning features from mel-spectrograms extracted from a dataset of single-note audio clips for 24 instruments. A SVM is trained to classify the instruments using the learned features achieving a classification accuracy of around 95% for 24 instrument classes [13].

2.2 Instrument Recognition in Polyphonic Music

Recent work on instrument recognition has focused on polyphonic musical signals. Polyphonic audio synthesized from datasets of individual instrument sounds, such as the RWC dataset [10], as well as real-world audio recordings have been used for this task.

Kitahara et al. extract spectral and temporal features along with PCA and Latent Discriminant Analysis (LDA) for classification in duo and trio music [17]. Heittola et al. combine the results of Non-negative Matrix Factorization (NMF) with excitations of notes obtained from a multi-pitch tracking algorithm [18] to extract harmonic spectra from a mixture signal. The separated spectra are represented by MFCCs and classified with a GMM [14].

Fuhrmann et al. extract a large set of features representing an audio clip and perform predominant instrument detection in real-world audio signals using one SVM per instrument [9]. The ‘predominant’ instrument is defined as one with continuous presence in a snippet of audio and is easily audible for a human listener. Bosch et al. extend the work by utilizing source separation to segregate the polyphonic audio into streams: ‘bass,’ ‘drums,’ ‘melody,’

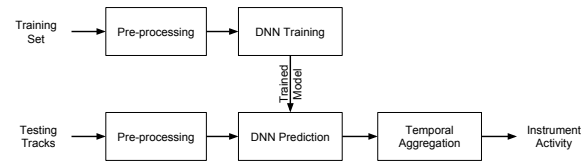


Figure 1. Block Diagram for DNN-based IAD System

and ‘other.’ The segregated audio is subsequently used for classification using the aforementioned system [2].

Han et al. apply deep CNNs for the task of predominant instrument recognition and report a significant improvement of results over previous approaches [12]. The authors also provide an in-depth discussion of the model parameters and a qualitative analysis of the CNN models.

2.3 Related Tasks

In music tagging, a track is labeled with a variety of labels that describe it, such as genre, instruments, and mood. IAD may be considered a sub-task in music tagging since the tags often include instrumentation. Choi et al. use CNNs and CRNNs for the task of automatic tagging [5, 6]. Liu and Yang further proposed a method to localize the events in music tagging [20] which may be compared to IAD.

Sound Event Detection (SED) aims at detecting environmental sounds in a stream of audio. Some examples of sound events are gunshots, car horns, baby cries, dog barks, etc. Cakir et al. explore this task with deep neural networks on a dataset of environmental sounds [3, 4]. The main difference between SED and IAD is that in SED the sound events are uncorrelated and thus easier to discriminate while musical sources tend to have higher correlation in popular music. Music instrument sounds might also have a rich harmonic structure absent in most environmental sounds.

3. METHOD

A high-level block diagram for the presented IAD system is shown in Fig. 1. The individual processing steps are described in detail below.

3.1 Pre-processing

All tracks are downsampled to 22.05 kHz, downmixed to mono and normalized by the root mean square energy. Each track is chunked to 1 s long snippets. Each snippet is transformed into a mel-spectrogram, which is motivated by the non-linear frequency resolution of the human auditory system [22], and has been proven to be a useful input representation for multiple MIR tasks such as automatic tagging [5], onset detection [25], and feature learning [29].

The mel-spectrograms are calculated using Librosa [21] with 96 mel bands from 0–11.025 kHz. The block size and hop size are 46.4 ms and 11.6 ms, respectively. Decibel scaling is applied to the Mel-Spectrogram energies. The result is a matrix of dimension 96×86 .

CNN	CRNN
Conv2D $k = 3 \times 3, d = 64$ MP ($p = 2, 2$)	Conv2D $k = 3 \times 3, d = 128$ MP ($p = 2, 2$)
Conv2D $k = 3 \times 3, d = 256$ MP ($p = 3, 3$)	Conv2D $k = 3 \times 3, d = 256$ MP ($p = 2, 2$)
Conv2D $(k = 3 \times 3, d = 640)$ MP ($p = 3, 3$)	Conv2D $(k = 3 \times 3, d = 256)$ MP ($p = 2, 2$)
FC ($h = 128$)	GRU ($h = 256$)
FC ($h = 18$)	

Table 1. Model Architecture. (Conv2D: 2D Convolutional Layer, MP: 2D Max-Pooling, k : kernel size, d : filter depth)

3.2 Model Architectures

Deep Neural Networks (DNNs) have consistently outperformed traditional MIR approaches in several tasks such as, music transcription [26, 30], onset detection [25], music tagging [5]. As this is also true for predominant instrument classification (compare Sect. 2), we choose to investigate DNNs for the task of IAD. Our architectural choices are influenced by the work of both Choi and Cakir [4–6].

The usability of DNNs stems from their ability to approximate complex non-linear functions mapping an input feature space to the outputs. This enables researchers to provide raw or minimally processed data to a DNN so that it may learn features relevant for the task at hand.

We compare the three broad classes of DNNs: multi-layer perceptrons, convolutional neural networks, and —since convolutional networks are useful for acoustic modeling [5]— a convolutional-recurrent network instead of a traditional RNN. The benefit of CRNN lies in the fact that it is able to learn both local and temporal features. Note that the model hyperparameters have been chosen so that the number of parameters for the three models is comparable.

3.2.1 Multi-Layer Perceptron

The input mel-spectrogram matrix is flattened into a vector for the MLP model. A fairly simple architecture is chosen: 4 hidden layers with 256 hidden units in each layer and an output layer of 18 hidden units. Dropout [28] is used with a keep probability of 0.5 at each layer.

3.2.2 Convolutional Neural Network

The CNN architecture is shown in Table 1 (left). Small square filters are chosen in order to facilitate hierarchical feature learning from local patches that grow larger in size with network depth. In order to preserve spatial dimensions, stride of 1 and *Same* zero-padding scheme is used for all the convolutional layers. Each Conv2D layer is followed by batch-normalization [16] and the Exponential Linear Unit (ELU) [7] activation function. The final convolution layer’s output is flattened before feeding it to a fully connected layer. Finally, we connect to an output layer of 18 units with a sigmoid activation function.

Instrument	Abbr.	Train		Test	
		T	#	T	#
drum set	dru	300	720036	79	15957
electric bass	bgtr	253	620592	62	13344
male singer	ms	200	351384	62	10038
dist. elec. gtr	dgtr	171	396204	40	7522
clean elec. gtr	cgtr	119	225456	34	5875
synthesizer	syn	118	295524	33	5712
acoustic gtr	agtr	91	230556	25	5241
piano	pf	89	187536	24	4063
vocalists	vox	84	154596	12	1895
female singer	fs	79	149232	23	3733
string section	str	24	39444	10	1278
elec. piano	epf	24	52680	14	2075
elect. organ	eorg	22	39516	11	2117
double bass	db	21	40116	9	1786
cello	vc	13	22176	9	1623
violin	vn	10	28452	15	2385
tabla	tab	9	41640	3	806
flute	fl	7	9972	7	1171

Table 2. Dataset distribution: T denotes tracks and # denotes 1 s snippets

3.2.3 Convolutional Recurrent Neural Network

The CRNN architecture is shown in Table 1 (right). CRNNs have been applied to tasks such as music tagging [6] and sound event detection [4]. We hypothesize that it is a good choice for IAD since we want the model to learn from the evolution of spectra over time. The same configuration of padding and striding, batch-normalization, and non-linear activation is used for the convolutional modules. Only the depth and height of the final Conv layer output is flattened, thus preserving the temporal structure of the high-level ConvNet features. Finally, the last GRU output is connected to the output layer consisting of 18 units with a sigmoid activation function.

3.2.4 Training Procedure

Binary cross-entropy is used as the loss function for all models. Stochastic gradient descent with a learning rate of 0.0001 and momentum of 0.9 is used to optimize the loss function. The models are trained using batches of 32 instances for 20 epochs, which is sufficient for the training and validation loss to converge for each of the architectures.

3.3 Temporal Aggregation

Since the neural networks are trained using 1 s snippets of audio, a prediction is made for every 1 s in the test track. For experiments and evaluation with varying time-resolution, we max-pool the predictions and the ground truth over non-overlapping segments according to the desired time-resolution. For example, in order to have a 5 s resolution, the maximum across 5 continuous predictions for every instrument is chosen as the predicted score for the corresponding 5 s snippet in the track.

4. EVALUATION

4.1 Dataset

The dataset used in previous work on predominant instrument detection [2, 9, 12], IRMAS, consists of a training set

with 3 s audio snippets manually annotated with one of 11 predominant (but non-percussive) instrument labels. These snippets may contain other instruments. The testing set contains audio snippets of variable length with 1 or more predominant instruments. We believe that training using polyphonic audio labeled with a single instrument may not be the ideal strategy for IAD. In this paper, we used multi-track audio to construct a dataset for IAD. The motivation behind using multi-track datasets is that the annotations for instrument activity can be generated automatically using stem energy as opposed to human annotations which may contain more errors. In addition, each snippet may contain multiple instrument labels, providing the models richer ground truth.

Two publicly available multi-track datasets are used for training and testing of the models. MedleyDB [1] and Mixing Secrets [11] were combined for this task in order to increase the number of tracks. In a pilot study involving only MedleyDB, we observed a significant improvement in model performance as the amount of training data increased. MedleyDB contains 330 multi-tracks and Mixing Secrets contains 258 multi-tracks. The two datasets combined contain tracks with approximately 100 different instruments. For this paper we consider 18 most frequently occurring instruments. The instruments considered are listed in Table 2. Note that the tracks may contain other instruments that the IAD system is not trained to detect.

Each multi-track in the dataset is associated with a mixed track. Instrument activation confidence is annotated automatically according to the process described in [1]. These annotations are computed with time-resolution of 0.0464 s. For our IAD system, however, we defined the minimum time-resolution to be 1 s. The annotations are aggregated by picking the maximum value across the time-axis to obtain one activation value per instrument per snippet. This allows for instruments to have a large activation value in the snippet even if they were active for a small period of time, as opposed to a value close to 0 if the mean was chosen for aggregation. Finally, the activations are binarized with a fixed threshold $\theta = 0.5$.

The datasets contain tracks where the stems have cross-talk or bleed. For these tracks, stem activations for a certain instrument may contain activity from another instrument. To prevent incorrect annotations, tracks with bleed are not considered for the IAD dataset, although we make exceptions for rare instruments such as tabla. Additionally, tracks without a single instrument of interest are not considered.

Subsequently, the dataset is split into a training and a testing set. We generate a random artist-conditional split to prevent the album or artist effect in the testing phase. The split is chosen such that there is a reasonable number of tracks per instrument. Table 2 lists the distribution of the data for the split. The training set consists of 361 tracks and the testing set consists of 100 tracks.¹ The training set is augmented using pitch-shifting: 6 semitones lower to 5 higher than the original with 1 semi-tone increments.

¹The track IDs for the dataset splits used are available at <https://github.com/SiddGururani/ISMIR2018>

4.2 Experimental Setup

First, we preprocess both splits of data as described in Sect. 3.1 resulting in a time-frequency input representation and ground truth pair for each 1 second snippet. Table 2 lists the distribution of the different instrument classes in terms of 1 second snippets. Next, we train each of the DNN architecture as described in Sect. 3.2. Since the models were observed to converge to a solution in 20 epochs, we do not perform any form of early-stopping. In addition, we generate a validation set using a randomly sampled set of tracks from both the training and testing set due to lack of data. 50 tracks from the training and testing splits are picked, resulting in a validation set of 100 tracks. We use this scheme since we want to validate on unseen data while not using the entire test set. The validation set is used to evaluate the models at the end of each epoch. Finally, we test the best performing model for each class of DNNs. We test the models for various time-resolutions of activity detection: 1 s, 5 s, 10 s and track-level aggregation.

4.3 Evaluation Metrics

Evaluation of IAD systems, when looked at in detail, poses some challenges. Since each snippet has zero or more instruments, IAD is a multi-label classification problem. The sigmoid activation leads to an output between 0 and 1, denoting the predicted activity of that instrument. However, as pointed out by Han et al. [12], binarizing the outputs using a fixed threshold and evaluating the accuracy depends on the selected threshold. Additionally, the dataset is not balanced across the instrument classes, hence stressing the need for metrics robust against unbalanced class distribution.

Previous work on predominant instrument recognition uses metrics relevant for multi-class classification systems such as precision, recall and f-measure [2, 12]. Since IAD is a multi-label classification problem, we use Label Ranking Average Precision (LRAP) and the Area Under Receiver Operating Characteristic curve (AUC-ROC).

4.3.1 Label Ranking Average Precision

LRAP was proposed in [24] to evaluate multi-label classification systems. Intuitively, the LRAP measures the ability of a model to assign better ranks to true labels for an instance. For example, if all the true labels for an instance are ranked higher than other labels in consideration, the ranking precision for this instance is 1. LRAP measures the average ranking precision across all the instances. In our experiments, we compute LRAP using 2 approaches: (i) Micro: LRAP computed using the concatenated outputs for all testing tracks. (ii) Macro: computed on the track level and averaged. This normalizes any effect of track length on the model performance, which could skew the results, for instance, if the model performs well for a particular long song but poorly for shorter songs with fewer snippets.

4.3.2 Area Under ROC Curve

The AUC-ROC or, in short, AUC is computed by first plotting the true positive rate and false positive rate on a plane for various classification thresholds, which results in a curve.

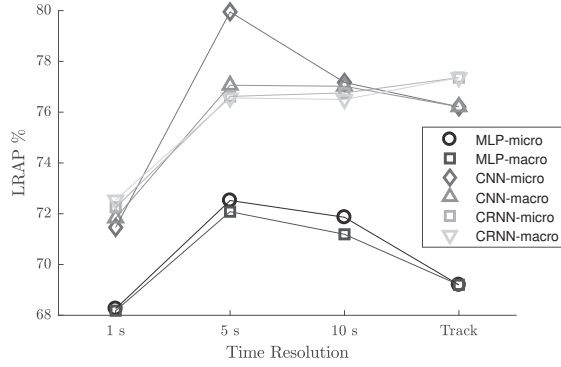


Figure 2. LRAP for various time-resolutions

AUC is the area under this curve. It measures the probability that the model assigns a higher score to a randomly selected positive instance than a negative instance. The AUC gives a summary of the model performance without the need to adjust a threshold for binarization.

Since AUC is usually applied to binary classification, we compute it per instrument class. Only the micro AUC is computed as not all tracks contain all instruments. We report an average AUC by taking the mean of the micro AUC per class.

The reason for selecting AUC instead of precision, recall or f-measure is that most literature on instrument classification tends to use a common fixed threshold for all classes which bears the risk of being suboptimal. Han et al. suggest the use of a different threshold per instrument class [12]. Using the AUC to summarize model performance alleviates the problem of threshold selection while making it easier to directly compare model performance.

4.4 Confusion Visualization

In multi-class classification, every data sample has only one possible prediction and one ground truth label. A confusion matrix visualizes the frequency of confusion between every pair of predicted class label vs. ground truth class label. In a multi-label classification problem such as IAD, every instance has multiple possible predictions and zero or more ground truth labels. Hence, a traditional confusion matrix cannot be computed. However, as a confusion matrix is an intuitive way to gain insights into the model, we propose an alternative form of confusion visualization computed from the binarized predictions and the ground truths.

We hypothesize that an instrument is wrongly detected due to the activity of some instrument present in the audio. We are particularly interested in looking at which instruments were incorrectly missed (false negative) when an instrument was wrongly detected (false positive). For a particular false positive instrument, this is equivalent to looking at the probability of observing false negatives for the other instruments. This probability can be estimated using a histogram of false negatives. Vertically stacking these histograms for each instrument results in a matrix of dimension $C \times C$ (C = number of instrument classes). We

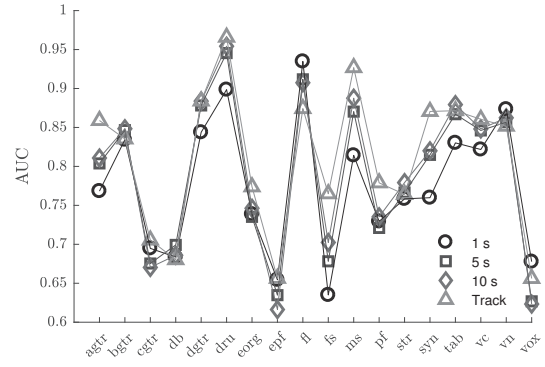


Figure 3. AUC per instrument for CRNN model

convert the histograms to probabilities by normalizing each row of the matrix to a sum of 1.

Note that unlike a traditional confusion matrix, this is not a symmetric matrix. We only focus on one row at a time in order to compare probabilities of observing false negatives for a given false positive instrument.

5. RESULTS AND DISCUSSION

A comparison of model performance is summarized in Figure 2 and Table 3. It can be observed that CNN and CRNN outperform MLP in both metrics. This is expected since the convolutional layers allow the model to learn hierarchical acoustic features from the time-frequency representation more efficiently. However, the CRNN does not outperform the CNN, which may be attributed to the fact that only 1 s second snippets are used. The temporal dimension of the input is reduced to only 5 time steps after the 4 CNN layers. The benefits of using recurrent layers are more noticeable when longer sequences are involved as in work by Choi et al. where they use inputs of length 29 s [6]. In addition, the receptive field of the deeper layers of the CNN is large enough for learning temporal features. Another observation is that output aggregation tends to improve models' label ranking performance and mean AUC.

Figure 3 shows the AUC per instrument of the CRNN model for the chosen time-resolution aggregation. We observe that using output aggregation in time leads to better performance in almost all instrument classes. The model achieves high AUC not only for majority instruments in the dataset but also for minority instruments such as flute, violin and cello, suggesting that it not simply predicting the majority. We also observe that the model does not seem to perform well for vocals in general. While it does achieve high AUC for male singers, the AUC for female singers and vocalists is low. We investigate this further using the

	MLP	CNN	CRNN
1 s	71.28	77.55	77.5
5 s	70.85	78.35	78.76
10 s	70.82	78.59	79.22
Track	71.1	80.92	80.1

Table 3. Mean AUC for various time-resolutions

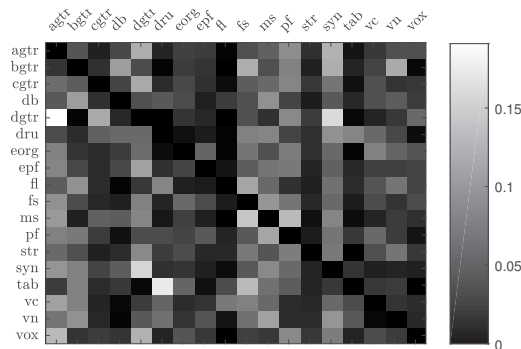


Figure 4. Distribution of false negative instruments conditioned on a true positive of a particular instrument

visualization method described in Sect. 4.4.

To construct the confusion visualization as described in Sect. 4.4, we pick the CRNN model and use the 1 s time-resolution outputs for the test set. Picking the threshold for binarizing the predictions is not straightforward. A fixed threshold of 0.5, for example, led to 0 detections for string section and electronic organ. Therefore, we adjusted the best thresholds for each class. These thresholds are determined by computing the class-wise f-measure at all score thresholds and selecting the threshold giving the best f-measure in the validation set. Figure 4 shows the constructed visualization. A high value in a row implies that the model may be confusing that particular pair of instruments more often than others. The following observations can be made from the figure:

- (*bgtr*, *db*): This confusion is possibly due to similar frequency range of the electric bass and double bass.
- (*dgtr*, *agtr*), (*dgtr*, *cgtr*), (*cgtr*, *dgtr*), and (*agtr*, *dgtr*): While confusion between acoustic and distorted guitar is unusual, the confusion between clean and distorted guitar is possibly explained by the variety in tone for both the clean and distorted guitars. A light crunch or low gain setting may possibly get misclassified. This could also explain the poor performance for clean electric guitar.
- (*dru*, *tab*) and (*tab*, *dru*): Both drum set and tabla are percussive instruments. In addition, one of the test tracks containing tabla has a ‘drum machine’ label which possibly causes drum false positives.
- (*dgtr*, *syn*) and (*syn*, *dgtr*): This is an interesting case since the variance in sound for both, the distorted guitar and synthesizers, is very large. Further investigation is needed to understand this case.

Next, we investigate the poor model performance on vocal classes. Figure 4 shows confusion between male and female singers implying that the model might be incorrectly classifying female singers as male. In order to investigate this phenomenon, the three vocal classes were combined for a follow-up experiment. We max-pool the predictions and

	1 s	5 s	10 s	Track
Average AUC (ms, fs, vox)	0.709	0.725	0.737	0.782
AUC vocals	0.822	0.96	0.975	0.998

Table 4. AUC for different time resolutions comparing pooled vocals against averaged AUC for vocal classes

the ground truth for these three classes, and recompute the AUC for this new ‘vocals’ class. Table 4 shows the average AUC of the three classes and the AUC of the combined ‘vocals’ class. The model performs significantly better for the combined class confirming our hypothesis that it confuses the vocal classes.

Another interesting finding is that the best threshold chosen per instrument for binarization ranges from 0.02 to 0.55 with lower thresholds for minority instruments in general. We observe a correlation coefficient of 0.9 between the thresholds and the training data distribution suggesting that the model has learned biases in the dataset. The impact of this finding requires further experiments.

6. CONCLUSION

We presented a DNN-based IAD system trained using multi-track datasets to detect 18 instruments. The CRNN and CNN outperform MLP architectures for the task and perform well for detecting instruments common in popular music, such as drums, electric bass, acoustic guitars, distorted guitars and vocals. It also performs well for instruments in classical music such as flute, cello, violin even though they were under-represented in the dataset. We also stress the need for multiple metrics and visualizations for evaluation of systems such as IAD which is non-trivial to evaluate.

As future work, a few extensions and research directions are: (i) pre-training the network using monophonic stems from the multi-track datasets and subsequently training and testing for IAD, (ii) designing the convolutional network for the CRNN as proposed by Jordi et al. [23] instead of the currently used 3×3 filters as is common in computer vision, (iii) converting the proposed monolithic model architecture for IAD to a hierarchical architecture for instrument family classification first and subsequently instrument classification. While this paper treats the model as a black box and focuses on evaluation and analysis of model outputs, it is worth studying the model to understand the internal representations by means of visualization tools such as t-SNE and saliency maps [27] as performed by Han et al. [12].

By drawing attention to challenges in IAD with this paper, we hope to encourage the MIR community to explore this task. IAD is a rewarding avenue for research due to its real-world use cases as well as the potential to augment and improve performance in other tasks in MIR.

7. ACKNOWLEDGMENTS

This work was funded by Gracenote. We thank them for their generous support. We would also like to thank Nvidia for supporting us with a Titan Xp awarded as part of the GPU grant program.

8. REFERENCES

- [1] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. Medleydb: A multitrack dataset for annotation-intensive mir research. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 155–160, 2014.
- [2] Juan J Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera. A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 559–564, 2012.
- [3] Emre Cakir, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Polyphonic sound event detection using multi label deep neural networks. In *Proc. International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2015.
- [4] Emre Cakir, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, Tuomas Virtanen, Emre Cakir, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 25(6):1291–1303, 2017.
- [5] Keunwoo Choi, György Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. In *Proc. of the International Society of Music Information Retrieval Conference (ISMIR)*, pages 805–811, 2016.
- [6] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *Proc. of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 2392–2396, 2017.
- [7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *International Conference on Learning Representations (ICLR)*, 2015.
- [8] Slim Essid, Gaël Richard, and Bertrand David. Musical instrument recognition on solo performances. In *Proc. of the 12th European Signal Processing Conference*, pages 1289–1292, 2004.
- [9] Ferdinand Fuhrmann, Martín Haro, and Perfecto Herrera. Scalability, generality and temporal aspects in automatic recognition of predominant musical instruments in polyphonic music. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 321–326, 2009.
- [10] Masataka Goto. Rwc music database: Music genre database and musical instrument sound database. In *Proc. International Conference on Music Information Retrieval*, 2003, pages 229–230, 2003.
- [11] Siddharth Gururani and Alexander Lerch. Mixing secrets: A multitrack dataset for instrument detection in polyphonic music. In *Late Breaking Demo (Extended Abstract), Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, 2017.
- [12] Yoonchang Han, Jaehun Kim, Kyogu Lee, Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 25(1):208–221, 2017.
- [13] Yoonchang Han, Subin Lee, Juhan Nam, and Kyogu Lee. Sparse feature learning for instrument identification: Effects of sampling and pooling methods. *The Journal of the Acoustical Society of America*, 139(5):2290–2298, 2016.
- [14] Toni Heittola, Anssi Klapuri, and Tuomas Virtanen. Musical instrument recognition in polyphonic audio using source-filter model for sound separation. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 327–332, 2009.
- [15] Perfecto Herrera-Boyer, Geoffroy Peeters, and Shlomo Dubnov. Automatic classification of musical instrument sounds. *Journal of New Music Research*, 32(1):3–21, 2003.
- [16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of the 32nd International Conference on Machine Learning (ICML)*, volume 37, pages 448–456. PMLR, 2015.
- [17] Tetsuro Kitahara, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Instrument identification in polyphonic music: Feature weighting to minimize influence of sound overlaps. *EURASIP Journal on Applied Signal Processing*, 2007(1):155–155, 2007.
- [18] Anssi Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 216–221, 2006.
- [19] AG Krishna and Thippur V Sreenivas. Music instrument recognition: from isolated notes to solo phrases. In *Proc. of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*, volume 4, pages iv–iv, 2004.
- [20] Jen-Yu Liu and Yi-Hsuan Yang. Event localization in music auto-tagging. In *Proc. of the 2016 ACM on Multimedia Conference*, pages 1048–1057. ACM, 2016.
- [21] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *14th python in science conference*, pages 18–25, 2015.

- [22] Brian CJ Moore. *An Introduction to the Psychology of Hearing*. Brill, 2012.
- [23] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra. Timbre analysis of music audio signals with convolutional neural networks. In *Proc. of the 25th European Signal Processing Conference*, Kos island, Greece, 28/08/2017 2017. IEEE.
- [24] Robert E Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168, 2000.
- [25] Jan Schluter and Sebastian Bock. Improved musical onset detection with convolutional neural networks. In *Proc. of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 6979–6983, 2014.
- [26] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(5):927–939, 2016.
- [27] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *International Conference on Learning Representations (ICLR)*, 2013.
- [28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [29] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2643–2651, 2013.
- [30] Richard Vogl, Matthias Dorfer, and Peter Knees. Drum transcription from polyphonic music with recurrent neural networks. In *Proc. of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 201–205, 2017.
- [31] Li-Fan Yu, Li Su, and Yi-Hsuan Yang. Sparse cepstral codes and power scale for instrument identification. In *Proc. of the International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 7460–7464, 2014.

JAZZ SOLO INSTRUMENT CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS, SOURCE SEPARATION, AND TRANSFER LEARNING

Juan S. Gómez Jakob Abeßer Estefanía Cano
Semantic Music Technologies Group, Fraunhofer IDMT, Ilmenau, Germany
{gomezjn, abr, cano}@idmt.fhg.de

ABSTRACT

Predominant instrument recognition in ensemble recordings remains a challenging task, particularly if closely-related instruments such as alto and tenor saxophone need to be distinguished. In this paper, we build upon a recently-proposed instrument recognition algorithm based on a hybrid deep neural network: a combination of convolutional and fully connected layers for learning characteristic spectral-temporal patterns. We systematically evaluate harmonic/percussive and solo/accompaniment source separation algorithms as pre-processing steps to reduce the overlap among multiple instruments prior to the instrument recognition step. For the particular use-case of solo instrument recognition in jazz ensemble recordings, we further apply transfer learning techniques to fine-tune a previously trained instrument recognition model for classifying six jazz solo instruments. Our results indicate that both source separation as pre-processing step as well as transfer learning clearly improve recognition performance, especially for smaller subsets of highly similar instruments.

1. INTRODUCTION

Automatic Instrument Recognition (AIR) is a fundamental task in Music Information Retrieval (MIR) which aims at identifying all participating music instruments in a given recording. This information is valuable for a variety of tasks such as automatic music transcription, source separation, music similarity computation, and music recommendation, among others. In general, musical instruments can be categorized based on their underlying sound production mechanisms. However, various aspects of human music performance such as dynamics, intonation, or vibrato create a large timbral variety that complicate the distinction of closely-related instruments such as a violin and a cello.

As part of the ISAD (Informed Sound Activity Detection in Music Recordings) research project, we aim at improving existing methods for timbre description and instru-

ment classification in ensemble music recordings. In particular, this paper focuses on the identification of predominant solo instruments in multitimbral music recordings, i. e., the most salient instruments in the audio mixture. This assumes that the spectral-temporal envelopes that describe the instrument’s timbre are dominant in the polyphonic mixture [11]. As a particular use-case, we focus on the classification of solo instruments in jazz ensemble recordings. Here, we study the task of instrument recognition both on a class and sub-class level, e. g. between soprano, alto, and tenor saxophone. Besides the high timbral similarity between different saxophone types, a second challenge lies in the large variety of recording conditions that heavily influence the overall sound of a recording [21, 25]. A system for jazz solo instrument classification could be used for content-based metadata clean-up and enrichment of jazz archives.

As the main contributions of this paper, we systematically evaluate two state-of-the-art source separation algorithms as pre-processing steps to improve instrument recognition (see Section 3). We extend and improve upon a recently proposed hybrid neural network architecture (see Figure 1) that combines convolutional layers for automatic learning of spectral-temporal timbre features, and fully connected layers for classification [28]. We further evaluate transfer learning strategies to adapt a given neural network model to more specific classification use-cases such as jazz solo instrument classification, which require a more granular level of detail [13].

2. RELATED WORK

The majority of work towards automatic instrument recognition has focused on instrument classification of isolated note events or monophonic phrases and melodies played by single instruments. Considering classification scenarios with more than 10 instrument classes, the best-performing systems achieve recognition rates above 90%, as shown for instance in [14, 27].

In polyphonic and multitimbral music recordings, however, AIR is a more complicated problem. Traditional approaches rely on hand-crafted audio features designed to capture the most discriminative aspects of instrument timbres. Such features are based on different signal representations based on cepstrum [8–10, 29], group delay [5], or line spectral frequencies [18]. A classifier ensemble focus-



© Juan S. Gómez, Jakob Abeßer, Estefanía Cano. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Juan S. Gómez, Jakob Abeßer, Estefanía Cano. “Jazz Solo Instrument Classification with Convolutional Neural Networks, Source Separation, and Transfer Learning”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

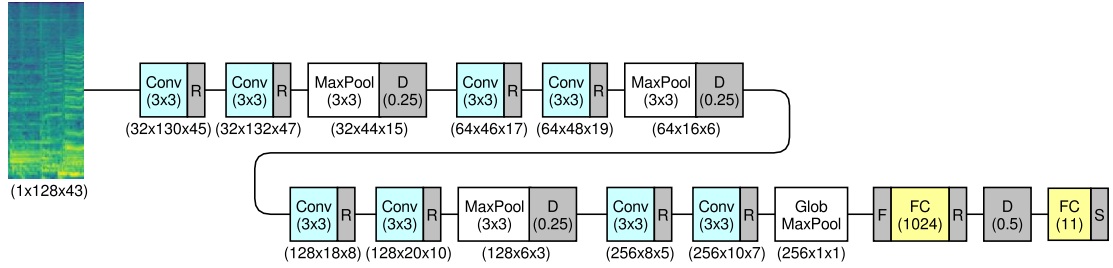


Figure 1. Reference model proposed by Han et al. [28]. Time-frequency spectrogram patches are processed by successive pairs of convolutional layers (Conv) with ReLU activation function (R), max pooling (MaxPool), and global max pooling (GlobMaxPool). Dropout (D) is applied for regularization in the feature extractor and classifier. Conv layers have increasing number of filters (32, 64, 128, and 256) and output shapes are specified for each layer.

ing on note-wise, frame-wise, and envelope-wise features was proposed in [14]. We refer the reader to [11] for an extensive overview of AIR algorithms that include hand-crafted audio features.

Novel deep learning algorithms, particularly convolutional neural networks (CNN), have been widely used for various image recognition tasks [13]. As a consequence, these methods were successfully adopted to MIR tasks such as chord recognition [17] and music transcription [1], where they significantly improved upon previous state-of-the-art results. Similarly, the first successful AIR methods based on deep learning were recently proposed and designed from the combination of convolutional layers for feature learning, and fully-connected layers for classification [24, 28]. Park et al. use a CNN to recognize instruments using single tone recordings [24]. Han et al. [28] propose a similar architecture and evaluate different late-fusion results to obtain clip-wise instrument labels. The authors aim at classifying predominant instruments in polyphonic and multitimbral recordings, and improve upon previous state-of-the-art systems by around 0.1 in f-score. Li et al. [20] propose to use end-to-end learning, considering a different network architecture. By these means, they use raw audio data as input without relying on spectral transformations such as mel spectrograms.

A variety of pre-processing strategies have been been applied MIR tasks such as singing voice detection [19] and melody line estimation [26]. Regarding the AIR task, several algorithms include a preceding source separation step. In [2], Bosch et al. evaluate two segregation methods for stereo recordings—a simple LRMS (Left/Right-Mid/Side) separation and FASST (Flexible Audio Source Separation Framework) developed by Ozerov et al. [22]. The authors report improvements of 19% in f-score using a simple panning separation, and up to 32% when the model was trained with previously separated audio, taking into account the typical artifacts produced by source separation techniques. Heittola et al. [16] propose a system that uses a source-filter model for source separation in a non-negative matrix factorization (NMF) scheme. The spectral basis functions are constrained to have harmonic spectra with smooth frequency responses. Using a Gaussian mixture model, the

authors achieved a 59% recognition rate for six polyphonic notes randomly chosen from 19 different instruments.

3. PROCESSING STEPS

3.1 Baseline Instrument Recognition Framework

In this section, we briefly summarize the instrument recognition model proposed by Han et al. [28], which we use as the starting point for our experiments. As a first step, monaural audio signals are processed at a sampling rate of 22.05 kHz. A mel spectrogram with a window size of 1024, a hop size of 512, and 128 mel bands is then computed. After applying a logarithmic magnitude compression, spectral patches one second long are used as input to the deep neural network. The resulting time-frequency patches have shape $x_i \in \mathbb{R}^{128 \times 43}$.

The network architecture is illustrated in Figure 1 and consists of four pairs of convolutional layers with a filter size of 3×3 and ReLU activation functions. The input of each convolution layer is zero-padded with 1×1 , considered in the output shape of each layer. The number of filters in the conv layer pairs increases from 32 to 256. Max pooling over both time and frequency is performed between successive layer pairs. Dropout of 0.25 is used for regularization. An intermediate global max pooling layer and flatten layer (F) connect the feature extractor with the classifier. Finally, a fully-connected layer (FC), dropout of 0.5, and a final output layer sigmoid activation (S) with 11 classes are used. The model was trained with a learning rate of 0.001, a batch size of 128, and the Adam optimizer.

In the post-processing stage, Han et al. compare two aggregation strategies to obtain class predictions on a audio file level: first, they apply thresholds over averaged and normalized segment-wise class predictions (S1 strategy). Secondly, a sliding window of 6 segments and hop-size 3 segments is used for local aggregation prior to performing S1 strategy (S2 strategy). Refer to [28] for the identification threshold estimation. Apart from the model ensembling step (which combines different predictors), we were able to reproduce the evaluation results reported in [28], in terms of recognition performance, intermediate activation function (ReLU), and the optimal identification threshold

Method	Model Ensembling	Data set	Activation Function	Agg.	Micro Averaging			Macro Averaging			Opt. θ
					P	R	F	P	R	F	
Baseline system [28]	✓	IRMAS	ReLU	S2	0.657	0.603	0.629	0.540	0.547	0.517	0.55
Reproduction	-	IRMAS	ReLU	S1	0.591	0.548	0.568	0.530	0.477	0.471	0.40
			ReLU	S2	0.609	0.544	0.574	0.501	0.507	0.475	0.55
Experiment	-	MONOTIMBRAL	LReLU	S1	0.645	0.678	0.661	0.685	0.681	0.657	0.8
			LReLU	S2	0.619	0.695	0.655	0.657	0.690	0.649	0.7

Table 1. Performance metrics precision (P), recall (R), and F-score (F) from best results reported by [28], its reproduction with the IRMAS data set, and an experiment with the MONOTIMBRAL data set. The displayed results are the best settings obtained with respect to ReLU/LReLU activation functions, and S1/S2 aggregation strategies (see Section 3.1).

θ as shown in Table 1. Additionally, an experiment was conducted using monotimbral audio as input data to train the neural network. Following [28], we tested different intermediate activation functions (ReLU and LReLU) and both aggregation strategies. The monotimbral audio used for this experiment is further explained in Section 4.2.

3.2 Source Separation

Motivated by the previous experiment, which showed that recognition performance increases 5-10% by using monotimbral data as input, we explore the use of sound source separation as a pre-processing stage to musical instrument classification. The idea is to evaluate whether isolating the desired instrument from the mixture can improve classification performance. This section briefly describes two sound separation methods used in our experiments.

3.2.1 Phase-based Harmonic / Percussive Source Separation

The harmonic-percussive separation described in [3] works under the assumption that harmonic music instrument will exhibit stable phase contours as the ones obtained by differentiating the phase spectrogram in time. In contrast, given the broadband and transient-like characteristics of percussive instruments, this stability in phase cannot be expected. This system takes advantage of this fundamental distinction between harmonic and percussive instruments, and by calculating the expected phase change for a given frequency bin and hop size, a separation mask is created to extract harmonic components from the mix. The effects of the harmonic-percussive separation can be observed in Figure 2, where the spectrogram of the original audio mixture and of the harmonic and percussive components are displayed.

3.2.2 Pitch-Informed Solo/Accompaniment Separation

To extract solo instruments from multitimbral music, the method proposed in [4] was also used in our experiments. The system performs separation by first extracting pitch information from the solo instrument, and then closely tracking its harmonic components to create a spectral mask. To extract pitch information, the method proposed in [7] is used for main melody extraction. Pitch information is extracted by performing a pair-wise evaluation of spectral peaks, and by finding partials with well-defined frequency ratios. The pitch information extracted is then used to

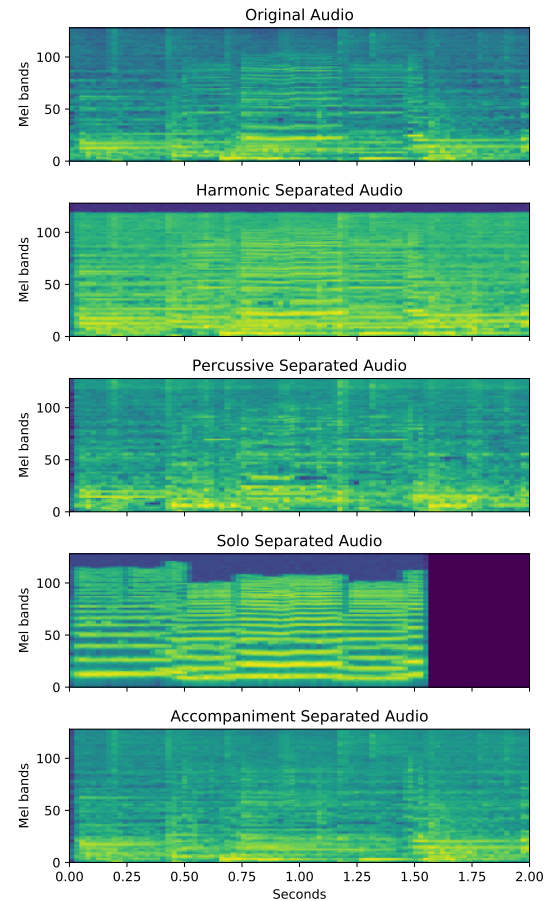


Figure 2. Mel-spectrograms of the original audio track, the harmonic/percussive components, and the solo/accompaniment components for a jazz excerpt of a saxophone solo played by John Coltrane. The audio mixture contains the solo saxophone, piano, bass and drums.

track the harmonic components in the separation stage, using common amplitude modulation, inharmonicity, attack length, and saliency as underlying concepts.

The performance of both the pitch detection and the separation stage in this system highly depend on the musical instrument to be separated: for musical instruments with clear, stable partials the separation performance can be very good. This is the case of woodwinds and string instruments such as the violin. However, for musical instru-

ments with a less stable spectral behavior such as the xylophone, or instruments with strong distortion effects such as electric guitars, separation can be noisy. The effects of the solo/accompaniment separation can be observed in Figure 2, where the spectrogram of the original audio mixture and of the solo and accompaniment components are displayed. It can be seen that starting from 1.50 seconds, the solo instrument is not detected and hence, no energy is assigned to the solo track.

3.3 Transfer Learning

For the special use-case of solo instrument recognition in jazz ensemble recordings, we aim at training a recognition model despite the small amount of available training data (see the JAZZ data set in Section 4.3). Here, transfer learning can be applied to fine-tune an existing classification model [13]. We assume that initially learnt feature representations for predominant AIR are highly relevant and therefore transferable for our use-case. Transfer learning has been successfully used in MIR for the task of sound event tagging in [6]. We refer the reader to [23] for a comprehensive overview of transfer learning in classification, regression, and clustering applications.

4. DATA SETS

4.1 IRMAS

The IRMAS data set (Instrument Recognition in Music Audio Signals) for predominant instrument recognition was first introduced by Bosch et al. in [2]. It is partitioned into separate training and test sets. The training set includes 6705 stereo audio files with a duration of 3 seconds each, extracted from more than 2000 recordings. All the recordings in the training data set are single-labeled and have a single predominant instrument. The amount of audio files per instrument is unevenly distributed and ranges from 388 to 778. The test set consists of 2874 stereo audio files with variable duration ranging from 5 to 20 seconds. These recordings are multi-labeled and cover 1-5 instrument labels per sample. The test set also shows a highly uneven instrument distribution with 62 to 1044 audio files per instrument class. As shown in Table 2, the data set contains 11 musical instruments: cello, clarinet, flute, acoustic guitar, electric guitar, organ, piano, saxophone, trumpet, violin, and singing voice. In the experiments described in Section 5.2.2, we use a subset denoted as IRMAS-Wind, which includes all recordings of the wind instruments in the IRMAS data set: flute, clarinet, saxophone, and trumpet. The motivation to create this subset is the improved performance of the solo/accompaniment separation algorithm (see section Section 3.2.1) and its timbral similarity to the JAZZ data set to apply transfer learning strategies (see Section 4.3). Following [28], training data was randomly split to training (85%) and validation (15%) to prevent overfitting by implementing early stopping. Testing data was randomly split into development testing data (50%) for optimum thresholding in post-processing, and

pure testing data (50%) to obtain the final performance metrics (see Table 3).

Instrument		IRMAS		MONO.		JAZZ	
Class	Subclass	#	h	#	h	#	h
Cello		499	0.87				
Clarinet		567	0.71	26	0.32	31	0.53
Flute		614	1.17	29	0.42		
Acoustic Guitar		1172	3.08	30	0.38		
Electric Guitar		1702	5.00				
	Clean			28	0.43		
	Distorted			30	0.34		
Organ		1043	2.25				
Hammond Organ				30	0.44		
Piano		1716	5.40	27	0.38		
Electric Piano				29	0.31		
Saxophone		952	2.16	29	0.34		
	Soprano					30	0.53
	Alto					29	0.53
	Tenor					32	0.53
Trombone						27	0.53
Trumpet		744	1.29	29	0.35	36	0.53
Violin		791	1.56	27	0.47		
Voice		1822	5.38				
	Female			21	0.26		
	Male			20	0.26		
Double Bass				27	0.28		
Synthesizer				30	0.77		
TOTAL		11622	28.87	412	5.75	185	3.18

Table 2. Overview of the three data sets IRMAS, MONO-TIMBRAL, and JAZZ, which includes various instrument classes and subclasses. Both the number of labels (#) and the total duration in hours (h) is given for each data set.

4.2 MONOTIMBRAL

The MONOTIMBRAL data set includes monotimbral (single-labeled) recordings, i.e., monophonic or polyphonic recordings without overlap of other instruments, of 15 musical instrument classes: acoustic guitar, clarinet, double bass, electric guitar clean, electric guitar distorted, electric piano, flute, hammond organ, piano, saxophone, female singing voice, male singing voice, synthesizer, trumpet, and violin. The data set contains 412 stereo audio files with variable duration from 10 to 120 seconds, manually selected from various segments of YouTube videos. The MONOTIMBRAL data set was randomly split equally into a training and test set based on an equal distribution of audio files per instrument class (see Table 3).

4.3 JAZZ

As one specific use-case, we aim at classifying among the six most popular brass and reed instruments in jazz solos: trumpet (tp), clarinet (cl), trombone (tb), alto saxophone (as), tenor saxophone (ts), and soprano saxophone (ss). While the number of instruments is smaller compared to the IRMAS and MONOTIMBRAL data sets, they have a higher timbral similarity, considering particularly the three saxophone subclasses. In order to prepare a data set, we first randomly selected solos from the Weimar Jazz Database [25] and enriched the data set with additional jazz solos. While the number of instruments is smaller compared to the IRMAS and MONOTIMBRAL data sets, the audio samples were chosen to maximize diversity of

performing artists. Moreover, examples from each class were randomly selected to have the same duration (see Table 2), achieving equal distribution of spectrogram examples across instrument classes. As with the other data sets, the JAZZ data set split randomly as the other data sets (see Table 3). Since jazz recordings cover many decades of the 20th century, the instrument recognition task is further complicated by different recording techniques.

For additional information regarding the MONOTIMBRAL and JAZZ data sets, refer to the complimentary website for this paper [12].

	Training Data Set (85/15)		Testing Data Set (50/50)	
	Train	Validation	Development	Pure
IRMAS	17094	3021	48064	48055
IRMAS-Wind	5486	970	10447	10446
Monotimbral	8676	1539	10620	10610
JAZZ	7206	1275	1678	1271

Table 3. Number of mel spectrogram examples for each data set split into Train, Validation, Development, Pure data sets.

5. EVALUATION

5.1 Metrics

Following [2, 11, 28], precision, recall, and f-scores were calculated for both micro and macro averages. Micro averaging gives more weight to instrument classes with higher appearance in the data distribution. Macro averaging is calculated per label, representing an overall performance of the system.

5.2 Improving Predominant Instrument Recognition using Source Separation

5.2.1 Harmonic / Percussive Separation

After processing the audio files with the harmonic/percussive separation introduced in Section 3.2.1, we first retrained the baseline model independently on the harmonic stream and percussive stream. Furthermore, we created a two-branch model that processes the harmonic and percussive stream in parallel and fuses the results in the final fully-connected layers, similar to [15]. As shown in Figure 3, using the harmonic stream marginally improved recognition results for both aggregation strategies S1 and S2 by up to 3% in f-score for the multitimbral IRMAS data set. In contrast, we did not observe an improvement for the MONOTIMBRAL data set. Using the two-branch model did not improve the performance on the IRMAS data set and worsens the performance on the MONOTIMBRAL data set.

5.2.2 Solo / Accompaniment Separation

The aim of performing this separation is to further improve the quality of the input audio to the classification system. All experiments described in this section were performed on the IRMAS-Wind and the JAZZ data sets (see Section 4), given the performance of the

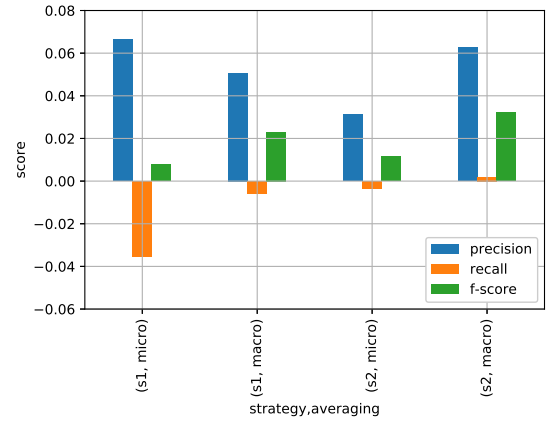


Figure 3. Comparison of the AIR system trained on the harmonic stream and the baseline model trained with the original IRMAS data set. Differences between evaluation metrics are shown for both aggregation strategies S1 and S2 (compare Section 3.1) as well as micro and macro averaging (compare Section 5.1).

solo/accompaniment algorithm. Both data sets also have similar timbral characteristics, which represents our targeted scenario.

We compare AIR models trained on the original audio tracks with models trained on the solo stream obtained from the solo/accompaniment separation. As shown in Table 4, applying the solo/accompaniment separation as pre-processing step improves the AIR performance by 3.8% in macro f-score for the IRMAS-Wind data set and 13.4% for the JAZZ data set using the S1 strategy. Additionally both micro and macro averages result in similar values, given the even distribution of examples of the JAZZ data set. The results might also indicate that error propagation from transcription errors to the source separation algorithm are not critical, since the instrument recognition results are averaged over time and the approximate accuracy of the pitch detection algorithm is 80% [7].

Data set	S/A Separation	F-Score	
		Micro	Macro
IRMAS-Wind	-	0.684	0.598
IRMAS-Wind	✓	0.713	0.636
JAZZ	-	0.657	0.669
JAZZ	✓	0.805	0.803

Table 4. Performance metrics obtained by training the baseline model with the IRMAS-Wind and JAZZ data sets. Best results were obtained using aggregation strategy S1.

5.3 Combining Source Separation and Transfer Learning for Jazz Solo Instrument Recognition

For our final use-case of recognizing jazz solo instruments, we aim at combining solo/accompaniment separation and transfer learning strategies. We use the models trained on the IRMAS-Wind data set (with and with-

out solo/accompaniment separation) as starting point for the transfer learning approach. All models were trained from scratch following the original parameters from [28]. The JAZZ data set includes recordings from trombone and three saxophone subclasses: tenor, alto, and soprano. Additionally, the trumpet and the clarinet classes were already included in the IRMAS-Wind data set. One main challenge is that while the characteristics of the predominant melody instruments in the IRMAS and JAZZ data sets are similar, the background instrumentation and recording conditions are often very different. We remove the last sigmoid layers of models pre-trained with the IRMAS-Wind data set and replace them by a 6-class sigmoid layer, considering the JAZZ data set. For testing, we compare two approaches: (1) the one-pass method which re-trains the last classification layer using a learning rate of $\alpha = 0.01$ (10 times the original learning rate), while all remaining layers remain fixed, and (2) the two-pass approach where we further re-train all layers in a second training step with a smaller learning rate of $\alpha = 0.001$. Table 5 shows the classification performance on the JAZZ data set for different system configurations with the one-pass and two-pass strategies, as well as with and without the solo/accompaniment separation. The best performance was achieved by combining solo/accompaniment separation and the two-pass transfer learning strategy.

S/A Separation	Transfer Learning	F-score	
		Micro	Macro
-	One-pass	0.605	0.621
✓	One-pass	0.738	0.748
-	Two-pass	0.583	0.610
✓	Two-pass	0.787	0.780
✓	-	0.805	0.803

Table 5. Performance metrics obtained by combining solo/accompaniment separation with transfer learning on the JAZZ data set. The results obtained by training the model from scratch (without transfer learning) are also shown in the bottom row for reference. Best results were obtained using aggregation strategy S1.

It can also be observed that the transfer learning model shows a lower macro f-measure of 0.780 than the model trained from scratch with 0.803 (see bottom row of Table 5). To further understand this behavior, six additional 10 s (unseen) jazz solo excerpts¹ were analyzed. Figure 4 shows segment- and clip-wise predictions for these six solo excerpts using solo/accompaniment separation. The figure shows the results for the best transfer learning system and the model trained on the JAZZ data set from scratch [12]. A total of 20 predictions were generated per excerpt on 1 s long windows using a 50 % overlap. These results suggest that transfer learning can improve generalization of unseen data, but needs further systematic investigations on a larger testing data set.

¹ Ornette Coleman - Ramblin (as), Buddy DeFranco - Autumn Leaves (cl), John Coltrane - My Favorite Things (ss), Frank Rossolino - Moonlight in Vermont (tb), Lee Morgan - The Sidewinder (tp), Michael Brecker - African Skies (ts)

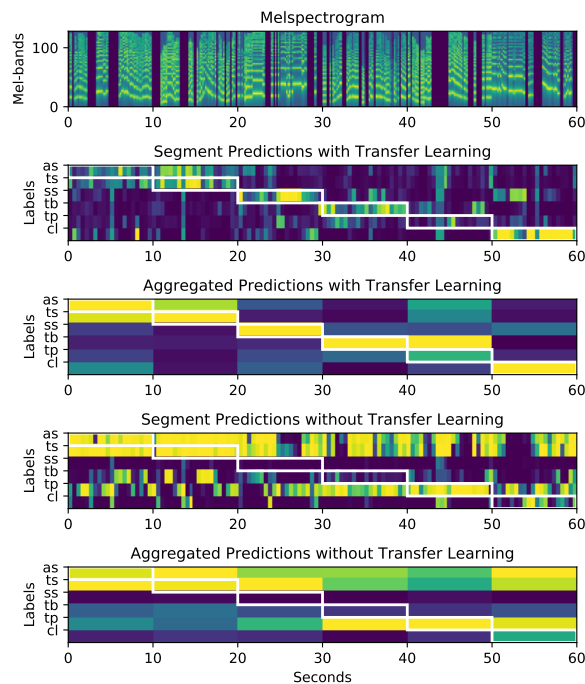


Figure 4. Mel-spectrogram of 10 second excerpts from six jazz solos covering all solo instruments (top), segment-wise and aggregated clip-wise predictions (using strategy S1) are shown below for a model trained via transfer learning (two-pass) and a model trained from scratch. Clip-wise ground truth is plotted in white rectangles [12].

6. CONCLUSION

In this paper, we investigated two methods to improve upon a system for AIR on multitimbral ensemble recordings. We first evaluated two state-of-the-art source separation methods and showed that on multitimbral audio data, analyzing the harmonic and solo streams can be beneficial compared to the mixed audio data.

For the specific use-case of jazz solo instrument classification, which involves classifying six instruments with high timbral similarity, combining solo/accompaniment source separation and transfer learning methods seems to lead to AIR models with better generalization to unseen data. This must be further investigated by increasing the size of the JAZZ data set. While source separation allows to narrow the focus on the predominant instrument, transfer learning allows to exploit useful feature representations learned from related instruments. In the future, a deep learning model capable of discriminating highly similar instruments could potentially be applied in other timbre-related recognition tasks such as performer identification [25].

7. ACKNOWLEDGEMENTS

This work has been supported by the German Research Foundation (AB 675/2-1).

8. REFERENCES

- [1] Rachel M. Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan P. Bello. Deep salience representations for f0 estimation in polyphonic music. In *Proceedings of the International Society of Music Information Retrieval (ISMIR)*, Suzhou, China, October 2017.
- [2] Juan Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera. A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 559–564, Porto, Portugal, 2012.
- [3] Estefanía Cano, Mark D. Plumbley, and Christian Dittmar. Phase-based harmonic/percussive separation. In *Proceedings of the Annual Conference of the International Speech Communication Association (INTER-SPEECH)*, pages 1628–1632, Singapore, 2014.
- [4] Estefanía Cano, Gerald Schuller, and Christian Dittmar. Pitch-informed solo and accompaniment separation towards its use in music education applications. *EURASIP Journal on Advances in Signal Processing*, 23:1–19, 2014.
- [5] Aleksandr Diment, Padmanabhan Rajan, Toni Heittola, and Tuomas Virtanen. Modified group delay feature for musical instrument recognition. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research*, pages 431–438, Marseille, France, 2013.
- [6] Aleksandr Diment and Tuomas Virtanen. Transfer learning of weakly labelled audio. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 6–10, New Paltz, USA, 2017.
- [7] Karin Dressler. *Automatic transcription of the melody from polyphonic music*. PhD thesis, TU Ilmenau, Germany, Jul 2017.
- [8] Zhiyao Duan, Bryan Pardo, and Laurent Daudet. A novel cepstral representation for timbre modeling of sound sources in polyphonic mixtures. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7495–7499, Florence, Italy, May 2014.
- [9] Antti Eronen and Anssi Klapuri. Musical instrument recognition using cepstral coefficients and temporal features. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 753–756, Istanbul, Turkey, 2000.
- [10] Slim Essid, Gael Richard, and Bertrand David. Musical instrument recognition on solo performances. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 1289–1292, Vienna, Austria, 2004.
- [11] Ferdinand Fuhrmann. *Automatic musical instrument recognition from polyphonic music audio signals*. PhD thesis, Universitat Pompeu Fabra, 2012.
- [12] Juan S. Gómez, Jakob Abeßer, and Estefanía Cano. Complementary website. https://github.com/dfg-isad/ismir_2018_instrument_recognition.
- [13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [14] Mikus Grasis, Jakob Abeßer, Christian Dittmar, and Hanna Lukashevich. A multiple-expert framework for instrument recognition. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research (CMMR)*, Marseille, France, October 2013.
- [15] Thomas Grill and Jan Schlüter. Music Boundary Detection Using Neural Networks on Spectrograms and Self-Similarity Lag Matrices. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Nice, France, 2015.
- [16] Toni Heittola, Anssi Klapuri, and Tuomas Virtanen. Musical instrument recognition in polyphonic audio using source-filter model for sound separation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 327–332, Kobe, Japan, 2009.
- [17] Filip Korzeniowski and Gerhard Widmer. A fully convolutional deep auditory model for musical chord recognition. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing, MLSP*, pages 1–6, Salerno, Italy, 2016.
- [18] A. G. Krishna and T. V. Sreenivas. Music instrument recognition: from isolated notes to solo phrases. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages 265–268, Quebec, Canada, 2004.
- [19] Simon Leglaive, Romain Hennequin, and Roland Badeau. Singing voice detection with deep recurrent neural networks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–125, Brisbane, Australia, April 2015.
- [20] Peter Li, Jiyuan Qian, and Tian Wang. Automatic instrument recognition in polyphonic music using convolutional neural networks. *CoRR*, abs/1511.05520, 2015.
- [21] Daniel Matz, Estefanía Cano, and Jakob Abeßer. New sonorities for early jazz recordings using sound source separation and automatic mixing tools. In *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, pages 749–755, Malaga, Spain, 2015.

- [22] Alexey Ozerov, Emmanuel Vincent, and Frederic Bimbot. A general flexible framework for the handling of prior information in audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(4):1118–1133, May 2012.
- [23] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.
- [24] Taejin Park and Taejin Lee. Musical instrument sound classification with deep convolutional neural network using feature fusion approach. *CoRR*, abs/1512.07370, 2015.
- [25] Martin Pfeleiderer, Klaus Frieler, Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart, editors. *Inside the Jazzomat - New Perspectives for Jazz Research*. Schott Campus, 2017.
- [26] H. Tachibana, T. Ono, N. Ono, and S. Sagayama. Melody line estimation in homophonic music audio signals based on temporal-variability of melodic source. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 425–428, Dallas, Texas, March 2010.
- [27] Steven Tjoa and K. J. Ray Liu. Musical instrument recognition using biologically inspired filtering of temporal dictionary atoms. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 435–440, Utrecht, The Netherlands, 2010.
- [28] Yoonchang Han and Jaehun Kim and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):208–221, Jan 2017.
- [29] Li-Fan Yu, Li Su, and Yi-Hsuan Yang. Sparse cepstral codes and power scale for instrument identification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7460–7464, Florence, Italy, 2014.

ALIGNED SUB-HIERARCHIES: A STRUCTURE-BASED APPROACH TO THE COVER SONG TASK

Katherine M. Kinnaird

Data Sciences Initiative and Division of Applied Mathematics

Brown University, USA

katherine.kinnaird@brown.edu

ABSTRACT

Extending previous structure-based approaches to the song comparison tasks such as the fingerprint and cover song tasks, this paper introduces the *aligned sub-hierarchies* (AsH) representation. Built by applying a post-processing technique to the aligned hierarchies of a song, the AsH representation is the set of unique aligned hierarchies for repeats (called AH^R) encoded in the original aligned hierarchies of the whole song. Effectively each AH^R within AsH is a section of the aligned hierarchies for the original song. Like aligned hierarchies, the AsH representation can be embedded into a classification space with a natural metric that makes inter-song comparisons based on sections of the songs. Experiments addressing a version of the cover song task on score-based data using AsH as the basis of inter-song comparison demonstrate potential of AsH-based approaches for MIR tasks.

1. INTRODUCTION

A common starting point in music information retrieval tasks is the creation of visualizations and representations for music-based data streams. One of the most influential representations is Foote’s self-similarity matrix (SSM) [4], which continues to be one of the most recognizable images in MIR. Much of the work starting with an SSM or a self-dissimilarity matrix (SDM) like [1, 5, 7, 9–12] create post-processing techniques that seek to enhance certain properties. This paper also offers a new post-processing technique that can be applied to either the SSM or SDM and extends the work of [7].

Under the music comparison tasks like the fingerprint task and the cover song task, structure-based approaches like [5, 7] seek to compare songs via their whole song representations. While this type of approach has varied success, there can be obvious issues. For example, with more rigid comparisons such as [7], whole song comparisons may only be meaningful if the songs have the same number of time-steps. Similarly whole song comparisons can fall

victim to large artistic choices. For example, [5] created a smoothed image of the thresholded and resampled SDM, which was then compared using the Euclidean distance. While effective, this approach stumbled comparing recordings of Mazurka Op. 68 No. 4 where Chopin neglected to include a *fine* marking, causing some pianists to play the piece twice, while others played the piece once [5].

The contrast to the whole song approach is using sections of a song. In [2, 3], audio shingles representing sections of recordings are compared to address the fingerprint, cover song, and remix tasks. In [17], the fingerprint task is tackled by comparing sections of recordings’ constellation maps marking their spectrogram peaks. Both approaches require access to the original audio signal.

This work introduces the aligned sub-hierarchies (AsH), a structure-based representation that can be used to compare songs based on sections of the songs. This AsH representation exists between the section-based comparison approaches like [2, 3, 17] and the structure-based approaches in [5, 7]. Furthermore, this representation embeds into a classification space with a natural metric that observes the triangle inequality.

The paper is organized as follows. Section 2 motivates the necessity of the extension of aligned hierarchies into AsH representation in context of MIR tasks. In Section 3, we formalize the definition of the AsH representation, detail the construction of AsH, and describe embedding AsH into a classification space with a natural metric. In Section 4, we use AsH representations to perform experiments for a version of the cover song task on a set of Mazurka scores. We offer future directions for research in Section 5.

2. MOTIVATION AND BACKGROUND

The aligned sub-hierarchies (AsH) representation is an extension of the aligned hierarchies from [7] that is motivated and inspired by making comparisons based on sections of songs or musical scores like those of [2, 3, 17]. This new representation seeks to combine the strengths of [5, 7] while addressing their limitations. Like their predecessor, AsH embeds into a classification space with a natural metric, but unlike in [7], the metric for AsH allows comparison between songs of differing lengths. Inspired particularly by the work in [5] stumbling on comparisons where some artists chose to repeat a song in its entirety, AsH seeks to note whether two songs share sections of unique structural



© Katherine M. Kinnaird. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Katherine M. Kinnaird. “Aligned sub-Hierarchies: a structure-based approach to the cover song task”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

decompositions at the original scale of those sections. In other words, in contrast to [5], there is no resampling in the formulation nor in the comparison of AsH.

Before detailing the AsH representation, we will present a summary of the aligned hierarchies introduced in [7]. This structure-based visualization catalogues all meaningful repeats present in music-based data streams, showing all possible structural hierarchies aligned on one common time axis. Each aligned hierarchies representation H comprises of three components: an onset matrix B_H , a length vector w_H , and an annotation vector α_H . There are as many rows in these three components as there are kinds of repetitions present in a music-based data stream, with each row being tied to a particular type of repeat. The binary matrix B_H encodes when each repeat begins with a entry of 1 at each starting time step. The vectors w_H and α_H together act as a key for B_H , while the former encodes the lengths of each type of repeat, and where the latter assigns annotations for the types of repeats so that types of the same length have different annotations. We can visualize the information contained in $H = (B_H, w_H, \alpha_H)$ as shown in Figures 1 and 3.1(b).

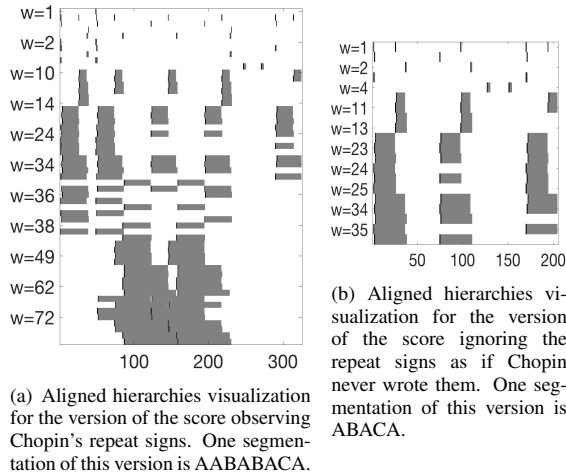


Figure 1. Aligned hierarchies for versions of Chopin's Mazurka Op. 6, No. 1 score, under threshold $T = 0.02$, with shingle size 12. Grey blocks denote repeats and rows denote types of repeats. Vertical labels are a subset of repeat lengths, while the horizontal ones mark the time steps.

The aligned hierarchies also provide an approach to the fingerprint task as they can be embedded into a classification space. However, nuanced comparisons can only be made between two aligned hierarchies with the exact same number of time steps [7]. This rigidity makes aligned hierarchies-based comparisons for the cover song task inappropriate since two cover songs are unlikely to be the same length. For certain MIR tasks, like the cover song task, we instead would like to compare parts of the songs' aligned hierarchies to each other, making comparisons based on these smaller aligned hierarchies a more suitable choice. This is the motivation and the inspiration for AsH. Take for example Figure 1 showing two versions of Chopin's Mazurka Op. 6, No. 1 score: Figure 1(a) shows

the aligned hierarchies for the score as Chopin intended and Figure 1(b) shows the aligned hierarchies for the score with Chopin's repeat signs ignored. Under one version of the cover song task, we would like to identify these two versions as being based off the same score, and just comparing these two aligned hierarchies, we notice similarities between the sections of the shown structural hierarchies.

3. ALIGNED SUB-HIERARCHIES

This section introduces both the aligned sub-hierarchies (AsH) representation and the classification space that AsH representations embed into. Starting with the aligned hierarchies for a song or a musical score, we isolate different repeated patterns and find the individual aligned hierarchies for each isolated pattern. The collection of the unique aligned hierarchies for sections of music-based data stream is called the *aligned sub-hierarchies*, abbreviated to *AsH*.

AsH is the result of a post-processing technique on aligned hierarchies that is similar to the result of treating sections of a song as songs in their own right and then finding each section's aligned hierarchies. Like other comparison methods like [2, 3, 17] that compare sections of songs to each other, the AsH finds all possible structural hierarchies for sections of each song. By leveraging structural information already encoded in aligned hierarchies for the whole song, we have already found all the repeated sections with smaller repeats within them and potentially keep additional structure information that would have been hidden had we build an aligned hierarchies directly from the song's section.

3.1 Defining AsH

In this section, we will formally define the AsH representation and consider a motivating example. The below definition for the aligned hierarchies of a given repeat R_i^k , particularly the third condition, ensures that we capture all possible song sections with their own aligned hierarchies.

Definition 3.1. Consider a song and let H be the aligned hierarchies for the song. Let R_i^k be a repeat of length k beginning at time step i , encoded in H . Then the set of repeats meeting the following three conditions form the *aligned hierarchies* of R_i^k or $AH^{R_i^k}$:

1. Encoded in H that are of length less than k ,
2. Contained in the set of time steps $[i, (i + k - 1)] \cap \mathbb{N}$,
3. Have at least one corresponding repeat that is also contained within the time steps $[i, (i + k - 1)] \cap \mathbb{N}$.

We encode $AH^{R_i^k}$ as $h_{R_i^k} = (B_H|_{R_i^k}, w_H|_{R_i^k}, \alpha_H|_{R_i^k})$, where each component of $h_{R_i^k}$ is defined similarly to those in $H = (B_H, w_H, \alpha_H)$. Here, the onset matrix $B_H|_{R_i^k}$ has k columns, encoding repeats in $h_{R_i^k}$ by their relative position to the start of R_i^k . For a general repeat (without a specified start or length), we say *aligned hierarchies of a repeat* and shorten to AH^R .

Example 3.1. Consider a song with the thresholded distance matrix \mathcal{T} shown in Figure 3.1(a). This song has three

kinds of structure: A (which occurs four times), B (which occurs five times), and C (which occurs only once). The aligned hierarchies for the song is shown in Figure 3.1(b).

We can build $\text{AH}^{R_1^{30}}$ from the aligned hierarchies by considering the blocks contained within the beats 1-30. In this case, those structures are the (ABA) structure on beats 1-25; the (BAB) structure on beats 11-30; the (AB) structure on beats 1-15 and 16-30; the (BA) structure on beats 11-25; the B structure on beats 11-15 and 26-30; and the two A structures on beats 1-10 and 16-25. Given that the A , B , and (AB) structures each repeat within beats 1-30, then by Definition 3.1, $\text{AH}^{R_1^{30}}$, encoded into $h_{R_1^{30}}$, contains these structures. A visualization for $h_{R_1^{30}}$ is shown in Figure 3.1(c).

Since the notion of time for $\text{AH}^{R_i^k}$ is relative to beat i , then for the Example 3.1, we have that $\text{AH}^{R_{36}^{30}}$ will encode the same information as $\text{AH}^{R_1^{30}}$. So $h_{R_1^{30}} = h_{R_{36}^{30}}$.

Definition 3.2. Let H be the aligned hierarchies for a song. The unordered set of *unique* AH^R representations denoted $\{h\} = \{h_1, h_2, \dots, h_m\}$ where each $h_i \in \{h\}$ is the AH^R for at least one repeat encoded in H is the *aligned sub-hierarchies* (or AsH) of the song.

Example 3.2. Consider Example 3.1 shown in Figure 3.1. The repeats (BAB) , (ABA) , $(ABAB)$ each have an AH^R . Although each of these occur twice, the AsH representation is comprised of only *unique* AH^R representations. So $\{h\} = \{h_{(BAB)}, h_{(ABA)}, h_{(ABAB)}\}$.

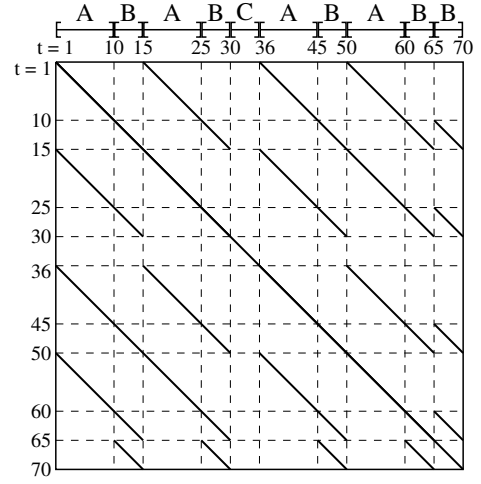
3.2 Building AsH from Aligned Hierarchies

In this section, we explain crafting $\{h\}$, the AsH representation of a song, from aligned hierarchies. First we detail constructing $\text{AH}^{R_i^k}$ for each repeat encoded in H and then explain when $\text{AH}^{R_i^k}$ is added to the AsH representation.

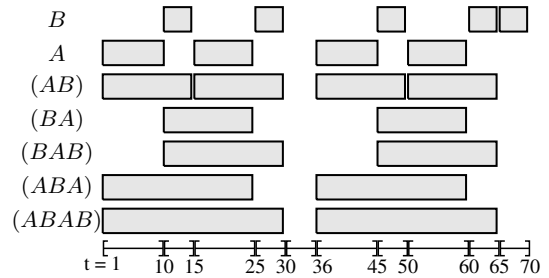
For each repeat in H , we note the starting time step i and the length of the repeat k , and then form $\text{AH}^{R_i^k}$ as follows. We first isolate the rows of $B_H \in H$ that correspond to repeats smaller than width k (that is the rows of B_H associated to the entries of $w_H < k$), and we further restrict this matrix to only the columns i through $(i + k - 1)$. We call the resulting sub-matrix $B_H|_{R_i^k}$. We also form the associated vectors $w_H|_{R_i^k}$ and $\alpha_H|_{R_i^k}$ as the entries of w_H and α_H that correspond to the rows of $B_H|_{R_i^k}$.

To satisfy Definition 3.1 as we build $\text{AH}^{R_i^k}$, we remove the rows of $B_H|_{R_i^k}$ that contain fewer than two repeats and then remove the corresponding entries in both $w_H|_{R_i^k}$ and $\alpha_H|_{R_i^k}$. Removing entries in $\alpha_H|_{R_i^k}$ may require adjusting the resulting values in $\alpha_H|_{R_i^k}$ so that for each repeat length k , the clusters of repeats of length k stored in $B_H|_{R_i^k}$ are identified with integers 1 through κ (that is, the number of clusters of repeats of length k stored in $B_H|_{R_i^k}$).

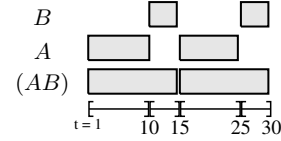
The resulting triple $h_{R_i^k} = (B_H|_{R_i^k}, w_H|_{R_i^k}, \alpha_H|_{R_i^k})$ is the $\text{AH}^{R_i^k}$ representation. We next check if $h_{R_i^k}$ is already in $\{h\}$, the unordered list of unique $\text{AH}^{R_i^k}$. If $h_{R_i^k} \notin \{h\}$, then $h_{R_i^k}$ is added to $\{h\}$.



(a) \mathcal{T} for the toy song with sections marked.



(b) Visualization for H of the toy song.



(c) Visualization for $h_{(ABAB)}$, which can be either denoted as $h_{R_1^{30}}$ or $h_{R_{36}^{30}}$, as they are equivalent.

Figure 2. Visualizations for toy song example with segmentation ABABCABABB.

By construction, a song's AsH can have AH^R of differing widths. If $h_i \in \{h\}$ is the AH^R for repeat R_i^k , then h_i is of width k .

3.3 Method for Comparing AsH

Comparing two AsH representations to each other requires finding the best alignments between collections of AH^R . This comparison must also respect that the AsH is an unordered set of AH^R representations and thus needs to be invariant to shifts in the ordering of the AH^R s. The AsH representation can be embedded into a space with a natural notion of distance, and this embedding leverages the embedding of the aligned hierarchies from Section 3 of [7].

3.3.1 Embedding AsH

As each AH^R is an aligned hierarchies representation, we start by embedding each AH^R into $(\mathcal{S}^*)^n$, the classification space for aligned hierarchies. To do this, a sequence of

binary matrices is created, where the k^{th} matrix is the rows of the aligned hierarchies associated to repeats of length k . The space \mathcal{S}^* is defined as \mathcal{S} / \sim , where \mathcal{S} is the space of $(m \times t)$ -binary matrices and where \sim is the equivalence relationship encoding that two matrices are equivalent if they are row permutations of each other.

In the case of AsH, we have a collection of AHR^R s each of which can be represented as an element of $(\mathcal{S}^*)^n$. By treating each $h_i \in \{h\}$ as a column of information, we consider p copies of $(\mathcal{S}^*)^n$. In this sense, we have a product space comprised of $(n \times p)$ -copies of the space \mathcal{S}^* , arranged into n rows and p columns, exactly like the entries of a $(n \times p)$ -matrix, with each element of AsH occupying a column of this matrix-like layout.

Since AsH is an unordered collection of AHR^R s, we need to define a space that is invariant to the ordering of the elements of AsH. We define the relationship \sim_* on $(\mathcal{S}^*)^{(n \times p)}$ such that two AsH representations are equivalent under \sim_* if they are different orderings of the same set of AHR^R representations. We can show that \sim_* is an equivalence relation on $(\mathcal{S}^*)^{(n \times p)}$, which allows us to define the following:

Definition 3.3. Let \mathcal{P} be the quotient space $(\mathcal{S}^*)^{(n \times p)} / \sim_*$. For $\{A\} \in \mathcal{P}$, the i -th column is A_i . We write $A_i \in \{A\}$ and call A_i a *column* of $\{A\}$.

The AsH representation $\{h\}$ of a song can be represented as an element of \mathcal{P} where $h_i \in \{h\}$, the i -th AHR^R , gets placed in the i -th column and so $\{h\} \in \mathcal{P}$. This space \mathcal{P} encodes the invariance of the ordering of the AHR^R s in an AsH representation, due to the equivalence relation \sim_* , meaning that the AHR^R (for a given AsH representation) can be placed into \mathcal{P} in any order.

3.3.2 Metric on \mathcal{P}

To compare two songs via AsH, we find the pairs of the AHR^R from the first song with those from the second song that minimize the sum of the distances between the pairs. We first consider the AHR^R s within the AsH representations that are of a fixed length k . Then we add all the distances from the identified matchings across the possible values of k . The resulting sum encodes the total dissimilarity between the repeated patterns of all sizes present in all of AHR^R s contained within the two AsH representations.¹

To first compare AHR^R s of the same length, consider two AsH representations $\{A\} = \{A_1, A_2, \dots, A_q\}$ and $\{B\} = \{B_1, B_2, \dots, B_r\}$, with all AHR^R s of length k . Assuming that $q, r \in \mathbb{Z}_{\geq 0}$ and that $q \geq r$, and ensuring that we are comparing lists of the same lengths, we append $(q - r)$ empty AHR^R s to $\{B\}$, each of which is a row-vector of k zeros. We recall that $d_H : (\mathcal{S}^*)^n \times (\mathcal{S}^*)^n \rightarrow \mathbb{R}$ is the distance between two aligned hierarchies encoding the total dissimilarity between them.

Below, we define f_L that permutes the elements of $\{B\}$ to find the optimal matching of AHR^R s in $\{B\}$ to those in $\{A\}$. This sum of the distances between the pairs of AHR^R s is the minimum across all possible matchings.

¹ The proofs for the material in this section can be found in the author's doctoral thesis [6].

Proposition 3.1. Let $\{A\}, \{B\} \in \mathcal{P}$. Let S_p be the symmetric group of degree p . Define $f_L : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}$ as

$$f_L(\{A\}, \{B\}) = \min_{\sigma \in S_p} \sum_{i=1}^p d_H(A_i, B_{\sigma(i)})$$

Then the function $f_L : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}$ is a distance function.

Proof Sketch: Leveraging properties of the symmetric group and the fact that d_H is a distance function, it is a straight forward check of the four requirements of a distance function: non-negativity, observance that the distance between two objects is zero if they are equivalent, reflectivity, and obeying the triangle inequality.² \square

While f_L is a notion of total dissimilarity between two sets of AHR^R s all of the same fixed length, a song's AsH representation likely contains AHR^R s of differing lengths. To find the total dissimilarity across all possible lengths of AHR^R s within AsH representations we add up the f_L distances found across all values of k . For clarity, we use the following definitions and notation:

Definition 3.4. Let $\{A\} \in \mathcal{P}$ such that the AHR^R s of $\{A\}$ are not necessarily the same width. Define $\{A^k\}$ to be the AHR^R s of $\{A\}$ that are of width k .

Corollary 1. Let $\{A\}, \{B\} \in \mathcal{P}$. Let M be the largest width of the AHR^R s in $\{A\}$ or $\{B\}$. Let $d_P : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}$ be given by:

$$d_P(\{A\}, \{B\}) = \sum_{i=1}^M f_L(\{A^i\}, \{B^i\}).$$

Then d_P is also a distance function.

Proof Sketch: Using that f_L is a distance function, check that d_P satisfies the definition of a distance function. \square

We note that the comparison of two songs using AsH requires that both songs have AsH representations. Due to the fact that we require each AHR^R to be the aligned hierarchies for a section of the song, it is possible that there exists aligned hierarchies for a song, but not one AHR^R . This would happen when all sections within a song do not have smaller repeated structures that repeat within that section. In this case, the song has an *empty AsH representation* and note that no comparisons can be made between a song with an empty AsH representation and any other song.

4. COVER SONG EXPERIMENTS

To test the validity of the using AsH representation and its associated metric to approach MIR tasks, we apply the AsH-based comparison method to address a version of the cover song task for a score-based data set.³ Each experiment follows the below procedure:

² We can further prove that if we first find and remove exact matches for the AHR^R s in the two AsHs, then the distance between the remaining AHR^R s in $\{A\}$ and $\{B\}$ is the same as the distance between the full AsHs $\{A\}$ and $\{B\}$. This fact adds efficiency to the computation of $f_L(\{A\}, \{B\})$. This proof proceeds by induction on the number of unmatched exact matches between $\{A\}$ and $\{B\}$.

³ The code used for these experiments as well as those in [7] can be found at <https://github.com/kmkinnaid/ThesisCode/releases/tag/vT.final2>

1. Pre-process the songs by building audio shingles using s concatenated beat synchronous Chroma feature vectors, and then creating and thresholding an SDM using a global threshold T
2. Construct the aligned hierarchies as in [7]
3. Extract AsH representations from each aligned hierarchies representation
4. Compute pairwise distances between pairs of AsH representations under the d_P metric
5. Match two songs if they are mutual nearest neighbors of each other
6. Evaluate the resulting matchings compared to the ground truth using precision and recall scores

4.1 Score-Based Data Set

For the experiments in this work, we use a score-based data set comprised of 52 Mazurka musical scores by Chopin. For each score, we download two `**kern` files posted on the KernScore online database⁴ (see [13]). The first file observes the repeated signs as marked by Chopin in the score, while the second ignores these repeat signs as if they are not written at all. If a score has no marked repeat signs, we download the single `**kern` file twice, marking one copy as observing the repeat signs and the second copy as having the repeat signs ignored. We refer to the versions of the scores as songs.

In this data set, each time step is in terms of beats with one time step being equivalent to one beat. We use the `music21` Python library⁵ to extract the Chroma feature vectors for each beat in the song. For each time step, we encode local information by creating the audio shingles (like those in [2, 3]) that are s concatenated Chroma feature vectors, for a fixed integer s . As most Mazurkas have 3 beats per measure, in these experiments (like those in [7]), we set $s = 6$ or $s = 12$. This means that we encode two or four (three beats) bars into each audio shingle.

We then create \mathcal{D} , the SDM for each song, by computing cosine dissimilarity measure between all pairs of audio shingles. So for audio shingles a_i, a_j associated to time steps i and j respectively, we define

$$\mathcal{D}_{i,j} = \left(1 - \frac{\langle a_i, a_j \rangle}{\|a_i\|_2 \|a_j\|_2}\right).$$

Then each SDM is thresholded based on the chosen global threshold T , which denotes how similar two audio shingles must be to be considered repetitions of each other. In these experiments, we choose global thresholds that are associated to very small differences between collections of 3 to 5 notes. To make this choice, we used the framework presented in [8] to connect our choice of T to the number of additions to the C-maj chord one can make and still be considered a repeat of the C-maj chord under the threshold T . This thresholding method differs from those in the literature that choose a threshold based on a fixed percentage of

pairwise measures to be selected such as [1, 5, 11] or based on a fixed number of nearest neighbors as in [14–16].

We complete the processing of each song by extracting the associated aligned hierarchies from the thresholded SDM as done in [7]. To find the AsH representation for each song, we apply the post-processing steps outlined in subsection 3.2. The AsH is the basis for our inter-song comparison in the following experiments.

4.2 Experimental Setup

For this work, we define the cover song task as matching the score's `**kern` file with the repeat signs observed to the score's `**kern` file that ignore the repeat signs. After finding the AsH for each song, we compute the pairwise distances between the songs' representations using d_P , and store the results in a pairwise distance matrix \mathbb{D} . We then perform a mutual nearest neighbor matching, by treating each song as a query track. Therefore each experiment has a maximum of 104 possible matches as each song as another version of its score to match with.

For these experiments, the ground truth is the song list with their cover as given by the meta data of each `**kern` file. We compute precision and recall by comparing the experiment's resulting matches to the ground truth.

4.3 Results

Table 1 reports experimental results for $s \in \{6, 12\}$ and $T \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$. The 10 experiments have high precision rates but more modest recall rates. Since not all songs have an AsH representation for each value of s and T , the number of possible matches varies.

For each experiment, we note the number of feature vectors per audio shingle, s , and the threshold value T that determines when two time steps are said to be repeats of each other. The choice of s and T affects the number of non-zero entries in the thresholded SDM, which determines whether or not a song has an AsH representation. If a song does not have an AsH associated to it, then we remove the row and column in \mathbb{D} associated to that song from consideration as well as remove that song from the ground truth listing. Our computations for precision and recall are based the adjusted ground truth list. In addition to reporting the precision and recall values for each experiment, we also report the number of possible matches that could be made based on the choices of s and T .

4.4 Discussion

The above results demonstrate the usability of the AsH representation in approaching MIR tasks. What is more, these results expose both strengths and weaknesses of using AsH as the basis for inter-song comparisons.

In considering the above results, we note that this version of the cover song task differs from the typical presentation of the cover song task. Most cover songs follow the original composer's intended structure fairly closely. In these experiments, half of our data set closely follows the composer's intentions while the other half blatantly makes

⁴ <http://kern.humdrum.org/search?s=t&keyword=Chopin>

⁵ <http://web.mit.edu/music21/>

s	T	Possible Matches	Precision Rate	Recall Rate	Empty AsHs
6	0.01	62	1	0.774	35
	0.02	66	0.962	0.757	33
	0.03	74	0.931	0.730	26
	0.04	78	1	0.692	24
	0.05	88	1	0.727	15
12	0.01	34	1	0.882	68
	0.02	44	1	0.818	57
	0.03	46	0.85	0.739	54
	0.04	56	0.84	0.75	45
	0.05	62	0.929	0.839	39

Table 1. Results for AsH for score data set on 10 experiments varying s , the width of the audio shingles, and T , the threshold used on the SDM for each song

large changes to the structure of the pieces. It is uncommon (though not unheard of as discussed in [5]) to see such large structural changes between two recordings of the same piece, which is what makes this version of the cover song task more challenging. Under this version of the cover song task, the AsH-based method was able to achieve strong experimental results.

The data in these experiments differs from the typical data set for the cover song task. For this score data, there are only two natural versions of each score: one with the repeat signs observed and the second with the repeat signs ignored. This means that every song has exactly one cover song to match with, contrasting from the typical data set for cover song retrieval that has an unknown number of covers for each query song. While a mutual nearest neighbors matching condition makes sense for this score-based data set (and other similar collections), it does mean that the choice of what each query track matches to is not independent from each other. An adjustment to the matching condition would need to be made for a data set of audio tracks with varying numbers of cover songs.

We also note that for the experiments in [7] nearly every song could be represented by aligned hierarchies, regardless of the shingle size s or threshold value T . In contrast, for the same data set under the same values for s and T , we find several songs without an AsH representation data set, meaning that those songs cannot be represented by an AsH representation or compared to other songs' AsH representations. Songs will lack an AsH representation if none of the repeats in the aligned hierarchies have smaller repeated structures within them. We can add flexibility to our definition of what it means for two sections to be repetitions of each other by increasing the value of T . Understandably, as the value of T rises, so do the number of songs with AsH representations, meaning that an appropriate choice of T is crucial to comparison methods based on AsH representations. Even with this caveat, the results from the above experiments provide evidence in favor of the usability of AsH as a low-dimensional representation for high-dimensional sequential data with lots of repeated structure.

Finally, the AsH comparison method is based on an accumulation of structure-based comparisons between structure decompositions of sections of the query song (represented by the collection AH^R for the query) and the structure decompositions of sections in every other song in the data set (also via their collections of AH^R). As with the aligned hierarchies, the AsH-based comparisons are based on hierarchical structure decompositions of sections of songs and are more than just one level or one size of structure. What is more, each AH^R , like the aligned hierarchies, encode not one possible structure hierarchy, but all structure hierarchies that exist within that section of the song. Matchings via AsH will occur when two songs have several sections that share hierarchical structure decompositions. This is a far more nuanced matching than just matching based on one segmentation. This AsH-based comparison method is a starkly different approach than [17] which compares just one section of the query track to the other songs in the data set. This approach is reminiscent of the work in [3] that takes a truncated sum of the distances between pairs of audio shingles. The crucial difference between [3] and this work is that the former is based directly on the audio frequencies within a section of a song, while the latter is based on the lengths and positioning of repeats within sections of the song.

5. CONCLUSION

In this paper, we introduce the aligned sub-hierarchies (AsH) representation, an extension of the aligned hierarchies in [7] that allows for structure-based comparisons between sections of songs. This representation seeks to address limitations of the approach in [5] to the cover song task by creating a collection of unique structure representations for sections of each song within a data set. There is a mathematical framework underpinning AsH as shown by embedding AsH representation into a classification space with a natural metric. Finally, we address a version of the cover song task using AsH-based pairwise song comparisons on a score-based data set. These experiments provide a proof of concept for using the AsH representation for highly repetitive, sequential data and offer new insights into structure-based approaches to comparison tasks based on sections of songs.

By existing between music comparisons based on whole song representations like [5, 7] and those based on partial song representations like [17], the AsH representation opens several new avenues of research. In future work, we plan to explore the impact of relaxing the third condition in Definition 3.1, both from the theory angle of creating an appropriate metric, like the one in subsection 3.3 and from the practical angle of being able to efficiently address MIR tasks on large data sets. Further exploration of the impact of T and s on AsH is also needed. As the experiments presented here were limited to score-based data, we also plan to apply AsH-based comparisons to the cover song task on collections of audio recordings.

Acknowledgements

Part of this work is a portion of the author's doctoral thesis [6], which was partially funded by the GK-12 Program at Dartmouth College (NSF award #0947790). The author also thanks Scott Pauls, Michael Casey, Dan Ellis, Jessica Thompson, and Brian McFee for their feedback on the earlier versions of this work.

6. REFERENCES

- [1] J. Bello. Measuring structural similarity in music. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2013–2025, 2011.
- [2] M. Casey, C. Rhodes, and M. Slaney. Analysis of minimum distances in high-dimensional musical spaces. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(5):1015 – 1028, 2008.
- [3] M. Casey and M. Slaney. Fast recognition of remixed audio. *2007 IEEE International Conference on Audio, Speech and Signal Processing*, pages IV – 1425 – IV–1428, 2007.
- [4] J. Foote. Visualizing music and audio using self-similarity. *Proc. ACM Multimedia 99*, pages 77–80, 1999.
- [5] P. Grosche, J. Serrà, M. Müller, and J.Ll. Arcos. Structure-based audio fingerprinting for music retrieval. *Proc. of 13th ISMIR Conference*, pages 55–60, 2012.
- [6] K. M. Kinnaird. *Aligned Hierarchies for Sequential Data*. PhD thesis, Dartmouth College, 2014.
- [7] K. M. Kinnaird. Aligned hierarchies: A multi-scale structure based representation for music-based data streams. *Proc. of 17th ISMIR Conference*, 2016.
- [8] K. M. Kinnaird. Examining musical meaning in similarity thresholds. *Proc. of 18th ISMIR Conference*, 2017.
- [9] B. McFee and D. P. W. Ellis. Analyzing song structure with spectral clustering. In *Proc. of 15th ISMIR Conference*, 2014.
- [10] M. Müller. *Information Retrieval for Music and Motion*. Springer Verlag, 2007.
- [11] M. Müller, P. Grosche, and N. Jiang. A segment-based fitness measure for capturing repetitive structures of music recordings. *Proc. of 12th ISMIR Conference*, pages 615–620, 2011.
- [12] M. Müller and F. Kurth. Enhancing similarity for music audio analysis. *Proc. of ICASSP*, 2006.
- [13] C.S. Sapp. Online database of scores in the humdrum file format. *Proc. of 6th ISMIR Conference*, pages 664–665, 2005.
- [14] J. Serrà, M. Müller, P. Grosche, and J.Ll. Arcos. Unsupervised detection of music boundaries by time series structure features. *Proc. of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.
- [15] J. Serrà, M. Müller, P. Grosche, and J.Ll. Arcos. Unsupervised music structure annotation by time series structure features and segment similarity. *IEEE Transactions on Multimedia*, 16(5), 2014.
- [16] J. Serrà, X. Serra, and R.G. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11(093017), 2009.
- [17] A. L. Wang. An industrial-strength audio search algorithm. In *Proc. of 4th ISMIR Conference*, 2003.

AUDIO-TO-SCORE ALIGNMENT USING TRANSPOSITION-INVARIANT FEATURES

Andreas Arzt¹

Stefan Lattner^{1,2}

¹ Institute of Computational Perception, Johannes Kepler University, Linz, Austria

² Sony Computer Science Laboratories (CSL), Paris, France

andreas.arzt@jku.at

ABSTRACT

Audio-to-score alignment is an important pre-processing step for in-depth analysis of classical music. In this paper, we apply novel transposition-invariant audio features to this task. These low-dimensional features represent local pitch intervals and are learned in an unsupervised fashion by a gated autoencoder. Our results show that the proposed features are indeed fully transposition-invariant and enable accurate alignments between transposed scores and performances. Furthermore, they can even outperform widely used features for audio-to-score alignment on ‘un-transposed data’, and thus are a viable and more flexible alternative to well-established features for music alignment and matching.

1. INTRODUCTION

The task of synchronising an audio recording of a music performance and its score has already been studied extensively in the area of intelligent music processing. It forms the basis for multi-modal inter- and intra-document navigation applications [6, 10, 35] as well as for the analysis of music performances, where e.g. aligned pairs of scores and performances are used to extract tempo curves or learn predictive performance models [12, 39].

Typically, this synchronisation task, known as audio-to-score alignment, is based on a symbolic score representation, e.g. in the form of MIDI or MusicXML. In this paper, we follow the common approach of converting this score representation into a sound file using a software synthesizer. The result is a low-quality rendition of the piece, in which the time of every event is known. Then, for both sequences the same kinds of features are computed, and a sequence alignment algorithm is used to align the audio of the performance to the audio representation of the score, i.e. the problem of audio-to-score alignment is treated as an audio-to-audio alignment task. The output is a mapping, relating all events in the score to time points in the

performance audio. Common features for this task include a variety of chroma-based features [8, 14, 15, 25], features based on the semitone scale [4, 6], and mel-frequency cepstral coefficients (MFCCs) [13].

In this paper, we apply novel low-dimensional features to the task of music alignment. The features represent local pitch intervals and are learned in an unsupervised fashion by a gated autoencoder [23]. We will demonstrate how these features can be used to synchronise a recording of a performance to a transposed version of its score. Furthermore, as they are only based on a local context, the features can even cope with multiple transpositions within a piece with only minimal additional alignment error, which is not possible at all with common pitch-based feature representations.

The main contributions of this paper are (1) the introduction of novel transposition-invariant features to the task of music synchronisation, (2) an in-depth analysis of their properties in the context of this task, and (3) a direct comparison to chroma features, which are the quasi-standard for this task. A cleaned-up implementation of the code for the gated autoencoder used in this paper is publicly available¹. The paper is structured as follows. In Section 2, the features are introduced. Section 3 briefly describes the alignment algorithm we are using throughout the paper. Then, in Section 4 we present detailed experiments on piano music, including a comparison of different feature configurations, results on transposed scores, and a comparison with chroma features. In Section 5 we discuss the application of the features in the domain of complex orchestral music. Finally, Section 6 gives an outlook on future research directions.

2. TRANSPOSITION-INVARIANT FEATURES FOR MUSIC SYNCHRONISATION

Transposition-invariant methods have been studied extensively in music information retrieval (MIR), for example in the context of music identification [2], structure analysis [26], and content-based music retrieval [19, 21, 36]. However, so far there has been limited success in transposition-invariant audio-to-score alignment. Currently, a typical approach is to first try to identify the transposition, transform the inputs accordingly, and then apply common alignment



© Andreas Arzt, Stefan Lattner. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Andreas Arzt, Stefan Lattner. “Audio-to-Score Alignment using Transposition-invariant Features”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ see <https://github.com/SonyCSLParis/cgae-invar>

techniques (see e.g. [33]). Another option is to perform the alignment multiple times, with different transpositions (e.g. the twelve possible transposition options when using chroma-based features) and then select the alignment which produced the least alignment costs (see e.g. [34]).

These are cumbersome and error-prone methods. In this paper, we demonstrate how to employ novel transposition-invariant features for the task of score-to-audio alignment, i.e. the features themselves are transposition-invariant. These features have been proposed recently in [17] and their usefulness has been demonstrated for tasks like the detection of repeated (but possibly transposed) motifs, themes and sections in classical music.

The features are learned automatically from audio data in an unsupervised way by a gated autoencoder. The main idea is to try to learn a *relative* representation of the current audio frame, based on a small local context (i.e., n -gram, the previous n frames). During the training process, the gated autoencoder is forced to represent this target frame via its preceding frames in a relative way (i.e. via interval differences between the local context and the target frame).

In the following, we give a more detailed description of how these features are learned. Specifics about the training data we are using in this paper can be found in the respective sections on applying this approach to piano music (Section 4) and orchestral music (Section 5).

2.1 Model

Let $\mathbf{x}_t \in \mathbb{R}^M$ be a vector representing the energy distributed over M frequency bands at time t . Given a temporal context $\mathbf{x}_{t-n}^t = \mathbf{x}_{t-n} \dots \mathbf{x}_t$ (i.e. the input) and the next time slice \mathbf{x}_{t+1} (i.e. the target), the goal is to learn a mapping \mathbf{m}_t (i.e. the transposition-invariant feature vector at time t) which does not change when shifting \mathbf{x}_{t-n}^{t+1} up- or downwards in the pitch dimension.

Gated autoencoders (GAEs, see Figure 1) are fundamentally different to standard sparse coding models, like denoising autoencoders. GAEs are explicitly designed to learn relations (i.e., covariances) between data pairs by employing an element-wise product in the first layer of the architecture. In musical sequences, using a GAE for learning relations between pitches in the input and pitches in the target naturally results in representations of musical *intervals*. The intervals are encoded in the latent variables of the GAE as mapping codes \mathbf{m}_t (refer to [17] for more details on interval representations in a GAE). The goal of the training is to find a mapping for any input/target pair which transforms the input into the given target by applying the represented intervals. The mapping at time t is calculated as

$$\mathbf{m}_t = \sigma_h(\mathbf{W}_1 \sigma_h(\mathbf{W}_0(\mathbf{U}\mathbf{x}_{t-n}^t \cdot \mathbf{V}\mathbf{x}_{t+1}))), \quad (1)$$

where \mathbf{U} , \mathbf{V} and \mathbf{W}_k are weight matrices, and σ_h is the hyperbolic tangent non-linearity. The operator \cdot (depicted as a triangle in Figure 1) denotes the Hadamard (or element-wise) product of the filter responses $\mathbf{U}\mathbf{x}_{t-n}^t$ and $\mathbf{V}\mathbf{x}_{t+1}$,

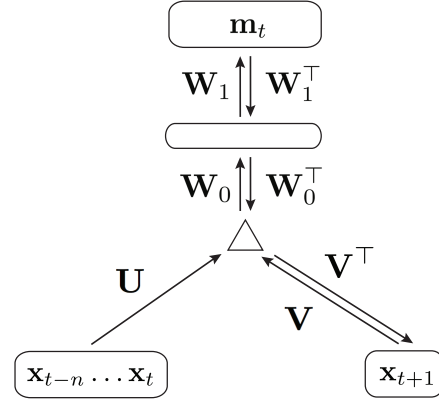


Figure 1. Schematic illustration of the gated autoencoder architecture used for feature learning. Double arrows denote weights used for both, inference of the mapping \mathbf{m}_t and the reconstruction of \mathbf{x}_{t+1} .

denoted as *factors*. The target of the GAE can be reconstructed as a function of the input \mathbf{x}_{t-n}^t and a mapping \mathbf{m}_t :

$$\tilde{\mathbf{x}}_{t+1} = \mathbf{V}^T(\mathbf{W}_0^T \mathbf{W}_1^T \mathbf{m}_t \cdot \mathbf{U}\mathbf{x}_{t-n}^t). \quad (2)$$

As cost function we use the mean-squared error between the target \mathbf{x}_{t+1} and the target's reconstruction $\tilde{\mathbf{x}}_{t+1}$ as

$$\text{MSE} = \frac{1}{M} \|\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}\|^2. \quad (3)$$

2.2 Training Data Preprocessing

Models are learned directly from audio data, without the need for any annotations. The empirically found preprocessing parameters are as follows. The audio files are re-sampled to 22.05 kHz. We choose a constant-Q transformed spectrogram using a hop size of 448 (~ 20 ms), and Hann windows with different sizes depending on the frequency bin. The range comprises 120 frequency bins (24 per octave), starting from a minimal frequency of 65.4 Hz. Each time slice is contrast-normalized to zero mean and unit variance.

2.3 Training

The model is trained with stochastic gradient descent in order to minimize the cost function (cf. Equation 3) using the training data as described in Section 2.2. In an altered training procedure introduced below, we randomly transpose the data during training and explicitly aim at transposition invariance of the mapping codes.

2.3.1 Enforcing Transposition-Invariance

As described in Section 2.1 the classical GAE training procedure derives a mapping code from an input/target pair, and subsequently penalizes the reconstruction error of the target given the input and the derived mapping code. Although this procedure naturally tends to lead to similar

mapping codes for input target pairs that have the same interval relationships, the training does not explicitly enforce such similarities and consequently the mappings may not be maximally transposition invariant.

Under ideal transposition invariance, by definition the mappings would be identical across different pitch transpositions of an input/target pair. Suppose that a pair $(\mathbf{x}_{t-n}^t, \mathbf{x}_{t+1})$ leads to a mapping \mathbf{m} (by Equation (1)). Transposition invariance implies that reconstructing a target \mathbf{x}_{t+1}' from the pair $(\mathbf{x}_{t-n}^t, \mathbf{m})$ should be as successful as reconstructing \mathbf{x}_{t+1} from the pair $(\mathbf{x}_{t-n}^t, \mathbf{m})$ when $(\mathbf{x}_{t-n}^t, \mathbf{x}_{t+1}')$ can be obtained from $(\mathbf{x}_{t-n}^t, \mathbf{x}_{t+1})$ by a single pitch transposition.

Our altered training procedure explicitly aims to achieve this characteristic of the mapping codes by penalizing the reconstruction error using mappings obtained from transposed input/target pairs. More formally, we define a transposition function $shift(\mathbf{x}, \delta)$, shifting the values (CQT frequency bins) of a vector \mathbf{x} of length M by δ steps:

$$shift(\mathbf{x}, \delta) = (x_{(0+\delta) \bmod M}, \dots, x_{(M-1+\delta) \bmod M})^\top, \quad (4)$$

and $shift(\mathbf{x}_{t-n}^t, \delta)$ denotes the transposition of each single time step vector *before* concatenation and linearization.

The training procedure is then as follows: First, the mapping code \mathbf{m}_{t+1} of an input/target pair is inferred as shown in Equation 1. Then, \mathbf{m}_{t+1} is used to reconstruct a *transposed* version of the target, from an equally *transposed* input (modifying Equation 2) as

$$\tilde{\mathbf{x}}'_{t+1} = \sigma_g(\mathbf{V}^\top (\mathbf{W}_0^\top \mathbf{W}_1^\top \mathbf{m}_t \cdot Ushift(\mathbf{x}_{t-n}^t, \delta))), \quad (5)$$

with $\delta \in [-60, 60]$ randomly chosen for each training batch. Finally, we penalize the error between the reconstruction of the transposed target and the actual transposed target (i.e., employing Equation 3) as

$$\text{MSE} = \frac{1}{M} \|shift(\mathbf{x}_{t+1}, \delta) - \tilde{\mathbf{x}}'_{t+1}\|^2. \quad (6)$$

This method amounts to both, a form of guided training and data augmentation.

2.3.2 Training Details

The architecture and training details of the GAE are as follows. In this paper, we use two models with differing n -gram lengths. The factor layer has 512 units for n -gram length $n = 16$, and 256 units for $n = 8$. Furthermore, for all models, there are 128 neurons in the first mapping layer and 64 neurons in the second mapping layer, i.e. the features we will be using throughout this paper for the alignment task are 64-dimensional.

L2 weight regularization for weights \mathbf{U} and \mathbf{V} is applied, as well as sparsity regularization [18] on the top-most mapping layer. The deviation of the norms of the columns of both weight matrices \mathbf{U} and \mathbf{V} from their average norm is penalized. Furthermore, we restrict these norms to a maximum value. We apply 50% dropout on the input and no dropout on the target, as proposed in [23]. The learning rate ($1e-3$) is gradually decremented to zero over 300 epochs of training.

3. ALIGNMENT ALGORITHM

The goal of this paper is to give the reader a good intuition about the novel transposition-invariant features for audio alignment and focus on their properties, without being distracted by a complicated alignment algorithm. Thus, we use a simple multi-scale variant of the dynamic time warping (DTW) algorithm (see [25] for a detailed description of DTW) for the experiments throughout the paper, namely FastDTW [32] with the radius parameter set to 50. We performed all experiments presented in this paper using the cityblock, Euclidean and cosine distance measures to compute distances between feature vectors. Because the choice of distance measure did not have a sizeable impact, we only report the results using the Euclidean distance. As FastDTW is a well-known and widely used algorithm, we refrain from describing the algorithm here in detail and refer the reader to the referenced works.

Obviously, a large number of more sophisticated alternatives to FastDTW exists. This includes methods based on hidden Markov and semi-Markov models [27–29], conditional random fields [16], general graphical models [5, 20, 30, 31], Monte Carlo sampling [7, 24], and extensions to DTW, e.g. multiple sequence alignment [37] and integrated tempo models [3]. We are confident that the presented features can also be employed successfully with these more sophisticated alignment schemes.

4. EXPERIMENTS ON PIANO MUSIC

In this section we present a number of experiments, showcasing the strengths of the proposed features as well as their weaknesses. We will do this on piano music first, before moving on to more complex orchestral music in Section 5. For learning the features, a dataset consisting of 100 random piano pieces of the MAPS dataset [9] (subset MUS) was used. As discussed in Section 2, no annotations are needed, thus actually any available audio recording of piano music could be used. For the experiments, we trained two models, differing in the size of their local context: an 8-gram model and a 16-gram model (referred to as *8G Piano* and *16G Piano* in the remainder of the paper).

For the evaluation of audio-to-score alignment, a collection of annotated test data (pairs of scores and exactly aligned performances) is needed. We performed experiments on four datasets (see Table 1). *CB* and *CE* consist of 22 recordings of the Ballade Op. 38 No. 1 and the Etude Op. 10 No. 3 by Chopin [11], *MS* contains performances of the first movements of the piano sonatas KV279-284, KV330-333, KV457, KV475 and KV533 by Mozart [38], and *RP* consists of three performances of the Prelude Op. 23 No. 5 by Rachmaninoff [1]. The scores are provided in the MIDI format. Their global tempo is set such that the score audio roughly matches the mean length of the given performances. The scores are then synthesised with the help of *timidity*² and a publicly available sound font. The resulting audio files are used as score representations for the alignment experiments.

²<https://sourceforge.net/projects/timidity/>

ID	Dataset	Files	Duration
CE	Chopin Etude	22	~ 30 min.
CB	Chopin Ballade	22	~ 48 min.
MS	Mozart Sonatas	13	~ 85 min.
RP	Rachmaninoff Prelude	3	~ 12 min.

Table 1. The evaluation data set for the experiments on piano music (see text).

In the experiments, we use two types of evaluation measures. For each experiment, the 1st quartile, the median, and the 3rd quartile of the absolute errors at aligned reference points is given. We also report the percentage of reference points which have been aligned with errors smaller or equal 50 ms, and smaller or equal 250 ms (similar to [6]).

4.1 Experiment 1: Feature Configurations

The first experiment compares the performance of the two feature configurations *8G Piano* and *16G Piano* on the piano evaluation set (see Table 2). The differences between the two configurations are relatively small, although the 8-gram feature consistently works slightly better than the 16-gram features. The danger of using a larger local context is that different tempi can lead to very different contexts (e.g. faster tempi result in more notes contained in the local context), which in turn leads to different features, which is a problem for the matching process. We will return to this problem in a later experiment (see Section 4.4). Because of space constraints, in the upcoming sections, we will only report the results for *8G Piano*.

4.2 Experiment 2: Transposition-invariance

Next, we demonstrate that the learned features are actually invariant to transpositions. To do so, we transposed the score representations by -3, -2, -1, 0, +1, +2 and +3 semitones and tried to align the untransposed performances to these scores. The results for the *8G Piano* features are shown in Table 3. The results for all transpositions, including the untransposed scores, are very similar. Only minor fluctuations occur randomly.

In addition, we prepared a second, more challenging experiment. We manipulated the scores such that after every 30 seconds another transposition from the set $\{-3, -2, -1, +1, +2, +3\}$ is randomly applied. From each score, we created five such randomly changing score representations and tried to align the performances to these scores. The results are shown in the rightmost column of Table 3. Again, there is no difference to the other results. Basically, the transpositions only lead to at most eight noisy feature vectors every time a new transposition is applied, which is not a problem for the alignment algorithm. We would also like to note that very few algorithms or features would be capable of solving this task (see [26] for another option). Other methods that first try to globally identify the transposition and then use traditional methods for the alignment are clearly not applicable here.

Dataset	Measure	8G Piano	16G Piano
CB	1 st Quartile	10 ms	11 ms
	Median	22 ms	24 ms
	3 rd Quartile	39 ms	45 ms
	Error \leq 50 ms	83%	79%
	Error \leq 250 ms	94%	95%
CE	1 st Quartile	10 ms	12 ms
	Median	21 ms	25 ms
	3 rd Quartile	36 ms	45 ms
	Error \leq 50 ms	87%	79%
	Error \leq 250 ms	96%	95%
MS	1 st Quartile	6 ms	6 ms
	Median	13 ms	14 ms
	3 rd Quartile	25 ms	26 ms
	Error \leq 50 ms	90%	91%
	Error \leq 250 ms	100%	100%
RP	1 st Quartile	14 ms	16 ms
	Median	34 ms	40 ms
	3 rd Quartile	90 ms	92 ms
	Error \leq 50 ms	63%	57%
	Error \leq 250 ms	90%	93%

Table 2. Comparison of the 8-gram and the 16-gram feature models.

4.3 Experiment 3: Comparison to Chroma Features

It is now time to compare the *8G Piano* features to well-established features for the task of music alignment in the normal, un-transposed alignment setting. To this end, we computed the *chroma_cqt* features³ (henceforth referred to as *Chroma*) as provided by *librosa*⁴ [22] (with standard parameters except for the normalisation parameter, which we set to 1; the hop size is roughly 20 ms), and aligned the performances to the scores. The results are shown in Table 4. On this dataset, the proposed *transposition-invariant* features consistently outperform the well-established *Chroma* features, which are based on *absolute pitches*. To summarise, so far the proposed features show state-of-the-art performance on the standard alignment task, while additionally being able to align transposed sequences to each other with no additional error.

4.4 Experiment 4: Robustness to Tempo Variations

Next, we have a closer look at the influence of different tempi on our features. As they are based on a fixed local context (a fixed number of frames), the tempo plays an important role in their computation. For example, if the tempo doubles, this means that musically speaking the local context is twice as large as at the normal tempo and additional notes might be included in this context, which would not be part of the local context in the case of the

³ We also tried the *CENS* features, which are a variation of chroma features, but as they consistently performed worse than the *Chroma* features, we are not reporting the results here.

⁴ Version 0.6, DOI:10.5281/zenodo.1174893

Dataset	Measure	Transposition in Semitones							Rand. Transp.
		-3	-2	-1	0	1	2	3	
CB	1 st Quartile	10 ms	10 ms	11 ms	10 ms	10 ms	11 ms	10 ms	10 ms
	Median	22 ms	22 ms	23 ms	22 ms	22 ms	23 ms	22 ms	22 ms
	3 rd Quartile	39 ms	39 ms	41 ms	39 ms	40 ms	40 ms	38 ms	39 ms
	Error \leq 50 ms	84%	83%	81%	83%	82%	83%	84%	83%
	Error \leq 250 ms	95%	94%	94%	94%	93%	95%	95%	95%
CE	1 st Quartile	10 ms	10 ms	8 ms	10 ms	9 ms	9 ms	10 ms	9 ms
	Median	21 ms	20 ms	18 ms	21 ms	19 ms	18 ms	20 ms	19 ms
	3 rd Quartile	37 ms	32 ms	30 ms	36 ms	33 ms	32 ms	33 ms	32 ms
	Error \leq 50 ms	83%	90%	91%	87%	89%	88%	90%	90%
	Error \leq 250 ms	93%	97%	98%	96%	97%	98%	97%	97%
MS	1 st Quartile	6 ms	6 ms	6 ms	6 ms	6 ms	6 ms	6 ms	6 ms
	Median	13 ms	13 ms	13 ms	13 ms	13 ms	14 ms	13 ms	13 ms
	3 rd Quartile	24 ms	24 ms	24 ms	25 ms	25 ms	26 ms	25 ms	25 ms
	Error \leq 50 ms	91%	91%	92%	90%	91%	90%	90%	91%
	Error \leq 250 ms	100%	100%	100%	100%	100%	100%	100%	100%
RP	1 st Quartile	17 ms	16 ms	15 ms	14 ms	13 ms	12 ms	15 ms	14 ms
	Median	45 ms	44 ms	36 ms	34 ms	35 ms	31 ms	34 ms	37 ms
	3 rd Quartile	136 ms	151 ms	106 ms	90 ms	130 ms	90 ms	103 ms	122 ms
	Error \leq 50 ms	53%	53%	60%	63%	59%	64%	60%	58%
	Error \leq 250 ms	84%	83%	88%	90%	85%	90%	89%	86%

Table 3. Results for 8-gram piano features on transposed versions of the scores (from -3 to +3 semitones). The rightmost column gives the results on scores with randomly changing transpositions after every 30 seconds (see Section 4.2).

DS	Measure	Chroma	8G Piano
CB	1 st Quartile	15 ms	10 ms
	Median	34 ms	22 ms
	3 rd Quartile	80 ms	39 ms
	Error \leq 50 ms	64%	83%
	Error \leq 250 ms	85%	94%
CE	1 st Quartile	13 ms	10 ms
	Median	29 ms	21 ms
	3 rd Quartile	56 ms	36 ms
	Error \leq 50 ms	71%	87%
	Error \leq 250 ms	94%	96%
MS	1 st Quartile	7 ms	6 ms
	Median	16 ms	13 ms
	3 rd Quartile	31 ms	25 ms
	Error \leq 50 ms	85%	90%
	Error \leq 250 ms	98%	100%
RP	1 st Quartile	17 ms	14 ms
	Median	43 ms	34 ms
	3 rd Quartile	113 ms	90 ms
	Error \leq 50 ms	55%	63%
	Error \leq 250 ms	91%	90%

Table 4. Comparison of the transposition-invariant *8G Piano* features to the *Chroma* features on untransposed scores.

normal tempo. To test the influence of tempo differences, we created score representations using different tempi and aligned the unchanged performances to them. Table 5 summarises the results for the *Chroma* and the *8G Piano* features on scores synthesised with the base tempo, as well as with $\frac{2}{3}$ -times and $\frac{4}{3}$ -times the base tempo. Unsurprisingly, tempo in general influences the alignment results. However, while the *Chroma* features are much more robust to differences in tempo between the sequences to be aligned, the *8G Piano* features struggle in this experiment. We repeated the experiment with more extreme tempo changes, which confirmed this trend. While with the *Chroma* features it is possible to more or less align sequences with tempo differences of a factor of three, the transposition-invariant features fail in these cases.

5. FIRST EXPERIMENTS ON ORCHESTRAL MUSIC

In addition to the promising results on piano music, we also present first experiments on orchestral music. To this end, we trained an additional model on recordings of symphonic music (seven full commercial recordings of symphonies by Beethoven, Brahms, Bruckner, Berlioz and Strauss), which will be referred to in the following as *8G Orch*. For comparison, we also evaluated the model from the previous section (*8G Piano*) and the *Chroma* features on the evaluation data. The evaluation data consists of two recordings of classical symphonies: the 3rd symphony by Beethoven (B3) and the 4th symphony by Mahler (M4).

DS	Measure	Chroma			8G Piano		
		$\frac{2}{3}$ Tempo	Base Tempo	$\frac{4}{3}$ Tempo	$\frac{2}{3}$ Tempo	Base Tempo	$\frac{4}{3}$ Tempo
CB	1 st Quartile	19 ms	15 ms	13 ms	24 ms	10 ms	32 ms
	Median	43 ms	34 ms	29 ms	63 ms	22 ms	120 ms
	3 rd Quartile	137 ms	80 ms	66 ms	116 ms	39 ms	205 ms
	Error ≤ 50 ms	54%	64%	67%	47%	83%	33%
	Error ≤ 250 ms	82%	85%	85%	87%	94%	84%
CE	1 st Quartile	14 ms	13 ms	12 ms	27 ms	10 ms	26 ms
	Median	30 ms	29 ms	25 ms	70 ms	21 ms	83 ms
	3 rd Quartile	65 ms	56 ms	53 ms	116 ms	36 ms	176 ms
	Error ≤ 50 ms	69%	71%	73%	40%	87%	38%
	Error ≤ 250 ms	90%	94%	94%	93%	96%	80%
MS	1 st Quartile	8 ms	7 ms	9 ms	7 ms	6 ms	9 ms
	Median	18 ms	16 ms	20 ms	16 ms	13 ms	21 ms
	3 rd Quartile	42 ms	31 ms	49 ms	33 ms	25 ms	52 ms
	Error ≤ 50 ms	79%	85%	75%	84%	90%	74%
	Error ≤ 250 ms	98%	98%	97%	99%	100%	98%
RP	1 st Quartile	18 ms	17 ms	20 ms	22 ms	14 ms	30 ms
	Median	44 ms	43 ms	58 ms	69 ms	34 ms	86 ms
	3 rd Quartile	116 ms	113 ms	141 ms	184 ms	90 ms	202 ms
	Error ≤ 50 ms	53%	55%	56%	43%	63%	37%
	Error ≤ 250 ms	92%	91%	87%	82%	90%	85%

Table 5. Comparison of Chroma and 8G Piano features for alignments to scores in different tempi (see Section 4.4).

DS	Measure	Chroma	8G Piano	8G Orch
B3	1 st Quartile	20 ms	25 ms	22 ms
	Median	48 ms	54 ms	49 ms
	3 rd Quartile	108 ms	104 ms	111 ms
	Err. ≤ 50 ms	52%	47%	51%
	Err. ≤ 250 ms	88%	90%	89%
M4	1 st Quartile	46 ms	50 ms	57 ms
	Median	110 ms	129 ms	142 ms
	3 rd Quartile	278 ms	477 ms	535 ms
	Err. ≤ 50 ms	27%	25%	23%
	Err. ≤ 250 ms	73%	66%	62%

Table 6. Comparison of the transposition-invariant and chroma features on orchestral music (see Section 5).

Both have been manually annotated at the downbeat level.

In this alignment experiment, the *Chroma* features outperform both the *8G Piano* and the *8G Orch* features, especially on the symphony by Mahler (see Table 6). We mainly contribute this to the fact that these rather long recordings contain a number of sections with different tempi, which is not reflected in the score representations. As has been established in Section 4.4, the transposition-invariant features struggle in these cases. Still, we will have to further investigate the use of these features for orchestral music.

It is interesting to note that *8G Piano* gives slightly better results than *8G Orch*, even though this dataset solely consists of orchestral music. It turns out that the learned

features are very general and can be readily applied to different instruments. We also tried to overfit on the test data, i.e., we trained a feature model using the audio files we would later use for the alignment experiments. Even this approach only led to fractionally better results.

6. CONCLUSIONS

In this paper, we reported on audio-to-score alignment experiments with novel transposition-invariant features. We have shown that the features are indeed fully invariant to transpositions and in many settings can outperform the quasi-standard features for this task, namely chroma-based features. On the other hand, we also demonstrated the weaknesses of the transposition-invariant features, especially their fragility regarding different tempi, which is a serious limitation in the context of alignment tasks.

In the future, we will study this weakness in depth and will try to alleviate this problem. Ideas include further experiments with different n-gram lengths, the adoption of alignment schemes including tempo models which iteratively adapt the local tempi of the representations, and to try to include tempo-invariance as an additional goal in the learning process of the features.

7. ACKNOWLEDGEMENTS

This research has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (ERC grant agreement No. 670035, project CON ESPRESSIONE).

8. REFERENCES

- [1] Andreas Arzt. *Flexible and Robust Music Tracking*. PhD thesis, Johannes Kepler University Linz, 2016.
- [2] Andreas Arzt, Sebastian Böck, and Gerhard Widmer. Fast identification of piece and score position via symbolic fingerprinting. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 433–438, Porto, Portugal, 2012.
- [3] Andreas Arzt and Gerhard Widmer. Simple tempo models for real-time music tracking. In *Proceedings of the Sound and Music Computing Conference (SMC)*, Barcelona, Spain, 2010.
- [4] Andreas Arzt, Gerhard Widmer, and Simon Dixon. Adaptive distance normalization for real-time music tracking. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 2689–2693, Bucharest, Romania, 2012.
- [5] Arshia Cont. A coupled duration-focused architecture for real-time music-to-score alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):974–987, 2010.
- [6] Simon Dixon and Gerhard Widmer. MATCH: A music alignment tool chest. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 492–497, London, UK, 2005.
- [7] Zhiyao Duan and Bryan Pardo. A state space model for online polyphonic audio-score alignment. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 197–200, Prague, Czech Republic, 2011.
- [8] Daniel P.W. Ellis and Graham E. Poliner. Identifying ‘cover songs’ with chroma features and dynamic programming beat tracking. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages 1429–1432, Honolulu, Hawaii, USA, 2007.
- [9] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.
- [10] Christian Fremerey, Frank Kurth, Meinard Müller, and Michael Clausen. A demonstration of the SyncPlayer system. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 131–132, Vienna, Austria, September 2007.
- [11] Werner Goebel. The vienna 4x22 piano corpus, 1999. <http://dx.doi.org/10.21939/4X22>.
- [12] Maarten Grachten, Carlos Eduardo Cancino Chacón, Thassilo Gadermaier, and Gerhard Widmer. Towards computer-assisted understanding of expressive dynamics in symphonic music. *IEEE Multimedia*, 24(1):36–46, 2017.
- [13] Maarten Grachten, Martin Gasser, Andreas Arzt, and Gerhard Widmer. Automatic alignment of music performances with structural differences. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 607–612, Curitiba, Brazil, 2013.
- [14] Ning Hu, Roger B. Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, 2003.
- [15] Cyril Joder, Slim Essid, and Gaël Richard. A comparative study of tonal acoustic features for a symbolic level music-to-score alignment. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, Texas, USA, 2010.
- [16] Cyril Joder, Slim Essid, and Gaël Richard. A conditional random field framework for robust and scalable audio-to-score matching. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(8):2385–2397, 2011.
- [17] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. Learning transposition-invariant interval features from symbolic music and audio. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*.
- [18] Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng. Sparse deep belief net model for visual area V2. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 873–880, 2007.
- [19] Kjell Lemström and Mika Laitinen. Transposition and time-warp invariant geometric music retrieval algorithms. *2011 IEEE International Conference on Multimedia and Expo*, pages 1–6, 2011.
- [20] Akira Maezawa, Katsutoshi Itoyama, Kazuyoshi Yoshii, and Hiroshi G. Okuno. Bayesian audio alignment based on a unified generative model of music composition and performance. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 233–238, Taipei, Taiwan, 2014.
- [21] Matija Marolt. A mid-level representation for melody-based retrieval in audio collections. *IEEE Trans. Multimedia*, 10(8):1617–1625, 2008.

- [22] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, Bucharest, Romania, 2015.
- [23] Roland Memisevic. Gradient-based learning of higher-order image features. In *IEEE International Conference on Computer Vision (ICCV), 2011*, pages 1591–1598. IEEE, 2011.
- [24] Nicola Montecchio and Arshia Cont. A unified approach to real time audio-to-score and audio-to-audio alignment using sequential Montecarlo inference techniques. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 193–196, Prague, Czech Republic, 2011.
- [25] Meinard Müller. *Fundamentals of Music Processing*. Springer Verlag, 2015.
- [26] Meinard Müller and Michael Clausen. Transposition-invariant self-similarity matrices. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 47–50, Vienna, Austria, September 2007.
- [27] Eita Nakamura, Tomohiko Nakamura, Yasuyuki Saito, Nobutaka Ono, and Shigeki Sagayama. Outer-product hidden markov model and polyphonic midi score following. *Journal of New Music Research*, 43(2):183–201, 2014.
- [28] Nicola Orio and François Déchelle. Score following using spectral analysis and hidden markov models. In *Proceedings of the International Computer Music Conference (ICMC)*, Havana, Cuba, 2001.
- [29] Christopher Raphael. Automatic segmentation of acoustic musical signals using hidden Markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:360–370, 1999.
- [30] Christopher Raphael. A probabilistic expert system for automatic musical accompaniment. *Journal of Computational and Graphical Statistics*, 10(3):487–512, 2001.
- [31] Christopher Raphael. A hybrid graphical model for aligning polyphonic audio with musical scores. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 387–394, Barcelona, Spain, 2004.
- [32] Stan Salvador and Philip Chan. FastDTW: Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.
- [33] Sertan Şentürk, Andre Holzapfel, and Xavier Serra. Linking scores and audio recordings in makam music of turkey. *Journal of New Music Research*, 43(1):34–52, 2014.
- [34] Sertan Şentürk and Xavier Serra. Composition identification in Ottoman-Turkish makam music using transposition-invariant partial audio-score alignment. In *Proceedings of 13th Sound and Music Computing Conference (SMC 2016)*, pages 434–441, Hamburg, Germany, 2016.
- [35] Verena Thomas. *Music Synchronization, Audio Matching, Pattern Detection, and User Interfaces for a Digital Music Library System*. PhD thesis, University of Bonn, 2013.
- [36] Thomas C. Walters, David A. Ross, and Richard F. Lyon. The intervalgram: An audio feature for large-scale cover-song recognition. In *From Sounds to Music and Emotions - 9th International Symposium, CMMR 2012, London, UK, June 19-22, 2012, Revised Selected Papers*, pages 197–213, 2012.
- [37] Siying Wang, Sebastian Ewert, and Simon Dixon. Robust joint alignment of multiple versions of a piece of music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 83–88, Taipei, Taiwan, 2014.
- [38] Gerhard Widmer. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence*, 146(2):129–148, 2003.
- [39] Gerhard Widmer, Simon Dixon, Werner Goebel, Elias Pampalk, and Asmir Tobudic. In search of the Horowitz factor. *AI Magazine*, 24(3):111–130, 2003.

SEMI-SUPERVISED LYRICS AND SOLO-SINGING ALIGNMENT

Chitralekha Gupta^{1,2}

Rong Tong⁴

Haizhou Li³

Ye Wang^{1,2}

¹ NUS Graduate School for Integrative Sciences and Engineering, ² School of Computing,

³ Electrical and Computer Engineering Dept., National University of Singapore, Singapore

⁴ Alibaba Inc. Singapore R&D Center, Singapore

chitralekha@u.nus.edu, rong.tong@alibaba-inc.com, haizhou.li@nus.edu.sg,
wangye@comp.nus.edu.sg

ABSTRACT

We propose a semi-supervised algorithm to align lyrics to the corresponding singing vocals. The proposed method transcribes and aligns lyrics to solo-singing vocals using the imperfect transcripts from an automatic speech recognition (ASR) system and the published lyrics. The ASR provides time alignment between vocals and hypothesized lyrical content, while the non-aligned published lyrics correct the hypothesized lyrical content. The effectiveness of the proposed method is validated through three experiments. First, a human listening test shows that 73.32% of our automatically aligned sentence-level transcriptions are correct. Second, the automatically aligned sung segments are used for singing acoustic model adaptation, which reduces the word error rate (WER) of automatic transcription of sung lyrics from 72.08% to 37.15% in an open test. Third, another iteration of decoding and model adaptation increases the amount of reliably decoded segments from 44.40% to 91.96% and further reduces the WER to 36.32%. The proposed framework offers an automatic way to generate reliable alignments between lyrics and solo-singing. A large-scale solo-singing and lyrics aligned corpus can be derived with the proposed method, which will be beneficial for music and singing voice related research.

1. INTRODUCTION

Lyrics serve as an important component of music, that often defines the mood of the song [2, 4], affects the opinion of a listener about the song [3], and even improves the vocabulary and pronunciation of a foreign language learner [14, 30]. Research in Music Information Retrieval (MIR) in the past has explored tasks involving lyrics such as automatic lyrics recognition [15, 19, 26, 28] and automatic lyrics alignment [5, 11, 27] for various applications such as karaoke singing, song subtitling, query-by-singing as well as acoustic modeling for singing voice. In spite of

huge advances in speech technology, automatic lyrics transcription and alignment in singing face challenges due to the differences between sung and spoken voices [11, 26], and a lack of transcribed singing data to train phonetic models for singing [11, 15, 26–28].

As singing and speech differ in many ways such as pitch dynamics, duration of phonemes, and vibrato [11, 26], the direct use of ASR systems for lyrics alignment or transcription of singing voice will result in erroneous output. Therefore, speech acoustic models need to be adapted to singing voice [27]. For training singing-adapted acoustic models, lyrics-aligned singing dataset is necessary. Lack of annotated singing datasets has been a bottleneck for research in this field. Duan et al. [8] published a small singing dataset (1.92 hours) with phone-level annotations, which were done manually that requires a lot of time and effort, and is not scalable. One way of getting data for training is to force-align the lyrics with singing using speech models, and use this aligned singing data for model training and adaptation. But due to the differences in speech and singing acoustic characteristics, alignment of lyrics with speech acoustic models will be prone to errors, that will result in badly adapted singing acoustic models.

With the increase in popularity of mobile phone karaoke applications, singing data collected from such apps are being made available for research. Smule’s *Sing!* karaoke dataset, called Digital Archive of Mobile Performances (DAMP) [33], is one such dataset that contains more than 34K a capella (solo) singing recordings of 301 songs. But it does not have time-aligned lyrics, although the textual lyrics are available on Smule’s website. The data also contains inconsistencies in recording conditions, out-of-vocabulary words, and incorrectly pronounced words because of unfamiliar lyrics or non-native language speakers. Although the presence of such datasets is a huge boon to MIR research, we need tools to further clean up such data to make them more usable. There is a need for aligned lyrics transcriptions for singing vocals while also eliminating inconsistent or noisy recordings. To address this need, we propose a simple yet effective solution to produce clean audio segments with aligned transcriptions.

In this work, we study a strategy to obtain time-aligned sung-lyrics dataset with the help of the state-of-the-art ASR as well as an external resource, i.e. published lyrics.



© Chitralekha Gupta, Rong Tong, Haizhou, Ye Wang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Chitralekha Gupta, Rong Tong, Haizhou, Ye Wang. “Semi-supervised lyrics and solo-singing alignment”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

We use the speech acoustic models to transcribe solo-singing audio segments, and then align this imperfect transcription with the published lyrics of the song to obtain a better transcription of the sung segments. We hypothesize that this strategy will help in correcting the imperfect transcriptions from the ASR module and in cleaning up bad audio recordings. We validate our hypothesis by a human listening experiment. Moreover we show that a semi-supervised adaptation of speech acoustic models with this cleaned-up annotated dataset results in further improvement in alignment as well as transcription, iteratively. Hence, such an algorithm will potentially automate the labor-intensive process of time aligning lyrics such as in karaoke or MTV. Furthermore, it will enable large-scale singing transcription generation, thus increasing the scope of research in music information retrieval. We have applied our algorithm on a subset of the DAMP dataset, and have published the resulting dataset and code¹.

2. RELATED WORK

One of the traditional methods of aligning lyrics to music is with the help of the timing information from the musical structure such as chords [17, 24, 25, 35], and chorus [21], but such methods are more suitable for singing in the presence of background accompaniments. Another study uses musical score to align lyrics [13], but such methods would be applicable for professional singing where the notes are correctly sung. In karaoke applications, as addressed in this work, correctness of notes is less likely.

One of the pioneering studies of applying speech recognition for lyric alignment was by Mesaros and Virtanen [27], who used 49 fragments of songs, 20-30 seconds long, along with their manually acquired transcriptions to adapt Gaussian Mixture Model-Hidden Markov Model (GMM-HMM) speech models for singing in the same way as speaker adaptation is done. They then used these singing-adapted speech models to align vocal sections of songs with their manually paired lyrics lines using the Viterbi algorithm. In [28], the authors used the same alignment method to automatically obtain the singing-to-lyrics aligned lines, and then explored multiple model adaptation techniques, to report the best phoneme error rate (PER) of 80%. This work has provided a direction for solving the problem of lyrics alignment and recognition in singing, but it suffers from manual post-processing and the models are based on a small number of annotated singing samples.

Recently, with the availability of more singing data, a subset of the DAMP solo-singing dataset was used for the task of sung phoneme recognition by Kruspe [19, 20]. In this work, the author builds new phonetic models trained only on singing data (DAMP data subset) and compares it with a pitch-shifted, time-stretched, and vibrato-applied version of a speech dataset called *songified* speech data TimitM [18]. Their best reported PER was 80%, and weighted PER (that gives 0.5 weights to deletions and in-

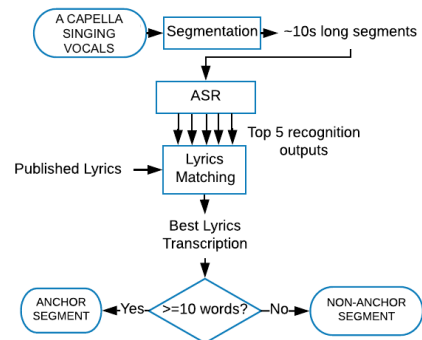


Figure 1: The diagram of lyrics to singing vocal alignment algorithm.

sertions) was 56%, using the DAMP data subset, which outperformed the songified dataset. This work shows an effective use of the available (unannotated) singing data to build improved singing phonetic models. But there is still room for improvement.

The first step in Kruspe’s work was to obtain aligned lyrics annotations of every song, for which the whole lyrics of a song was force-aligned with the audio using speech-trained models. These force-aligned sung phonemes were then used to build the new acoustic phonetic models for singing. This approach of forced-alignment of singing using speech acoustic models has also been applied in the earlier attempts of automatic lyrics alignment in a capella singing as well as in singing with background music [11, 16, 17, 35]. But, as noted by Kruspe [19], forced-alignment of singing with speech models causes unavoidable errors, because of the mismatch between speech and singing acoustic characteristics [10, 23], as well as the mismatch between the actual lyrics and what the singer sings. Thus, the lack of appropriate lyrics-aligned song dataset and the eventual use of forced-alignment with speech models to obtain this annotation is a source of errors.

3. SEMI-SUPERVISED LYRICS AND SINGING VOCALS ALIGNMENT ALGORITHM

We propose an algorithm to align lyrics to singing vocals, that consists of two main steps: dividing the singing vocals into shorter segments (Segmentation), and obtaining the aligned lyrics for each segment (Lyrics Matching). Figure 1 shows the overview of our algorithm.

3.1 Segmentation

One way to automatically align the published lyrics with a solo-singing audio is to force-align the lyrics with the full rendition audio (2 to 4 minutes long) using speech trained acoustic models, as discussed in [19]. However, the Viterbi alignment algorithm used in forced-alignment, fails to scale well for long audio segments leading to accumulated alignment errors [29]. In our singing-lyrics transcription and alignment algorithm, we propose to first divide the audio into shorter segments such that the ASR is less prone to the alignment errors. We find silent regions in the rendition by imposing constraints on the magnitude of the short time energy and the silence duration (Algorithm 1). The center of these silent regions are marked

¹ Dataset: <https://drive.google.com/open?id=1hGuE0Drv3tbN-YNRDzJJMHfzKH6e4O2A>;
Code: https://github.com/chitralekha18/AutomaticSungLyricsAnnotation_ISMIR2018.git

as boundaries of non-silent sub-segments. Such non-silent sub-segments are of varying lengths. So we stitch consecutive sub-segments together to make segments of ~ 10 seconds duration. We also add silence samples before and after every such segment so that the ASR has some time to adapt to the utterance and start recognition in the beginning of the utterance, and to avoid abrupt termination at the end of the utterance.

Algorithm 1 Segmentation algorithm

```

1: Calculate short time energy  $E$  for 32 ms window with 16 ms
   hop
2: if  $E > 0.1 \times \text{mean}(E)$  is true then
3:   non-silent region
4: else
5:   silent region
6: end if
7: if silent region duration  $\geq 200$  ms then
8:   valid silence region
9:   center of this region marks the boundary
10: else
11:   invalid silent region
12: end if
13: sub-segment = boundary-to-boundary region
14: segment = stitch together such sub-segments for  $\sim 10$ s duration
15: add 2s silence before and after every segment, to improve
    ASR performance
  
```

3.2 Lyrics Matching

We would like to obtain the best possible lyrics transcription for these short singing segments. Moreover, to obtain a clean transcribed dataset of singing vocals, we would also like to reject the noisy audio segments that contain out-of-vocabulary, incorrectly pronounced words, and background noise. We use ASR to decode these segments because such ASR transcription ideally suggests words that are actually sung and different from the published lyrics. The ASR transcription also help detect erroneous pronunciations, reject noise segments. We understand that the the state-of-the-art ASR is not perfect, and for singing it is even more unreliable, as the ASR is trained on speech while singing is acoustically different from speech. So we designed an algorithm to overcome these imperfections of the ASR. This algorithm produces time-aligned transcriptions of clean audio segments with the help of the published lyrics.

Algorithm 2 Lyrics Matching algorithm

```

1:  $X_{N \times 5}$  s.t.  $x_{i,j} = e$ 
   where,  $X$  = error matrix,
    $N$  = number of words in published lyrics,
    $e$  = ratio of number of errors obtained from Levenshtein distance
   between ASR output and published lyrics window, to the total
   number of words in the lyrics window
2:  $i_{min}, j_{min} = \text{argmin } X$ 
   where  $i_{min}$  = minimum distance transcription start index in
   lyrics,
   where  $j_{min}$  = minimum distance transcription slack window
   size
3: transcription = lyrics[ $i_{min} : i_{min} + M + j_{min}$ ]
   where,  $M$  is the number of words in ASR transcription
  
```

3.2.1 ASR Transcription of Lyrics

To obtain the transcription of each of the audio segments, we use the Google speech-to-text API package in python [36] that transcribes a given audio segment into a string of words, and gives a set of best possible transcriptions. We compare the top five of these transcriptions with the published lyrics of the song, and select the one that matches the most, as described in Algorithm 2. The idea is that the ASR provides a hypothesis of the aligned lyrics although imperfect, and the published lyrics helps in checking these hypothesized lyrics, and retrieving the correct lyrics. Also, we use the Google ASR to bootstrap, with a plan to improve our own ASR (as discussed further in Section 4.2). Different ASR systems have different error patterns, therefore we expect that the Google ASR would boost the performance of our ASR. We use the Google ASR only for bootstrapping, the rest of the experiments use our own ASR. Below is the description of the lyrics-matching algorithm.

For an ASR output of length M words, we took a lyrics window of size M , and also empirically decided to provide a slack of 0 to 4 words, i.e. the lyrics window size could be of length M to $M+4$. This slack provides room for accommodating insertions and deletions in the ASR output, thus allowing improvement in the alignment. So, starting from the first word of the published lyrics, we calculate the Levenshtein distance [22] between the ASR output and the lyrics window of different slack sizes, iterated through the entire lyrics by one word shifts. This distance represents the number of errors (substitutions, deletions, insertions) occurred in ASR output with respect to the actual lyrics.

For the lyrics of a song containing a total of N words, we obtain an error matrix X of dimensions $N \times 5$, where 5 is the number of slack lyric window sizes ranging from M to $M+4$. Each element e of the matrix is the ratio of the number of errors obtained from Levenshtein distance between the ASR output and the lyrics window, to the total number of words in that lyrics window. If (i_{min}, j_{min}) is the coordinate of the minimum error element of this matrix, then i_{min} is the starting index of the minimum distance lyrics transcription, j_{min} is the slack lyric window size. Amongst the top five ASR outputs, we choose the one that gives minimum error e , and select the corresponding lyrics window from the error matrix to obtain the best lyrics transcription for that audio segment. We illustrate this with the help of the following example.

Let's assume that the ASR transcription of an audio segment is "*the snow glows on the mountain*", therefore $M=6$. The slack window size will range from 6 to 10 words. The lyrics of this song contains a total of N words, where a word sub-sequence is "*the snow glows white on the mountain tonight not a footprint to be seen...*". The corresponding error matrix X is shown in Figure 2. The error element $e_{1,2}$ is the distance between the ASR transcription and the slack lyric window "*the snow glows white on the mountain*" which is 1. The error element $e_{2,1}$ is the distance between the ASR transcription and the slack lyric window "*snow glows white on the mountain*" which is 2, and so on.

$e_{i_{min}, j_{min}}$	1 (M=6)	2 (M=7)	3 (M=8)	4 (M=9)	5 (M=10)
Word 1	2	1	2	3	4
Word 2	2	3	4	5	6
....
Word N

Figure 2: Example of an error matrix X where the ASR transcript is “the snow glows on the mountain”, and the published lyrics of this song has N words where a word sub-sequence is “the snow glows white on the mountain tonight not a footprint to be seen...”.

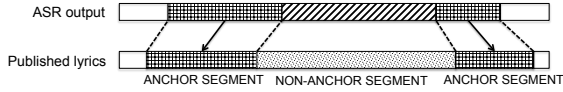


Figure 3: Anchor and non-anchor segments of a song based on sung-lyrics alignment algorithm. Anchor segments: ASR output and lyrics reliably match; Non-Anchor segments: ASR output and lyrics do not match.

So in this example, (i_{min}, j_{min}) is (1,2), i.e. the best lyrics transcription is “the snow glows white on the mountain”.

3.2.2 Anchor and Non-Anchor Segments

From our preliminary study, we found that many of the ASR transcriptions had missing words because either the audio contained background noise or there were incorrectly pronounced words or deviation of singing acoustics from speech. For example, a 10 seconds long non-silent segment from a popular English song would rarely ever have as few as four or five words. In order to retrieve more reliable transcripts, we added a constraint on the number of words, as described below.

To check the reliability of the lyrics transcriptions, we marked the best lyrics transcriptions of a small subset of 360 singing segments as correct or incorrect, depending on whether the transcription matched with the audio. We found that all those segments for which the best lyrics transcription had less than 10 words were more likely to be incorrect matches, as shown in Figure 4. The segment tran-

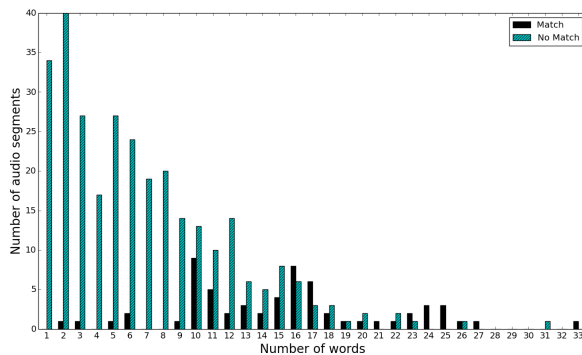


Figure 4: The number of audio segments with correct transcription (blue) or incorrect transcription (cyan) according to human judgment on y-axis versus the number of words in the transcription of an audio segment on x-axis. We set 10 words as the minimum threshold for a transcription to be valid for an approximately 10-seconds long segment.

scriptions were 94.0% times incorrect (235 incorrect out of 250 total number of segments) when they contained less than 10 words, while they were 57.3% times incorrect (63 out of 110) when they contained more than or equal to 10 words. So we empirically set 10 words as the threshold for selecting reliable audio segments and transcriptions. By applying this constraint, we reject those audio segments that are noisy, or have wrongly pronounced words, or cause errors in transcription because of model mismatch, thus deriving a clean transcribed singing dataset.

The audio segments with reliable transcription are labeled as *Anchor segments*, and the audio segment(s) between two anchor segments that have unreliable transcription, are strung together and labeled as *Non-Anchor segments*, as illustrated in Figure 3.

One may argue that we could have used the error score e to evaluate the reliability of a segment. However, if the ASR output itself is wrong, then this lyrics-matching error score will be misleading. For example, if only 4 words get detected by the ASR, out of 12 words in the audio segment, and all the 4 words are correct according to the published lyrics, then e will be zero for this transcription, which is incorrect, and also undetectable. Thus we set a threshold on the number of detected words (i.e. 10 words) as a way to measure the reliability of the segment and its transcription.

4. EXPERIMENTS AND RESULTS

In order to validate our hypothesis that our algorithm can retrieve good quality aligned transcriptions, we conducted three experiments: A) Human verification of the quality of the aligned transcriptions through a listening test, B) Semi-supervised adaptation of speech models to singing using our aligned sung-lyrics transcriptions for assessing the performance of automatic lyrics recognition, and C) Second iteration of alignment, and re-training of acoustic models, to check for further improvement in lyrics recognition.

Our experiments are conducted on 6,000 audio recordings from the DAMP dataset that was used by Kruspe [19]. The list of recordings used by Kruspe is here [1], however the training and test subsets are not clearly marked. Therefore we have defined our training and test datasets, and they are subsets of Kruspe’s dataset, as discussed in the following subsections. This data set contains recordings of amateur singing of English language pop songs with no background music but different recording conditions, which were obtained from the Smule Sing! karaoke app. Each performance is labeled with metadata such as the gender of the singer, the region of origin, the song title, etc. We obtained the textual lyrics of the songs from Smule Sing! website [34]. Since the songs in DAMP dataset were sung on Smule Sing! Karaoke app that uses these lyrics, it is safe to assume that these were the intended lyrics.

4.1 Experiment 1: Human Verification of the Quality of the Aligned-Transcriptions

In this experiment, we evaluate the quality of our aligned transcriptions (*segment transcriptions*), by asking participants to listen to the audio segments and verify if the given

transcription for the segment is correct or not. As opposed to word intelligibility evaluation tasks [6] where participants are asked to transcribe after listening to the stimuli once, in this task the participants were provided with the transcription and were free to listen to the audio as many times as they needed. Also the songs were popular English songs, that are less prone to perception errors [7].

4.1.1 Dataset

By applying our lyrics transcription and alignment algorithm (see Section 3), we obtained 19,873 of anchor segments (~ 58 hours) each ~ 10 seconds long, out of which we asked humans to validate 5,400 (15 hours) anchor segment transcriptions through a listening test. The only criterion to qualify for the listening test was to be proficient in English language. 15 university graduate students were the human listeners. Every listener was given one hour of audio segments containing 360 anchor segments along with the obtained lyrics transcription. The task was to listen to each of the audio segments and compare the given transcription with the audio. If at least 90% of the words in the transcription match with that in the audio, then the audio segment was marked as correctly transcribed. If not, then it was marked as incorrectly transcribed.

Similarly, we also tested the quality of the non-anchor segments. Non-anchor segments could be of varying durations, greater than or equal to 10 seconds. We conducted the same human validation task for 2 hours (1,262 segments) of the non-anchor segments of different durations.

4.1.2 Results and Discussion

There were two types of successful segment transcriptions, one was verified by humans as correct and also matched perfectly with the ASR output, and was labeled as *correct transcriptions fully matching ASR*. Another was verified as correct by humans but did not match with the ASR output due to ASR errors, but our algorithm could successfully retrieve the correct transcriptions, that we call *correct transcriptions partially matching ASR*. And the ones that were verified as wrong by humans are labeled as *error transcriptions due to imperfect ASR or incorrect singing*.

Anchor Segments: Table 1 shows the validation results for the anchor segments. We found that a total of 73.32% of the segments were transcribed correctly, where 57.80% of the segments were *partially matching ASR*. This means that our algorithm could successfully retrieve many incorrect ASR transcriptions, which validates our hypothesis that the extra information provided by the published lyrics coupled with ASR decoding produces good aligned transcriptions. We also found that incorrect singing of lyrics and imperfect ASR output resulted in 26.68% erroneous transcriptions. A common error reported by the listeners was many missing words at the trailing end of the incorrectly aligned transcriptions, although the correct words were clearly audible, which is possibly a result of model mismatch between singing and speech.

Non-Anchor Segments: From the human validation of the non-anchor segments, we find that 62.07% of the total of 1,262 non-anchor segments transcriptions are correct. This

Segment Transcriptions	#	%	Total %
Correct transcriptions fully matching ASR	838	15.52	73.32
Correct transcriptions partially matching ASR	3,121	57.80	
Error transcriptions due to imperfect ASR or incorrect singing	1,441	26.68	26.68

Table 1: A summary of correct and error transcriptions by the proposed algorithm. Google ASR is used for singing transcription. Total # anchor segments = 5,400 (15 hours).

suggests that these segments are relatively less reliable. Moreover, these audio segments could be long in duration (even more than a minute) that would cause errors in the Viterbi alignments. Thus in the subsequent experiments, we only use the anchor segments.

4.2 Experiment 2: Lyrics Transcription with Singing-Adapted Acoustic Models

In this experiment, we use our automatically generated aligned-transcriptions of sung audio segments in a semi-supervised adaptation of the speech models for singing. We use these singing-adapted models in an open test to validate our hypothesis that better aligned transcriptions for training singing acoustic models will result in improvement in automatic lyrics recognition compared to the best known baseline from the literature.

Adaptation of speech models for singing was previously attempted by Mesaros et al. [27, 28] who applied the speaker adaptation techniques to transform speech recognizer to singing voice. To reduce the mismatch between singing and speech, they used constrained maximum likelihood linear regression (CMLLR) to compute a set of transformations to shift the GMM means and variances of the speech models so that the resulting models are more likely to generate the adaptation singing data. In our work, we use CMLLR (also known as feature-space maximum likelihood linear regression (fMLLR)) [32] and our lyrics-aligned anchor segments to compute transformations for a semi-supervised adaptation of the speech models to singing. Adaptation can be done with the test dataset only, or the adaptation transformations can be applied at the time of training, called speaker adaptive training (SAT). Literature shows that the use of SAT with fMLLR transform requires minimum alteration to the standard code for training [12], and thus is a popular tool for speaker adaptation that we have used for singing adaptation here.

4.2.1 Dataset

The singing train set consists of 18,176 singing anchor segments from 2,395 singers while the singing test set consists of 1,697 singing anchor segments of 331 singers. The training set consists of both human verified and non-verified anchor segments, while the test set consists of only those anchor segment transcriptions that are verified as correct by humans. All of these anchor segments (training and test) are of ~ 10 seconds duration. There is no speaker overlap between the acoustic model training and test sets. A language model is obtained by interpolating a speech language model trained from Librispeech [31] text and a

Models Adapted by Singing Data	%WER	%PER
(1) Baseline (speech acoustic models)	72.08	57.52
(2) Adapted with test data	47.42	39.34
(3) Adapted (SAT) with training data	40.25	33.18
(4) Adapted (SAT+DNN) with training data	37.15	31.20
(5) Repeat (3) and (4) for 2nd round	36.32	28.49

Table 2: The sung word and phone error rate (WER and PER) in the lyrics recognition experiments with the speech acoustic models (baseline) and the singing-adapted acoustic models, on 1,697 correctly transcribed test singing anchor segments.

lyric language model trained from lyrics of the 301 songs of the DAMP dataset. The same language model is used in all the recognition experiments.

4.2.2 Results and Discussion

Table 2 reports the automatic lyrics recognition results on the singing test set using different acoustic models to observe the effect of adapting speech models for singing using our sung segments with aligned transcriptions.

The baseline speech acoustic model is a tri-phone HMM model trained on Librispeech corpus using MFCC features. Due to the mismatch between speech and singing acoustic characteristics, the WER and PER are high (Table 2 (1)). Adapting the baseline model with the singing test data results in a significant improvement in the error rates (Table 2 (2)). Speaker adaptive training (SAT) further improves the recognition accuracy (Table 2 (3)). A DNN model [9] is trained on top of the SAT model with the same set of training data. During DNN training, temporal splicing is applied on each frame with left and right context window of 4. The SAT+DNN model has 3 hidden layers and 2,976 output targets. With DNN training, the WER is reduced by about 7.7% relative to the SAT model (Table 2 (4)) and PER is 31.20%.

Mesaros et al. [27] reported the best PER to be 80% with speech models adapted to singing, while Kruspe [19] reported the best PER to be 80% and weighted PER to be 56% with pure singing phonetic models trained on a subset of the DAMP dataset. Compared to [19] and [27], our results show a significant improvement, which is attributed to three factors. One is that leveraging on ASR along with the published lyrics as an external resource to validate and clean-up the transcriptions has led to better aligned transcriptions for training. Two, our automatic method for generating aligned transcriptions for singing provides us with a much larger training dataset. And three, the segment-wise alignment is more accurate than the whole-song forced-aligned with the speech acoustic models.

4.3 Experiment 3: Alignment with Singing-Adapted Acoustic Models and Re-training

We would like to test if the singing-adapted acoustic models can provide more number of reliably aligned transcriptions in a second round of alignment. Moreover whether re-training the models with this second round of transcriptions lead to better lyrics recognition.

Model	# anchor	total # segments	% anchor
Google ASR	5,400	12,162	44.40
Adapted (DNN) with training data	11,184	12,162	91.96

Table 3: Comparing the number of anchor segments obtained from the proposed transcription and alignment algorithm using Google ASR and the singing-adapted models.

4.3.1 Dataset

We used the singing-adapted models obtained in Experiment 2 to decode 12,162 segments, and then applied our lyrics-alignment algorithm to obtain new anchor and non-anchor segments. For comparison, we obtained the same from the Google ASR on the same dataset.

4.3.2 Results and Discussion

Table 3 shows that the number of anchor segments with the new models have increased from 44.40% with Google ASR to 91.96% with the singing-adapted models, which means that the number of reliable segment transcriptions have increased significantly. With the new anchor segments, we re-train our singing-adapted acoustic models. Table 2 (5) shows the free-decoding results after this second round of training. The WER and PER have dropped further to 36.32% and 28.49% respectively.

The results of this experiment are promising as they show iterative improvement in the quality of our alignment and transcription. This means that we can apply the following strategy: use only the reliably aligned segments from the Google ASR to adapt acoustic models for singing, and use these models to improve the quality of alignment and transcription, and then again use the reliable segments from the improved alignments for further adaptations.

5. CONCLUSIONS

We propose an algorithm to automatically obtain time-aligned transcriptions for singing by using the imperfect transcriptions from the state-of-the-art ASR along with the non-aligned published lyrics. Through a human listening test, we showed that the extra information provided by the published lyrics helps to correct many incorrect ASR transcriptions. Furthermore, using the time-aligned lyrics transcriptions for iterative semi-supervised adaptation of speech acoustic models for singing shows significant improvement in automatic lyrics transcription performance. Thus our strategy to obtain time-aligned transcriptions for large-scale singing dataset is useful to train improved acoustic models for singing.

Our contribution provides an automatic way to obtain reliable lyrics transcription for singing, that results in an annotated singing dataset. Lack of such datasets has been a bottleneck in the field of singing voice research in MIR. This will not only generate lyrics transcription and alignment for karaoke and subtitling applications, but also provide reliable data to improve acoustic models for singing, thus widening the scope of research in MIR.

6. REFERENCES

- [1] MIREX 2017. 2017 Automatic Lyrics-to-Audio Alignment. http://www.music-ir.org/mirex/wiki/2017:Automatic_Lyrics-to-Audio_Alignment. [Online; accessed 15-March-2018].
- [2] S Omar Ali and Zehra F Peynircioglu. Songs and emotions: are lyrics and melodies equal partners? *Psychology of Music*, 34(4):511–534, 2006.
- [3] Bruce Anderson, David G Berger, R Serge Denisoff, K Peter Etzkorn, and Peter Hesbacher. Love negative lyrics: Some shifts in stature and alterations in song. *Communications*, 7(1):3–20, 1981.
- [4] Elvira Brattico, Vinoo Alluri, Brigitte Bogert, Thomas Jacobsen, Nuutti Vartiainen, Sirke Nieminen, and Mari Tervaniemi. A functional mri study of happy and sad emotions in music with and without lyrics. *Frontiers in psychology*, 2, 2011.
- [5] Yu-Ren Chien, Hsin-Min Wang, and Shyh-Kang Jeng. Alignment of lyrics with accompanied singing audio based on acoustic-phonetic vowel likelihood modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):1998–2008, 2016.
- [6] Lauren B Collister and David Huron. Comparison of word intelligibility in spoken and sung phrases. 2008.
- [7] Nathaniel Condit-Schultz and David Huron. Catching the lyrics: intelligibility in twelve song genres. *Music Perception: An Interdisciplinary Journal*, 32(5):470–483, 2015.
- [8] Zhiyan Duan, Haotian Fang, Bo Li, Khe Chai Sim, and Ye Wang. The nus sung and spoken lyrics corpus: A quantitative comparison of singing and speech. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*, pages 1–9. IEEE, 2013.
- [9] G. Hinton et al. Deep neural networks for acoustic modeling in speech recognition. In *IEEE Signal Processing Magazine*, volume 29, pages 82–97, 2012.
- [10] Hiromasa Fujihara and Masataka Goto. Lyrics-to-audio alignment and its application. In *Dagstuhl Follow-Ups*, volume 3. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.
- [11] Hiromasa Fujihara, Masataka Goto, Jun Ogata, and Hiroshi G Okuno. Lyricsynchronizer: Automatic synchronization system between musical audio signals and lyrics. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1252–1261, 2011.
- [12] Mark JF Gales. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998.
- [13] Rong Gong, Philippe Cuvillier, Nicolas Obin, and Arshia Cont. Real-time audio-to-score alignment of singing voice based on melody and lyric information. In *Interspeech*, 2015.
- [14] Arla J Good, Frank A Russo, and Jennifer Sullivan. The efficacy of singing in foreign-language learning. *Psychology of Music*, 43(5):627–640, 2015.
- [15] Jens Kofod Hansen and IDMT Fraunhofer. Recognition of phonemes in a-cappella recordings using temporal patterns and mel frequency cepstral coefficients. In *9th Sound and Music Computing Conference (SMC)*, pages 494–499, 2012.
- [16] Denny Iskandar, Ye Wang, Min-Yen Kan, and Haizhou Li. Syllabic level automatic synchronization of music signals and text lyrics. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 659–662. ACM, 2006.
- [17] Min-Yen Kan, Ye Wang, Denny Iskandar, Tin Lay Nwe, and Arun Shenoy. Lyrically: Automatic synchronization of textual lyrics to acoustic music signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):338–349, 2008.
- [18] Anna M Kruspe. Training phoneme models for singing with “songified” speech data. In *ISMIR*, pages 336–342, 2015.
- [19] Anna M Kruspe. Bootstrapping a system for phoneme recognition and keyword spotting in unaccompanied singing. In *ISMIR*, pages 358–364, 2016.
- [20] Anna M Kruspe. Retrieval of textual song lyrics from sung inputs. In *INTERSPEECH*, pages 2140–2144, 2016.
- [21] Kyogu Lee and Markus Cremer. Segmentation-based lyrics-audio alignment using dynamic programming. In *ISMIR*, pages 395–400, 2008.
- [22] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.
- [23] Alex Lescos, Pedro Cano, and Jordi Bonada. Low-delay singing voice alignment to text. In *ICMC*, 1999.
- [24] Matthias Mauch, Hiromasa Fujihara, and Masataka Goto. Lyrics-to-audio alignment and phrase-level segmentation using incomplete internet-style chord annotations. In *Proceedings of the 7th Sound and Music Computing Conference (SMC 2010)*, pages 9–16, 2010.
- [25] Matthias Mauch, Hiromasa Fujihara, and Masataka Goto. Integrating additional chord information into hmm-based lyrics-to-audio alignment. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):200–210, 2012.
- [26] Matt McVicar, Daniel PW Ellis, and Masataka Goto. Leveraging repetition for improved automatic lyric transcription in popular music. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3117–3121. IEEE, 2014.
- [27] Annamaria Mesaros and Tuomas Virtanen. Automatic alignment of music audio and lyrics. In *Proceedings of the 11th Int. Conference on Digital Audio Effects (DAFx-08)*, 2008.

- [28] Annamaria Mesaros and Tuomas Virtanen. Automatic recognition of lyrics in singing. *EURASIP Journal on Audio, Speech, and Music Processing*, 2010(1):546047, 2010.
- [29] Pedro J Moreno, Christopher F Joerg, Jean-Manuel Van Thong, and Oren Glickman. A recursive algorithm for the forced alignment of very long audio segments. In *ICSLP*, volume 98, pages 2711–2714, 1998.
- [30] Hitomi Nakata and Linda Shockey. The effect of singing on improving syllabic pronunciation–vowel epenthesis in japanese. In *International Conference of Phonetic Sciences*, 2011.
- [31] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *ICASSP*, 2015.
- [32] Daniel Povey and George Saon. Feature and model space speaker adaptation with full covariance gaussians. In *Ninth International Conference on Spoken Language Processing*, 2006.
- [33] Smule. Digital Archive Mobile Performances (DAMP). <https://ccrma.stanford.edu/damp/>. [Online; accessed 15-March-2018].
- [34] Smule. Digital Archive Mobile Performances (DAMP). <https://www.smule.com/songs>. [Online; accessed 15-March-2018].
- [35] Ye Wang, Min-Yen Kan, Tin Lay Nwe, Arun Shenoy, and Jun Yin. Lyrically: automatic synchronization of acoustic musical signals and textual lyrics. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 212–219. ACM, 2004.
- [36] A Zhang. Speech Recognition (Version 3.7) [Software]. https://github.com/Uberi/speech_recognition#readme, 2017. [Online; accessed 14-Oct-2017].

CONCERT STITCH: ORGANIZATION AND SYNCHRONIZATION OF CROWD-SOURCED RECORDINGS

Vinod Subramanian

Center for Music Technology
Georgia Institute of Technology
vsubramanian32@gatech.edu

Alexander Lerch

Center for Music Technology
Georgia Institute of Technology
alexander.lerch@gatech.edu

ABSTRACT

The number of audience recordings of concerts on the internet has exploded with the advent of smartphones. This paper proposes a method to organize and align these recordings in order to create one or more complete renderings of the concert. The process comprises two steps: first, using audio fingerprints to represent the recordings, identify overlapping segments, and compute an approximate alignment using a modified Dynamic Time Warping (DTW) algorithm and second, applying a cross-correlation around the approximate alignment points in order to improve the accuracy of the alignment. The proposed method is compared to two baseline systems using approaches previously proposed for similar tasks. One baseline cross-correlates the audio fingerprints directly without DTW. The second baseline replaces the audio fingerprints with pitch chroma in the DTW algorithm. A new dataset annotating real-world data obtained from the Live Music Archive is presented and used for evaluation of the three systems.

1. INTRODUCTION

Crowd-sourcing is the concept of presenting a problem to a large group of people and utilizing the best combination of the solutions received [12]. Although a large group of people can be used to obtain data, the data needs to be organized and labeled in a logical way to be useful. For instance, there has been an explosion in the number of audio and video recordings available online in the last few years. For large events such as concerts, speeches, and sports events, there are many recordings of (parts of) the same event. These recordings, however, are not annotated in a way that would allow a reconstruction of the complete timeline of the event. The focus of this research is, therefore, on the automatic organization and synchronization of the multiple recordings available of the same event.

Marshall and Shipman [16] analyze the people's reasons for recording events and report personal memorabilia, sharing on social platforms, creation of remixes, and online

republishing as the main reasons. This indicates that there is value attached to these recordings. Vihavainen et al. [24] showed in their work that a human-computer collaborative approach to remixing concerts is of interest to a concert audience. Although the subjects favored the manually edited concerts in this instance, it still emphasizes the value of recombining audience recordings

While recombining audience recordings creates a better audience experience beyond the concert, a tool for automatic concert "stitching", faces several challenges. For example, each recording will have different audio quality due to different recording devices, distance from the stage, local disturbances etc.

After meeting these challenges, the application of this research enables (a) improved audience experience through personalized, collaborative, or theme-driven reconstruction of the event thus creating a platform for derivative work, (b) analysis and improvement of stage setups by venues and performers through audience videos from a large variety of recording angles, and, more generally, (c) audio forensics to reconstruct a scene by synchronizing multiple recordings for surveillance and investigation.

The goal of this study is to present a method that can (a) reliably identify if multiple recordings from an event have common audio content and (b) provide a precise alignment between all pairs of recordings. In the hope of encouraging more research on this task, we also present a new dataset for training and evaluation.

2. RELATED WORK

The task of aligning multiple recordings of an event can be divided into two steps: first, using a representation the recordings to identify overlapping segments, and compute an approximate alignment and second, applying a cross-correlation around the approximate alignment points in order to improve the accuracy of the alignment.

In tasks such as speech recognition [6, 11] and music similarity [1, 8], Mel-Frequency Cepstral Coefficients (MFCCs) are widely used to measure similarity between audio files. The Mel-Cepstrum captures timbral information and the spectral shape of the audio file [3]. However, MFCCs do not contain musically meaningful information such as melody or rhythm which could be argued to be crucial for computing music similarity.

Music Information Retrieval tasks such as cover song



© Vinod Subramanian, Alexander Lerch. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: Vinod Subramanian, Alexander Lerch. "Concert Stitch: Organization and Synchronization of Crowd-Sourced Recordings", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

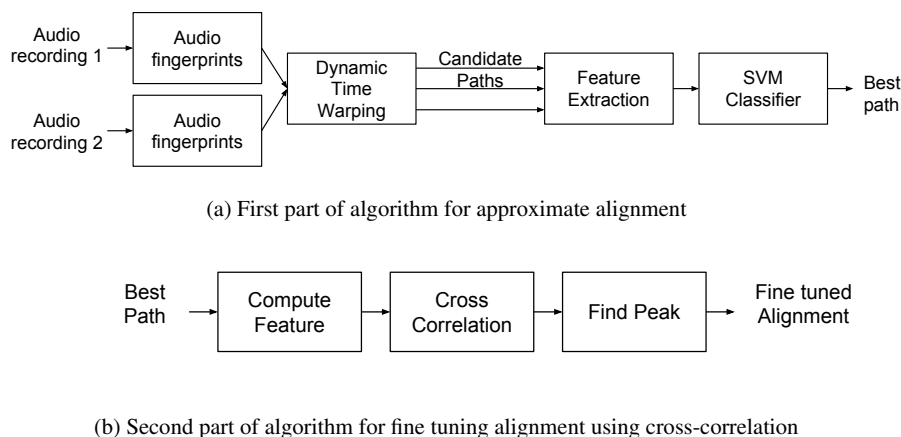


Figure 1: Proposed method block diagram

detection [19], audio thumbnailing [2], and genre classification [23] use pitch-based features such as a pitch chroma to compute a measure of similarity. The pitch chroma [15] is an octave-invariant representation of the tonal content of an audio file and is usually computed in intervals of approx. 10 ms. A useful property of the pitch chroma is its robustness to timbre variations, allowing it to compare the pitch content of two different versions of the same song without being strongly influenced by timbre variations.

Determining the similarity of two recordings is closely related to audio fingerprinting, which aims at identifying a recording from a large database of recordings. An audio fingerprint is a highly compressed and unique representation of a (part of a) song [10, 25]. Wang [25] introduced an audio fingerprinting technique based on so-called landmarks. A landmark is identified as the spatial relationship of the salient spectral peaks. This representation is also used for the task of audio alignment of concert recordings [4, 13]. Most audio fingerprinting methods are temporally sensitive, meaning that they are not designed to handle variations in playback speed — a scenario that is likely in the case of analog recordings of concerts. The audio fingerprinting method introduced by Haitsma and Kalker calculates a 32 bit sub-fingerprint for every block of audio by looking at the energy differences along the frequency and time axes. This fingerprint method is used by Shrestha et al. [21] in their work on alignment of concert recordings. Alternately, Wilmering et al. [26] use high-level audio features such as tempo and chords in combination with low-level audio features such as MFCCs and pitch chroma to detect audio similarity for audio alignment of different versions of concerts.

Identifying and aligning overlapping segments requires the computation of a similarity or distance measure across a sequence of signal descriptors. One way of doing this is cross-correlation. Most of the research in aligning concert recordings apply this approach [4, 5, 13, 21, 22]. One constraint of cross-correlation is that the two sequences are assumed to be at the same speed. It is apparent that cross-correlation cannot be easily applied to the task of aligning

analog recordings because there may be tempo variations and temporal fluctuations in the data. Another issue with cross-correlation is that a threshold needs to be set for what constitutes an alignment. To set the threshold some publications use heuristic methods based on their data [4, 5, 13, 22], while others [21] use a threshold determined by Haitsma and Kalker [10]. Using fixed thresholds bears the risk of errors when applying the system to unseen data.

Another method for computing overlaps is the use of Dynamic Time Warping (DTW), as it is able to handle temporal fluctuations between the signals [14]. Wilmering et al. apply DTW twice, the first time for aligning a recording to a reference audio file in order to identify the different playback speeds. Based on the result, the audio files are processed to mirror the playback speed of the reference. The second alignment is then applied to improve the accuracy of the first alignment. DTW is also used in the related task of sample detection, where it can help to identify the location of a sample in a song [9].

3. ALGORITHM DESCRIPTION

The first part of the algorithm, as shown in Figure 1a, computes audio fingerprints for each recording and uses these fingerprints to compute pairwise distance matrices. For each distance matrix, a DTW algorithm determines multiple possible path candidates representing the potentially overlapping region between that pair of recordings. For each of these candidates, features are extracted and an SVM classifier determines which path is the most likely. In the case that the pair is not overlapping, no path should be selected from the candidates. The second part of the algorithm as shown in Figure 1b takes the most likely path and computes a cross-correlation of the overlapping regions to determine the exact alignment of the pairs and to improve accuracy.

3.1 Audio Fingerprint Computation

The motivation for using audio fingerprints is that it is a representation of audio robust to noise and timbre [10,

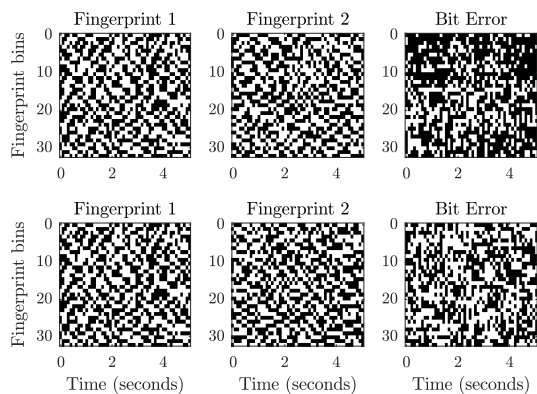


Figure 2: The top row shows the fingerprints from two recordings of the same 5 second snippet. The second row shows the fingerprints from two recordings of different 5 second snippets. For the Bit Error, the black regions indicate the fingerprints match and the white regions indicate the fingerprints are different.

25]. The audio fingerprinting technique utilized here is the Haitsma and Kalker algorithm [10]. The audio fingerprints are computed at a sampling rate of 5 kHz with a block size 2048 and a hop size of 512.

Figure 2 visualizes the robustness of audio fingerprints to noise distortion with an example. The upper row shows the bit error (in white) between the fingerprints of two matching but distorted recordings, the lower row shows the same for two different recordings. We can clearly see how the fingerprints retain the essential information even in the case of heavy distortion.

3.2 Modified Dynamic Time Warping

Dynamic Time Warping (DTW) is designed to align sequences with similar content but are temporally different. In the case of aligning concert recordings, the temporal fluctuations might occur due to inaccuracies in the sampling rate; in the case of analog recordings, the temporal fluctuations might be caused due to varying playback speeds.

The classical DTW algorithm introduced by Sakoe and Chiba works under the assumption that the start and end points of the two sequences are aligned [20]. A modification of the standard approach allows the algorithm to detect subsequences [18]; however, in the case of real life recordings, the most likely scenario is that a pair of recordings might have overlapping regions. Therefore, a pair of recordings will neither have the same start and end points, nor will one recording necessarily be a subsequence of the other. To address this issue, the subsequence DTW algorithm is modified to look for overlapping regions by doing the traceback from all possible end points.

The distance matrix is computed as the pairwise distance of two audio fingerprint matrices corresponding to two recordings. The dimension of one fingerprint matrix is $32 \times M$ and of the second is $32 \times N$ where M and N correspond to the number of blocks of audio that each recording was divided into. Using the Hamming distance, the result is

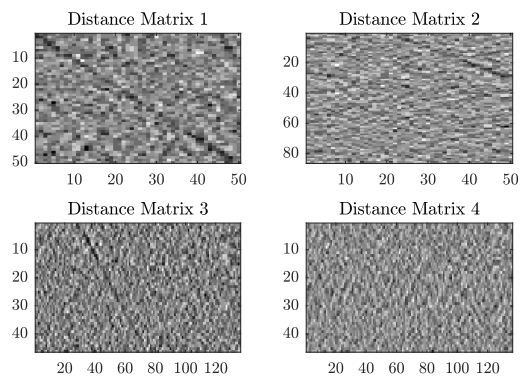


Figure 3: Distance matrix examples. The dark line indicates high similarity. For Distance matrix 4, there is no overlap, so there is no high similarity region

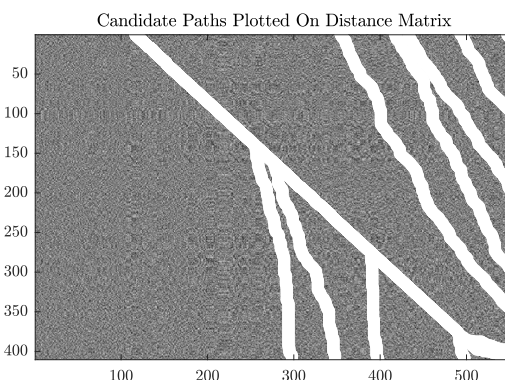


Figure 4: Different candidate path examples. The straightest line in the image represents the correct path.

a distance matrix D with the dimensions $M \times N$. Figure 3 shows examples of the distance matrix for different pairs of recordings; the top left matrix shows a standard DTW case with start and end points of both sequences aligned, the top right and bottom left are computed from pairs of recordings with overlapping regions and the bottom right matrix corresponds to a pair of recordings without overlapping regions.

A cost matrix is computed from the distance matrix as is done for the subsequence DTW algorithm [18]. In short, the initialization of the cost matrix computation is modified— as opposed to accumulating the distance across both the first row and first column, only the first column is accumulated.

We use the standard DTW technique to traceback the path; however, instead of doing this on just the minimum cost point, the traceback is performed on all possible path end points from the last row and last column. This results in multiple paths. Figure 4 illustrates a few paths that are computed for an example cost matrix.

3.3 Feature Extraction

To identify the most likely candidate path, we extract features from each path. Each possible path has three features:

(a) the DTW cost normalized by path length, (b) the slope of the line connecting the starting and ending points, and (c) the deviation of the path from the line connecting the start and end points. These paths are then clustered such that each cluster contains paths that share a start point; the end point for each cluster is the path with the lowest normalized cost. From each cluster, the minimum, mean, and standard deviation of the three path features are taken along with the number of paths in the cluster. These cluster features are similar to the ones proposed by Gururani and Lerch in the context of sample detection [9]. The extracted features per cluster have a dimensionality of 1×10 per cluster and are the input of a classifier estimating whether a path candidate represents a true overlap or not.

3.4 Classifier

A binary classifier is trained to determine which of the candidate paths is the most likely path for the alignment. A Support Vector Machine algorithm (SVM) with a linear kernel is used as this classifier. In the event that the classifier doesn't identify any of the candidate paths as a path for alignment, it is assumed that that pair of recordings do not have overlapping content. In the case of two or more paths being classified as true overlapping paths, the classifier's output probability is used to choose the most probable path.

3.5 Sample-Accurate Alignment

The audio fingerprinting technique used [10] downsamples the audio to 5000 Hz and blocks the audio by 1024 samples so the DTW alignment has a low resolution. As a more accurate result is desirable to reconstruct the timeline artifact-free (without 'jumps') when splicing two recordings together, a post-processing step is applied. One audio file is resampled based on the approximate alignment; then, the cross-correlation of overlapping regions of the pair of recordings is computed for 5 seconds around the detected start point. The result should then provide a synchronization point with improved accuracy.

3.6 Baseline

We compare the results of the proposed method to two baseline systems— one looking at the audio features and the other looking at the alignment stage.

3.6.1 Pitch chroma baseline

In order to investigate the effect of audio descriptors on the alignment accuracy, the pitch chroma is used as the audio representation instead of audio fingerprints. For the pitch chroma, a euclidean distance is used instead of the Hamming distance for calculating the distance matrix. Pitch chroma is a feature of interest as it is a typical feature used for audio similarity [2, 19, 23]. It has also been used in previous work on aligning concert recordings [26]. The pitch chroma is computed at a sampling rate of 11 kHz with a block size of 4096 and a hop size of 1024.

3.6.2 Cross-correlation baseline

The cross-correlation on audio fingerprints is the most established approach in the field of aligning noisy concert recordings [4, 13, 21]. For this process, the Hamming distance is computed at different levels of overlap and a threshold of 0.35 Bit Error Rate (BER) is set according to the recommendation by Haitsma and Kalker [10]. If the distance falls below the threshold then the pair of recordings are aligned at that overlap.

4. EXPERIMENTS

We run several experiments to investigate our algorithm. We evaluate the audio (feature) representation, approaches to alignment, and alignment accuracy.

4.1 Dataset

Two datasets are used in this study— a synthetic dataset and a real world dataset. The synthetic dataset created for simulating a real world scenario; the advantage is a sample-accurate ground truth. The synthetic dataset will be used as the training and validation set, as well as to provide some preliminary results with high accuracy. The real-world dataset is manually annotated from existing recordings and is used to test the overall performance of the algorithm.

4.1.1 Synthetic Dataset

The synthetic dataset is a collection of audio recordings downloaded from YouTube¹ consisting of live recordings of concerts. There are a 100 songs available in this dataset.

In order to create training data for the classifier, each song of the dataset is divided into 17 (can be varied) recordings with the constraint that each recording is longer than 20s and the entire song is covered. Each recording is modified by (a) resampling randomly between 42.9 kHz to 45.2 kHz, (b) either low pass filtering with a cutoff between 5000 Hz to 11600 Hz or high pass filtering with a cutoff between 200 Hz to 5000 Hz, (c) adding crowd sounds obtained from freesound.org [7], and (d) adding distortion using the 'live recording' and 'smart phone recording' simulations in the audio degradation toolbox [17]. The code for generating the synthetic dataset is available online².

4.1.2 Real World Dataset

The real world dataset consists of 5 audience recordings of a Grateful Dead concert performed on 1977-05-08. The audio data was obtained from the Live Music Archive³. The first 5 songs from the concert were selected and each of the 5 versions of the 5 songs were annotated. The annotations indicate the start and end points of the song. In case a part of the song is missing, the duration and location of the missing location is indicated. Since these recordings were made on analog devices, the data is prone to tempo and playback speed variation in addition to the usual filtering and distortion heard in audience recordings. The real world

¹ <https://www.youtube.com/> accessed March 1st 2018

² <https://github.com/VinodS7/ConcertStitch-dataset>

	precision	recall	f-measure
Fingerprints	0.9697	0.6732	0.8145
Pitch Chroma	0.6753	0.3191	0.4335

Table 1: Experiment 1: Overlap detection for audio fingerprints vs. pitch chroma on real world data

dataset is augmented by splitting each version of a song into 10 recordings, resulting in 50 simulated audience recordings per song. The songs are split in the same way as for the synthetic dataset.

4.2 Metrics

There are two metrics used for the evaluation of this task. The first metric is using the precision, recall, and f-measure to provide an understanding of whether an alignment is correctly detected for a pair of recordings. The second metric is the statistical analysis of the alignment accuracy in seconds where the median, standard deviation, and maximum values are used to measure how accurate the alignment is.

4.3 Experiment 1: Audio fingerprints vs. pitch chroma

The aim of this experiment is to compare the audio representation on which the distance computation is based. We investigate audio fingerprints and pitch chroma for the task of aligning noisy recordings.

To train the SVM classifier for the algorithm, the above-mentioned cluster features are extracted from the synthetic dataset for 25 songs. To extract the features for each pair of recordings, the DTW algorithm computes multiple possible paths corresponding to all unique starting points. All paths are labeled incorrect except the path that is closest to the ground truth in the case of overlapping recordings. The extracted feature matrix thus consists of the cluster features along with a label of whether those features correspond to an overlap or not. This process is applied to both audio fingerprints and pitch chromas.

Once the feature matrix is available, it is divided into an 80-20 split for training and validation, respectively. As each pair of recordings has multiple candidate paths with a maximum of only one being correct, there are far more negative observations in the feature matrix than positive observations. To counteract the high number of negative observations, the training data is sampled to reduce the number of negative observations. The ratio of negative to positive observations is 50:1 for the audio fingerprints classifier and 30:1 for the pitch chroma classifier.

The evaluation is performed on the real world dataset. Only the start points of the alignment are taken into account because the audio files are not modified or resampled based on the end points.

4.3.1 Results

Table 1 reports the precision, recall, and f-measure of the audio fingerprints and the pitch chroma. The fingerprint outperforms the pitch chroma considerably for all metrics. This

³ <https://archive.org/details/GratefulDead> accessed January 15th 2018

Real World	precision	recall	f-measure
DTW	0.9697	0.6732	0.8145
cross-correlation	0.4132	0.2534	0.3141
Synthetic	precision	recall	f-measure
DTW	0.9570	0.9319	0.9443
cross-correlation	0.6936	0.8956	0.7818

Table 2: Experiment 2: DTW vs. cross-correlation using audio fingerprints for real world and synthetic data

result is expected as the fingerprint is specifically designed to work in conditions with severe quality impairments. The poor performance of the pitch chroma can be traced back to computing the candidate paths in the DTW algorithm. Due to the noise, the candidate paths frequently do not contain the correct path for the pitch chroma. This adversely affects the training process for the SVM classifier and subsequently the performance on the real world data.

4.4 Experiment 2: DTW vs. cross-correlation

The aim of this experiment is to compare the performance of the DTW and the cross-correlation techniques when audio fingerprints are used as the audio representation. The audio fingerprints for the cross-correlation method are almost the same as for the DTW algorithm, the only difference is that the hop size is now 64 instead of 512.

The classifier for the DTW algorithm is set up the same way as in Experiment 1. The evaluation is performed on both the synthetic dataset with no temporal fluctuations and the real world dataset.

4.4.1 Results

Table 2 reports the precision, recall, and f-measure of the DTW method and the cross-correlation method on the two datasets. We observe that the DTW method clearly outperforms the cross-correlation method. This is especially true for the real-world data because the DTW is designed to handle temporal fluctuations while cross-correlation is not. On the synthetic dataset containing no temporal fluctuations, the cross-correlation method performs much better; however, it still does not perform as well as the DTW method. One possible reason might be that the cross-correlation method uses a strict threshold to identify alignment so the cross-correlation method does not scale well to different types of noise.

4.5 Experiment 3: DTW performance analysis

The goal of this experiment is to understand the strengths and weaknesses of the proposed algorithm.

For the first part of the experiment, the precision, recall, and f-measure are reported for difference tolerance thresholds on the real world dataset. The tolerance threshold gives maximum allowable deviation of the alignment provided by the algorithm from the ground truth. If the alignment exceeds the threshold then it means the algorithm predicted the alignment incorrectly.

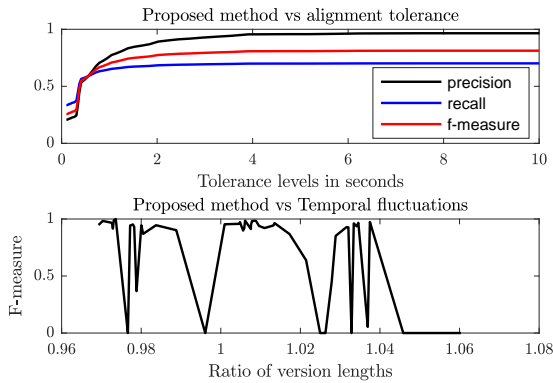


Figure 5: Experiment 3: Analyzing the performance of the proposed method

The second part of the experiment tests how robust to time stretching and pitch shifting the algorithm is. First, the sample rates for each of the recordings are identified using the same technique as Wilmering et al. [26]. Then, the alignment is calculated between each pair of recordings. Finally, the f-measure of the alignment is compared to the ratio of sample rates(or version lengths).

4.5.1 Results

The first part of Figure 5 shows the precision, recall, and f-measure at different tolerance thresholds. The plot shows that the performance decreases drastically for tolerances below 2 s. These results indicate a need to refine the alignment in order to provide a more accurate measure of alignment.

For the second part of Figure 5, we expect the algorithm to perform better if the ratio of lengths is closer to 1 and the performance to get worse the further away from 1. The reason is that if the ratio of sample rates is further away from one the pitch shifting becomes more significant which this algorithm is not designed to handle. However, the plot does not reflect this hypothesis because the pitch shifts in certain audio files is greater than expected. In addition to resampling there is more pitch shifting which causes the algorithm to fail since both the pitch chroma and fingerprints are sensitive to pitch shifting.

4.6 Experiment 4: Analysis of improved alignment accuracy

For a pair of recordings using the alignment, a resampling factor is calculated using a ratio of the length of the two paths. One recording is resampled so it has the same length as the other. We investigate and compare the spectral flux, spectral centroid, and time-domain raw audio for their ability to improve the alignment accuracy when cross-correlating a small segment around the previously estimated alignment points. For reference, the same features are computed without resampling the audio. The spectral flux and spectral centroid are calculated at a block size of 128 with a hop size of 32. The alignment accuracy for the raw audio, spectral flux, and spectral centroid for the original and resampled audio are compared against the original

	median	std	max
DTW alignment	5240	11503	125221
Raw audio	9043	49091	823906
Spectral Flux	7073	12554	108950
Spectral Centroid	7161	9919	54695
Res. Raw Audio	5801	23185	227631
Res. Spec. Flux	5078	13092	125846
Res. Spec. Centroid	5006	11128	100165

Table 3: Raw Audio vs Spectral Flux to improve alignment accuracy. The results are reported as deviation in samples at 44.1 kHz

DTW algorithm to evaluate the accuracy improvement. The evaluation for this task is done on the synthetic dataset because the annotations are more accurate than for the real world data.

4.6.1 Results

The results of Experiment 4 are reported in Table 3. The numbers indicate how close to the ground truth alignment the algorithm performs in samples at a sample rate of 44100 Hz. None of the finer alignment algorithms are able to significantly improve the alignment of the algorithm. However, it is important to note that by using the approximate alignment to resample the audio files, the results are much better than without resampling. One explanation for the limited improvement in performance is that the spectral centroid and spectral flux might not be too susceptible to noise.

5. CONCLUSION

This paper presented a method for accurately aligning recordings of a concert event given that these recordings are noisy snippets. The results show that audio fingerprints are better suited than pitch chroma for the task of representing noisy audio and that dynamic time warping performs better than cross-correlation for the alignment. Using a finer alignment on the resampled audio shows promise; however, the results are still unsatisfactory. The real world data has been made publicly available, and the used modifications of the data is published online⁴.

The biggest drawback of the algorithm is its inability to handle pitch shifts in audio recordings very well— a known issue with many fingerprinting systems. If the current audio fingerprinting algorithm is replaced with an algorithm that is robust to noise as well as to pitch shifts, we expect the performance of the system would improve considerably on our real world dataset.

Future work on this task will focus on the actual rendition of the complete event once the alignment is known and possibly combine audio with video. Selecting the segments, determining fade points, durations, and type in the overlapping regions, are all interesting and challenging tasks that have not been researched in depth yet.

⁴ <https://github.com/VinodS7/ConcertStitch-dataset>

6. REFERENCES

- [1] Jean-Julien Aucouturier, Francois Pachet, et al. Music similarity measures: What's the use? In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 13–17, 2002.
- [2] M. A. Bartsch and G. H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104, 2005.
- [3] A. Berenzweig, B. Logan, D. P. W. Ellis, and B. Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76, 2004.
- [4] N. J. Bryan, P. Smaragdis, and G. J. Mysore. Clustering and synchronizing multi-camera video via landmark cross-correlation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2389–2392, 2012.
- [5] C. V. Cotton and D. P. W. Ellis. Audio fingerprinting to identify multiple videos of an event. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2386–2389, March 2010.
- [6] Steven B Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. In *Readings in speech recognition*, pages 65–74. Elsevier, 1990.
- [7] Frederic Font, Gerard Roma, and Xavier Serra. Freesound technical demo. In *ACM International Conference on Multimedia (MM'13)*, pages 411–412, Barcelona, Spain, 21/10/2013 2013.
- [8] Jonathan T. Foote. Content-based retrieval of music and audio, 1997.
- [9] Siddharth Gururani and Alexander Lerch. Automatic Sample Detection in Polyphonic Music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, 2017.
- [10] Jaap Haitsma and Ton Kalker. A highly robust audio fingerprinting system. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 107–115, 2002.
- [11] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, Nov 2012.
- [12] Jeff Howe. The rise of crowdsourcing. *Wired magazine*, 14:1–4, 2006.
- [13] Lyndon Kennedy and Mor Naaman. Less talk, more rock: Automated organization of community-contributed collections of concert videos. In *Proc. of the 18th International Conference on World Wide Web*, pages 311–320, New York, 2009.
- [14] Holger Kirchhoff and Alexander Lerch. Evaluation of Features for Audio-to-Audio Alignment. *Journal of New Music Research*, 40(1):27–41, 2011.
- [15] Alexander Lerch. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. Wiley-IEEE Press, Hoboken, 2012.
- [16] Catherine C. Marshall and Frank M. Shipman. Saving, reusing, and remixing web video: Using attitudes and practices to reveal social norms. In *Proceedings of the 22Nd International Conference on World Wide Web, WWW '13*, pages 885–896, New York, NY, USA, 2013.
- [17] Matthias Mauch, Sebastian Ewert, et al. The audio degradation toolbox and its application to robustness evaluation. 2013.
- [18] Meinard Müller. *Information Retrieval for Music and Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
- [19] S. Ravuri and D. P. W. Ellis. Cover song detection: From high scores to general classification. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 65–68, March 2010.
- [20] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, Feb 1978.
- [21] Prarthana Shrestha, Peter H.N. de With, Hans Weda, Mauro Barbieri, and Emile H.L. Aarts. Automatic mashup generation from multiple-camera concert recordings. In *Proc. of the 18th ACM International Conference on Multimedia*, pages 541–550, New York, 2010.
- [22] Joren Six and Marc Leman. Synchronizing multimodal recordings using audio-to-audio alignment. *Journal on Multimodal User Interfaces*, 9(3):223–229, Sep 2015.
- [23] George Tzanetakis, Andrey Ermolinskyi, and Perry Cook. Pitch histograms in audio and symbolic music information retrieval. *Journal of New Music Research*, 32(2):143–152, 2003.
- [24] Sami Vihavainen, Sujeet Mate, Lassi Seppälä, Francesco Cricri, and Igor D.D. Curcio. We want more: Human-computer collaboration in mobile social video remixing of music concerts. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 287–296, New York, 2011.
- [25] Avery Wang. An industrial strength audio search algorithm. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 2003, pages 7–13. Washington, D.C., 2003.
- [26] Thomas Wilmering, Florian Thalmann, and Mark B. Sandler. Grateful live: Mixing multiple recordings of a dead performance into an immersive experience. In *Audio Engineering Society Convention 141*, 2016.

A DATA-DRIVEN APPROACH TO MID-LEVEL PERCEPTUAL MUSICAL FEATURE MODELING

Anna Aljanaki

Institute of Computational Perception,
Johannes Kepler University
aljanaki@gmail.com

Mohammad Soleymani

Swiss Center for Affective Sciences,
University of Geneva
mohammad.soleymani@unige.ch

ABSTRACT

Musical features and descriptors could be coarsely divided into three levels of complexity. The bottom level contains the basic building blocks of music, e.g., chords, beats and timbre. The middle level contains concepts that emerge from combining the basic blocks: tonal and rhythmic stability, harmonic and rhythmic complexity, etc. High-level descriptors (genre, mood, expressive style) are usually modeled using the lower level ones. The features belonging to the middle level can both improve automatic recognition of high-level descriptors, and provide new music retrieval possibilities. Mid-level features are subjective and usually lack clear definitions. However, they are very important for human perception of music, and on some of them people can reach high agreement, even though defining them and therefore, designing a hand-crafted feature extractor for them can be difficult. In this paper, we derive the mid-level descriptors from data. We collect and release a dataset¹ of 5000 songs annotated by musicians with seven mid-level descriptors, namely, melodiousness, tonal and rhythmic stability, modality, rhythmic complexity, dissonance and articulation. We then compare several approaches to predicting these descriptors from spectrograms using deep-learning. We also demonstrate the usefulness of these mid-level features using music emotion recognition as an application.

1. INTRODUCTION

In music information retrieval, features extracted from audio or a symbolic representation are often categorized as low or high-level [5], [17]. There is no clear boundary between these concepts and the terms are not used consistently. Usually, features that were extracted using a small analysis window that does not contain temporal information are called low-level (e.g., spectral features, MFCCs, loudness). Features that are defined within a longer con-

text (and often related to music theoretical concepts) are called high-level (key, tempo, melody). In this paper, we will look at these levels from the point of view of human perception, and define what constitutes low, middle and high levels depending on complexity and subjectivity of a concept. Unambiguously defined and objectively verifiable concepts (beats, onsets, instrument timbres) will be called low-level. Subjective, complex concepts that can only be defined by considering every aspect of music will be called high-level (mood, genre, similarity). Everything in between we will call mid-level.

Musical concepts can best be viewed and defined through the lens of human perception. It is often not enough to approximate them through a simpler concept or feature. For instance, music speed (whether music is perceived as fast or slow) is not explained by or equivalent to tempo (beats per minute). In fact, perceptual speed is better approximated (but not completely explained) by onset rate [8]. There are many examples of mid-level concepts: harmonic complexity, rhythmic stability, melodiousness, tonal stability, structural regularity [10], [24]. Such meta language could be used to improve search and retrieval, to add interpretability to the models of high-level concepts, and may be even break the glass ceiling in the accuracy of their recognition.

In this paper we collect a dataset and model these concepts directly from data using transfer learning.

2. RELATED WORK

Many algorithms have been developed to model features describing such aspects of music as articulation, melodiousness, rhythmic and dynamic patterns. MIRToolbox and Essentia frameworks offer many algorithms that can extract features related to harmony, rhythm, articulation and timbre [13], [3]. These features are usually extracted using some hand-crafted algorithm and have a differing amount of psychoacoustic and perceptual basis.

For example, Salamon et al. developed a set of melodic features which extract pitch contours from a melody obtained with a melody extraction algorithm [22]. There were proposed measures like percussiveness [17], pulse clarity [12], danceability [23]. Panda et al. proposed a set of algorithms to extract descriptors related to melody, rhythm and texture from MIDI and audio [19]. It is out of our scope to review all existing algorithms for detecting

¹ <https://osf.io/5aupt/>



© Anna Aljanaki, , Mohammad Soleymani. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Anna Aljanaki, , Mohammad Soleymani. "A data-driven approach to mid-level perceptual musical feature modeling", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

Perceptual Feature	Criteria when comparing two excerpts	Cronbach's α
Melodiousness	To which excerpt do you feel like singing along?	0.72
Articulation	Which has more sounds with staccato articulation?	0.8
Rhythmic stability	Imagine marching along with music. Which is easier to march along with?	0.69
Rhythmic complexity	Is it difficult to repeat by tapping? Is it difficult to find the meter? Does the rhythm have many layers?	0.27 (0.47)
Dissonance	Which excerpt has noisier timbre? Has more dissonant intervals (tritones, seconds, etc.)?	0.74
Tonal stability	Where is it easier to determine the tonic and key? In which excerpt are there more modulations?	0.44
Modality	Imagine accompanying this song with chords. Which song would have more minor chords?	0.69

Table 1. Perceptual mid-level features and the questions that were provided to raters to help them compare two excerpts.

what we call mid-level perceptual music concepts.

All the algorithms listed so far were designed with some hypothesis about music perception in mind. For instance, Essentia offers an algorithm to compute sensory dissonance, which sums up the dissonance values for each pair of spectral peaks, based on dissonance curves obtained from perceptual measurements [20]. Such an algorithm measures a specific aspect of music in a transparent way, but it is hard to say, whether it captures all the aspect of a perceptual feature.

Friberg et al. collected perceptual ratings for nine features (rhythmic complexity and clarity, dynamics, harmonic complexity, pitch, etc.) for a set of 100 songs and modeled them using available automatic feature extractors, which showed that algorithms can cope with some concepts and fail with some others [8]. For instance, for such an important feature like modality (majorness) there is no adequate solution yet. It was also shown that with just several perceptual features it is possible to model emotion in music with a higher accuracy than it is possible using features, extracted with MIR software [1], [8], [9].

In this paper we propose an approach to mid-level feature modeling that is more similar to automatic tagging [6]. We try to approximate the perceptual concepts by modeling them straight from the ratings of listeners.

3. DATA COLLECTION

From the literature ([10], [24], [8]) we composed a list of perceptual musical concepts and picked 7 recurring items. Table 1 shows the selected terms. The concepts that we are interested in stem from musicological vocabulary. Identifying and naming them is a complicated task that requires musical training. This doesn't mean that these concepts are meaningless and are not perceived by an average music listener, but we can not trust an average listener to apply the terms in a consistent way. We used Toloka² crowd-

sourcing platform to find people with musical training to do the annotation. We invited anyone who has music education to take a musical test, which contained questions on harmony (tonality, identifying mode of chords), expressive terms (rubato, dynamics, articulation), pitch and timbre. Also, we asked the crowd-sourcing workers to shortly describe their music education. From 2236 people who took the test slightly less than 7% (155 crowd sourcing workers) passed it and were invited to participate in the annotation.

3.0.1 Definitions

The terminology (articulation, mode, etc.) that we use is coming from musicology, but it was not designed to be used in a way that we use it. For instance, the concept of articulation is defined for a single note (or can also be extended to a group of notes). Applying it to a real-life recording with possibly several instruments and voices is not an easy task. To ensure common understanding, we offer the annotators a set of definitions as shown in Table 1. The general principle is to consider the recording as a whole.

3.1 Pairwise comparisons

It is easier for annotators to compare two items using a certain criterion, then to give a rating on an absolute scale, and especially so for subjective and vaguely defined concepts [14]. Then, a ranking can be formed from pairwise comparisons. However, annotating a sufficient amount of songs using pairwise comparisons is too labor intensive. Collecting a full pairwise comparison matrix (not counting repetitions and self-similarity) requires $(n^2 - n)/2$ comparisons. For our desired target of 5000 songs, that would mean ≈ 12.5 million comparisons. It is possible to construct a ranking with less than a full pairwise comparison matrix, but still for a big dataset it is not a feasible approach. We combine the two approaches. In order to do that, we first collected pairwise comparisons for a small

² toloka.yandex.ru

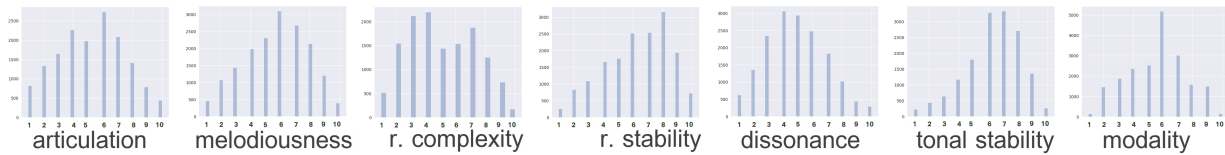


Figure 1. Distribution of discrete ratings per perceptual feature.

Feature	Articulation	R. comlexity	R. Stability	Dissonance	Tonal stability	Mode
Melodiousness	-0.13	-0.22	0.27	-0.59	0.58	-0.22
Articulation		0.39	0.60	0.45	-0.05	-0.14
R. complexity			-0.009	0.48	-0.30	0.06
R. stability				0.06	0.36	-0.17
Dissonance					-0.55	0.23
Tonal stability						-0.16

Table 2. Correlations between the perceptual mid-level features.

amount of songs, obtained a ranking, and then created an absolute scale that we used to collect the rankings.

In this way, we also implicitly define our concepts through examples without a need to explicitly describe all their aspects.

3.1.1 Music selection

For pairwise comparisons, we selected 100 songs. This music needed to be diverse, because it was going to be used as examples and needed to be able to represent the extremes. We used 2 criteria to achieve that - genre and emotion. From each of the 5 music preference clusters of Rentfrow et al. [21] we selected a list of genres belonging to these clusters and picked songs from the DEAM dataset [2] belonging to these genres (pop, rock, hip-hop, rap, jazz, classical, electronic), taking 20 songs from each of the preference clusters. Also, using the annotations from DEAM, we assured that the selected songs are uniformly distributed over the four quadrants of valence/arousal plane. From each of the songs we cut a segment of 15 seconds.

For a set of 100 songs we collected 2950 comparisons. Next, we created a ranking by counting the percentage of comparisons won by a song relative to an overall number of comparisons per song. By sampling from that ranking we created seven scales with song examples from 1 to 9 for each of the mid-level perceptual features (for instance, from the least melodious (1) to the most melodious (9)). Some of the musical examples appeared in several scales.

3.2 Ratings on 7 perceptual mid-level features

The ratings were again collected on Toloka platform, and the workers were selected using the same musical test. The rating procedure was as follows. First, a worker listened to a 15-second excerpt. Next, for a certain scale (for instance, articulation), a worker compared an excerpt with examples arranged from "legato" to "staccato" and found a proper rating. Finally, this was repeated for each of the 7 perceptual features.

3.2.1 Music selection

Most of the dataset music consists of Creative Commons licensed music from jamendo.com and magnatune.com. For annotation, we cut 15 seconds from the middle of the song. In the dataset, we provide the segments and the links to a full song. There is a restriction of no more than 5 songs from the same artist. The songs from jamendo.com were also filtered by popularity, in a hope to get music of a better recording quality. We also reused the music from datasets annotated with emotion [7], [18], [15] which we are going to use to indirectly test the validity of the annotations.

3.2.2 Data

Figure 1 shows the distributions of the ratings for every feature. The music in the dataset leans slightly towards being rhythmically stable, tonally stable and consonant. The scales could be also readjusted to have more examples in the regions of the most density. That might not necessarily help, because the observed distributions could also be the artifacts of people preferring to avoid the extremes. Table 2 shows the correlation between different perceptual features. There is a strong negative correlation between melodiousness and dissonance, a positive relationship between articulation and rhythmic stability. Tonal stability is negatively correlated with dissonance and positively with melodiousness.

3.3 Consistency

Any crowd-sourcing worker could stop annotating at any point, so the amount of annotated songs per person varied. An average amount of songs per worker was 187.01 ± 500.68 . On average, it took ≈ 2 minutes to answer all the seven questions for one song. Our goal was to collect 5 annotations per song, which amounts to ≈ 833 man-hours. In order to ensure quality, a set of songs with high quality annotations (high agreement by well-performing workers) was interlaced with new songs, and the annotations of every crowd-sourcing worker were compared against that golden standard. The workers that gave answers very far

Emotional dimension or category	Pearson's ρ (prediction)	Important features
Valence	0.88	Mode (major), melodiousness (pos.), dissonance (neg.)
Energy	0.79	Articulation (staccato), dissonance (pos.)
Tension	0.84	Dissonance (pos.), melodiousness (neg.)
Anger	0.65	Dissonance (pos.), mode (minor), articulation (staccato)
Fear	0.82	Rhythm stability (neg.), melodiousness (neg.)
Happy	0.81	Mode (major), tonal stability (pos.)
Sad	0.73	Mode (minor), melodiousness (pos.)
Tender	0.72	Articulation (legato), mode (minor), dissonance (neg.)

Table 3. Modeling emotional categories in Soundtracks dataset using seven mid-level features.

from the standard were banned. Also, the answers were compared to the average answer per song, and workers whose standard deviation was close to one one resulting from random guessing were also banned and their answers discarded. The final annotations contain answers of 115 workers out of a pool of 155, who passed the musical test.

Table 1 shows a measure of agreement (Cronbach's α) for each of the mid-level features. The annotators reach good agreement for most of the features, except rhythmic complexity and tonal stability. We created a different musical test, containing only questions about rhythm, and collected more annotations. Also, we provided more examples on the rhythm complexity scale. It helped a little (Cronbach's α improved from 0.27 to 0.47), but still rhythmic complexity has much worse agreement than other properties. In a study of Friberg and Hedblad [8], where similar perceptual features were annotated for a small set of songs, the situation was similar. The least consistent properties were harmonic complexity and rhythmic complexity.

We average the ratings for every mid-level feature per song. The annotations and the corresponding excerpts (or links to external reused datasets) are available online (osf.io/5aupt). All the experiments below are performed on averaged ratings.

3.4 Emotion dimensions and categories

Soundtracks dataset contains 15 second excerpts from film music, annotated with valence, arousal, tension, and 5 basic emotions [7].

We show that our annotations are meaningful by using them to model musical emotion in Soundtracks dataset. The averaged ratings per song for each of the seven mid-level concepts are used as features in a linear regression model (10-fold cross-validation).

Table 3 shows the correlation coefficient and the most important features for each dimension, which are consistent with the findings in the literature [10]. We can model most dimensions well, despite not having any information about loudness and tempo.

Cluster	AUC	F-measure
Cluster 1 passionate, confident	0.62	0.38
Cluster 2 cheerful, fun	0.7	0.5
Cluster 3 bittersweet	0.8	0.67
Cluster 4 humorous	0.65	0.45
Cluster 5 aggressive	0.78	0.64

Table 4. Modeling MIREX clusters with perceptual features.

3.5 MIREX clusters

Multimodal dataset contains 903 songs annotated with 5 clusters used in MIREX Mood recognition competition³ [18]. Table 4 shows results of predicting the five clusters using the seven mid-level features and an SVM classifier. The average weighted F1 measure on all the clusters on this dataset is 0.54. In [18], with an SVM classifier trained on 253 audio features, extracted with various toolboxes, F1 measure was 44.9, and 52.3 with 98 melodic features. By combining these feature sets and doing feature selection by using feature ranking, the F1 measure was increased to 64.0. Panda et al. hypothesize that Multimodal dataset is more difficult than MIREX dataset (their method performed better (0.67) in MIREX competition than on their own dataset). In MIREX data, the songs went through an additional annotation step to ensure agreement on cluster assignment, and only songs that 2 out of 3 experts agreed on were kept.

³ www.music-ir.org/mirex

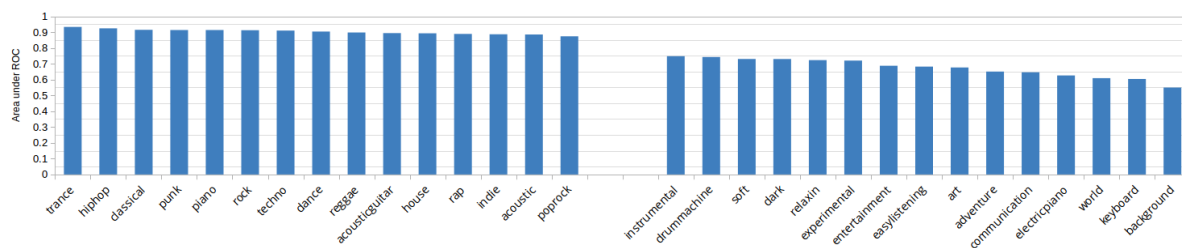


Figure 2. AUC per tag on the test set.

4. EXPERIMENTS

We left out 8% of the data as a test set. We split the train set and test set by performer (no performer from the test set appears in the training set). Also, all the performers in the test set are unique. For pretraining, we used songs from `jamendo.com`, making sure that the songs used for pretraining do not reappear in the test set. The rest of the data was used for training and validation (whenever we needed to validate any hyperparameters, we used 2% of the train set for that).

From each of the 15-second excerpts we computed a mel-spectrogram with 299 mel-filters and a frequency range of 18000Hz, extracted with 2048 sample window (44100 sampling rate) and a hop of 1536. In order to use it as an input to a neural network, it was cut to a rectangular shape (299 by 299) which corresponds to about 11 seconds of music. Because the original mel-spectrogram is a bit larger, we can randomly shift the rectangular window and select a different set. For some of the songs, full-length songs are also available, and it was possible to extract the mel-spectrogram from any place in a song, but in practice this worked worse than selecting a precise spot.

We also tried other data representations: spectrograms and custom data representations (time-varying chroma for tonal features and time-varying bark-bands for rhythmic features). Custom representations were trained with a two-layer recurrent network. These representations worked worse than mel-spectrograms with a deep network.

4.1 Training a deep network

We chose Inception v3 architecture [4]. First five layers are convolutional layers with 3 by 3 filters. Twice max-pooling is applied. The last layers of the network are the so-called "inception layers", which apply filters of different size in parallel and merge the feature maps later. We begin by training this network without any pretraining.

4.1.1 Transfer learning

With a dataset of only 5000 excerpts, it is hard to prevent overfitting when learning features from the very basic music representation (mel-spectrogram), as it was done in [6] on a much larger dataset. In this case, transfer learning can help.

4.1.2 Data for pretraining

We crawl data and tags from Jamendo, using the API provided by this music platform. We select all the tags, which were applied to at least 3000 songs. That leaves us with 65 tags and 184002 songs. For training, we extract a mel-spectrogram from a random place in a song. We leave 5% of the data as a test set. After training on mini-batches of 32 examples with Adam optimizer for 29 epochs, we achieve an average area under receiver-operator curve of 0.8 on the test set. The AUC on the test set grouped by tag are shown on Figure 2 (only 15 best and 15 worst performing tags). Some of the songs in the mid-level feature dataset also were chosen from Jamendo.

4.1.3 Transfer learning on mid-level features

The last layer of Inception, before the 65 neurons that predict classes (tags), contains 2048 neurons. We pass through the mel-spectrograms of the mid-level feature dataset and extract the activations of this layer. We normalize these extracted features using mean and standard deviation of the training set. On the training set, we fit a PCA with 30 principle components (the number was chosen based on decline of eigenvalues of the components) and then apply the learned transformation on a validation and test set. On a validation set, we tune parameters of a SVR with a radial basis function kernel and finally, we predict the seven mid-level features on the test set.

4.2 Fine-tuning trained model for mid-level features

On top of the last Inception layer we add two fully-connected layers with 150 and 30 neurons, both with ReLU activation, and an output layer with 7 nodes with no activation (we train on all the features at the same time). First, we freeze the pre-trained weights of the Inception and train the last layer weights until there's no improvement on the validation set anymore. At this point, the network reaches the same performance on the test set as it reached using transfer learning and PCA (which is what we would expect). Now, we unfreeze the weights and with a small learning rate continue training the whole network until it stops improving on validation set.

4.3 Existing algorithms

There are many feature extraction frameworks for MIR. Some of those (jAudio, Aubio, Marsyas) only offer timbral and spectral features, others (Essentia, MIRToolbox,

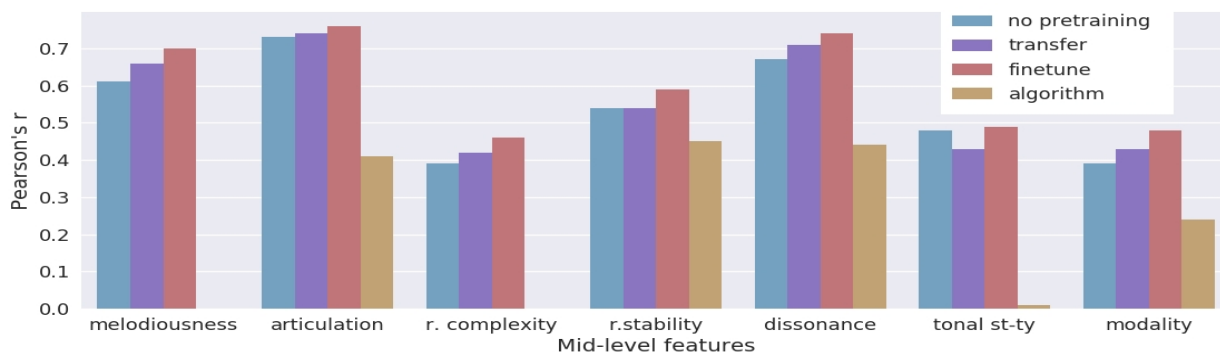


Figure 3. Performance of different methods on mid-level feature prediction.

VAMP Plugins for Sonic Annotator) offer features, which are similar to the mid-level features of this paper. Figure 3 shows the correlation of some of these features with our perceptual ratings:

1. *Articulation*. MIRToolbox offers features describing characteristics of onsets (attack time, attack slope, leap (duration of attack), decay time, slope and leap. Out of this features leap was chosen (it had the strongest correlation with perceptual articulation feature).
2. *Rhythmic stability*. Pulse clarity (MIRToolbox) [16].
3. *Dissonance*. Both Essentia and MIRToolbox offer a feature describing sensory dissonance (in MIRToolbox, it is called roughness), which is based on the same research of dissonance perception [20]. We extract this feature and inharmonicity. Inharmonicity only had a weak (0.22) correlation with perceptual dissonance. Figure 3 shows a result for the dissonance measure.
4. *Tonal stability*. HCDF (harmonic change detection function) in MIRToolbox is a feature measuring the flux of a tonal centroid [11]. This feature was not correlated with our tonal stability feature.
5. *Modality*. MIRToolbox offers a feature called mode, which is based on an uncertainty in determining the key using pitch-class profiles.

We could not find features corresponding to melodiousness and rhythmic complexity. Perceptual concepts lack clear definitions, so it is impossible to say that the feature extractor algorithms are supposed to directly measure the same concepts that we had annotated. However, from Figure 3 we can see that the chosen descriptors do indeed capture some part of variance in the perceptual features.

4.4 Results

Figure 3 shows the results for every mid-feature. For all the mid-features, the best result was achieved by pretraining and fine-tuning the network. Melodiousness, articulation and dissonance could be predicted with a much bet-

ter accuracy than rhythmic complexity, tonal and rhythmic stability, and mode.

5. FUTURE WORK

In this paper, we only investigated seven perceptual features. Other interesting features include tempo, timbre, structural regularity. Rhythmic complexity and tonal stability features had low agreement. It is probable that contributing factors need to be explicitly specified and studied separately. The accuracy could be improved for modality and rhythmic stability. It is not clear whether strong correlations between some features are an artifact of the data selection or music perception.

6. CONCLUSION

Mid-level perceptual music features could be used for music search and categorization and improve music emotion recognition methods. However, there are multiple challenges in extracting such features: first, such concepts lack clear definitions, and we do not quite understand the underlying perceptual mechanisms yet. In this paper, we collect annotations for seven perceptual features and model them by relying on listener ratings. We provide the listeners with scales with examples instead of definitions and criteria. Listeners achieved good agreement on all the features but two (rhythmic complexity and tonal stability). Using deep learning, we model the features from data. Such an approach has its advantages as compared to specific algorithm-design by being able to pick appropriate patterns from the data and achieve better performance than an algorithm based on a single aspect. However, it is also less interpretable. We release the mid-level feature dataset, which can be used to further improve both algorithmic and data-driven methods of mid-level feature recognition.

7. ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC) under the EUs Horizon 2020 Framework Program (ERC Grant Agreement number 670035, project "Con Espresione"). This work was also supported by an FCS grant.

8. REFERENCES

- [1] A. Aljanaki, F. Wiering, and R.C. Veltkamp. Computational modeling of induced emotion using gems. In *15th International Society for Music Information Retrieval Conference*, 2014.
- [2] A. Aljanaki, Y.-H. Yang, and M. Soleymani. Developing a benchmark for emotional analysis of music. *PLOS ONE*, 12(3), 2017.
- [3] D. Bogdanov, N. Wack, E. Gomez, S. Gulati, P. Herrera, and et al. O. Mayor. Essentia: an audio analysis library for music information retrieval. In *14th International Society for Music Information Retrieval Conference*, pages 493–498, 2013.
- [4] S. Ioffe J. Shlens Z. Wojna C. Szegedy, V. Vanhoucke. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [5] M.A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-Based Music Information Retrieval: Current Directions and Future Challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.
- [6] K. Choi, G. Fazekas, and M. Sandler. Convnet: Automatic tagging using deep convolutional neural networks. In *17th International Society for Music Information Retrieval Conference*, 2016.
- [7] T. Eerola and J.K. Vuoskoski. A comparison of the discrete and dimensional models of emotion in music. *Psychology of Music*, 39(1):1849, 2011.
- [8] A. Friberg and A. Hedblad. A comparison of perceptual ratings and computed audio features. In *8th Sound and Music Computing Conference*, pages 122–127, 2011.
- [9] A. Friberg, E. Schoonderwaldt, A. Hedblad, M. Fabiani, and A. Elowsson. Using listener-based perceptual features as intermediate representations in music information retrieval. *The Journal of the Acoustical Society of America*, 136(4):1951–63, 2014.
- [10] A. Gabrielsson and E. Lindstrm. *Music and Emotion: Theory and Research*, chapter The Influence of Musical Structure on Emotional Expression, page 22348. Oxford University Press, 2001.
- [11] Christopher Harte, Mark Sandler, and Martin Gasser. Detecting harmonic change in musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia - AMCM '06*, page 21. ACM Press, 2006.
- [12] O. Lartillot, T. Eerola, P. Toivainen, and J. Fornari. Multi-feature modeling of pulse clarity: Design, validation, and optimization. In *9th International Conference on Music Information Retrieval*, 2008.
- [13] O. Lartillot, P. Toivainen, and T. Eerola. A matlab toolbox for music information retrieval. *Data Analysis, Machine Learning and Applications, Studies in Classification, Data Analysis, and Knowledge Organization*, 2008.
- [14] J. Madsen, Jensen B. S, and J. Larsen. Predictive Modeling of Expressed Emotions in Music Using Pairwise Comparisons. pages 253–277. Springer, Berlin, Heidelberg, 2013.
- [15] R. Malheiro, R. Panda, P. Gomes, and R. Paiva. Bi-modal music emotion recognition: Novel lyrical features and dataset. In *9th International Workshop on Music and Machine Learning MML2016*, 2016.
- [16] Petri Toivainen Jose Fornari Olivier Lartillot, Tuomas Eerola. Multi-feature modeling of pulse clarity: Design, validation, and optimization. In *9th International Conference on Music Information Retrieval*, 2008.
- [17] E. Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Vienna University of Technology, 2012.
- [18] R. Panda, R. Malheiro, B. Rocha, A. Oliveira, and R. P. Paiva. Multi-modal music emotion recognition: A new dataset, methodology and comparative analysis. In *10th International Symposium on Computer Music Multidisciplinary Research*, 2013.
- [19] Renato Panda, Ricardo Manuel Malheiro, and Rui Pedro Paiva. Novel audio features for music emotion recognition. *IEEE Transactions on Affective Computing*.
- [20] R. Plomp and W. J. M. Levelt. Tonal Consonance and Critical Bandwidth. *The Journal of the Acoustical Society of America*, 38(4):548560, 1965.
- [21] P. J. Rentfrow, L. R. Goldberg, and D. J. Levitin. The structure of musical preferences: a five-factor model. *Journal of personality and social psychology*, 100(6):1139–57, 2011.
- [22] J. Salamon, B. Rocha, and E. Gomez. Musical genre classification using melody features extracted from polyphonic music signals. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 81–84. IEEE, 2012.
- [23] S. Streich and P. Herrera. Detrended fluctuation analysis of music signals danceability estimation and further semantic characterization. In *AES 118th Convention*, 2005.
- [24] L. Wedin. A Multidimensional Study of Perceptual-Emotional Qualities in Music. *Scandinavian Journal of Psychology*, 13:241257, 1972.

DISAMBIGUATING MUSIC ARTISTS AT SCALE WITH AUDIO METRIC LEARNING

Jimena Royo-Letelier

Romain Hennequin

Viet-Anh Tran

Manuel Moussallam

Deezer, 12 rue d'Athènes, 75009 Paris, France

research@deezer.com

ABSTRACT

We address the problem of disambiguating large scale catalogs through the definition of an unknown artist clustering task. We explore the use of metric learning techniques to learn artist embeddings directly from audio, and using a dedicated homonym artists dataset, we compare our method with a recent approach that learn similar embeddings using artist classifiers. While both systems have the ability to disambiguate unknown artists relying exclusively on audio, we show that our system is more suitable in the case when enough audio data is available for each artist in the train dataset. We also propose a new negative sampling method for metric learning that takes advantage of side information such as music genre during the learning phase and shows promising results for the artist clustering task.

1. INTRODUCTION

1.1 Motivation

With contemporary on-line music catalogs typically proposing dozens of millions of recordings, a major problem is the lack of an universal and reliable mean to identify music artists. Contrarily to albums' and tracks' ISRC¹, and despite some initiative such as ISNI², there exist no unique standardized identifier for artists in the industry. As a direct consequence, the name of an artist remains its de-facto identifier in practice although it results in common ambiguity issues. For example, name artist collisions (e.g. *Bill Evans* is the name of a jazz pianist but also the name of a jazz saxophonist and the name of a blackgrass banjo player) or artist aliases (e.g. *Youssou N'Dour* vs. *Youssou Ndour*, *Simon & Garfunkel* vs *Paul Simon and Art Garfunkel*, *Cat Stevens* vs *Yusuf Islam*) are usual. Relying on human resources to clean or verify all artists in the database is practically impossible, although a major issue resulting from these ambiguities is the difficulty to correctly credit

artists for their work and the confusion that may arise for end users while exploring catalogs.

Automatically distinguishing between artists is a complicated task, even for human specialists, since there is no one to one relation between a track and an artist. Tracks can be performed by several artists (e.g. *duets* and *features*). Albums may contain tracks from different artists (e.g. in *compilations*). Even artist denominations may drastically evolve during their careers. In this work, our goal is, given a set of recordings, to find a partition of this set for which all tracks from a given subset are associated to the same artist.

In the MIR literature, problems dealing with artist assignment from a recording are most of the time addressed as a classification problem [3, 9, 10], where a set of predefined artists is known. This is not a real case scenario for evolving large music catalogs, since the number of artists can be huge (several millions) and new artists are added every day. In this paper, we propose a new task of unknown artists clustering from audio, without having any ground truth data about the identities of the artist nor a prior information about the number of different artists present in a cluster. To the best of our knowledge, there are no prior work addressing this. Disambiguating homonym artists is a practical application of this task, where a set of tracks must be split into an unknown number of clusters, each corresponding to a different artist entity. We believe that accurately solving this task could results in a major improvement in the quality of large sized catalogs.

To tackle this new task, we propose to use metric learning methods to train a system that outputs artist embeddings from audio. Indeed, as we will explain later, metric learning objective function is primarily designed to ensure that embeddings of samples from the same entity are clustered together. Metric learning also offers the interesting possibility of controlling what an embedding system learns by means of the sampling necessary to feed its loss. Here we also suggest to leverage musical relationships among audio tracks as source of information to strengthen the representation learning, allowing to incorporate music side information -such as genre, mood or release date- to the training process.

1.2 Overview

The paper is organized as follows. In Section 2 we expose how this paper relates to prior work. In Section 3, we detail the metric learning system used to learn artist embeddings

¹ <http://isrc.ifpi.org>

² <http://www.isni.org>



© First author, Second author, Third author, Fourth author. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** First author, Second author, Third author, Fourth author. "Disambiguating Music Artists at Scale with Audio Metric Learning", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

from audio. In Section 4, we introduce the newly proposed artist disambiguation task and the datasets used for experiments. In Section 5, we show our results and compare to the previous systems. Finally, we draw conclusions in Section 6.

2. PREVIOUS WORKS

In this paper we propose a method to learn artist embeddings from audio. This approach falls in a general category of methods in identity disambiguation problems that try to learn a parametric map from content samples (for instance face pictures [18] or speaker speech recordings [5]) to a metric space, so that same identity samples are closely located and different identity samples are far away. The same idea has been exploited to address item retrieval in large image datasets (cars, birds, on-line products) [21, 25] and to learn audio representations for sound event classification [12].

For music artist, the embedding approach has been addressed previously in [3], where a representation space is constructed using the output probabilities of a multi-class classification system with Mel-Frequency Cepstral Coefficients (MFCC) as input. The space is only used to address the classification of known artists, but would also be suitable to unknown ones. Convolutional deep belief networks were used to learn features that were afterwards used for artist classification [14]: while the evaluation is done on only four artists, the approach learns a representation from unlabeled data which can generalize to unknown artists. In [28] the authors train a linear system that attempts to capture the semantic similarities between items in a large database by modeling audio, artist names and tags in a single space. The system is trained with multi-task ranking losses, which highly resembles metric learning methods: each ranking loss takes as input triplets of samples from possibly different kind of sources. Although this approach is very promising, both for the objective function and the use of side information, the same artists are used for train and evaluation. Unfortunately, direct comparison is hard since little details are given about how datasets are obtained. In [16], artist embeddings are learned using 1d convolutional neural networks trained as mono-label classifier used afterwards as general features extractors. Their approach is able to deal with artists not seen during the training phase. Information is given on how train databases are obtained from the Million Song Dataset (MSD) [4] using the *artist7*³ labels.

Several other works address directly the artist classification problem. The current state of the art is inspired by speaker recognition system and makes use of I-vectors to separate artists [9–11]. In [20], an artist fingerprint based on MFCC is proposed to tackle the problem of retrieving an artist from an audio track at scale. In [13], multivariate similarity kernel methods are proposed to tackle (among other tasks) artist classification. In [22], the authors focus on the main vocalist, and then vocal separation is used

as a preprocessing for artist classification. In [2], a multi-modal approach taking advantage of both lyrics and audio is proposed to perform artist classification. In [7], the authors use a convolutional neural network to perform artist recognition on a 50 artists dataset. While the techniques employed in these works are of interest for their potential use in extracting representations of unknown artists, they usually only consider at the classification of known artists and give no results on the generalization to new artist not seen during training phase, nor address the extraction of representations useful for unknown artists.

3. PROPOSED METHOD

3.1 Metric learning

The main idea in metric learning is to learn a metric preserving map f from one metric space into another. Using a discrete metric in the first space, this can be exploited to learn embedding spaces where distances correspond with membership to some category. Here we use the discrete metric defined by artist membership in the space of musical audio recordings.

In this membership context, the learning of f can be achieved using the triplet loss mechanism [27]. This relies on using triplets $X = (x_a, x_+, x_-)$, where (x_a, x_+) is a positive pair (samples with same artist membership) and (x_a, x_-) is a negative pair (samples with different artist membership). The triplet loss ℓ contains a term that tries to bring closer the images by f of samples in a positive pairs, and another term that tries to separate the images by f of samples in a negative pair. It writes

$$\ell(X) = \|f(x_a) - f(x_+)\|_2^2 - \|f(x_a) - f(x_-)\|_2^2 + \alpha|_+ \quad (1)$$

where $|s|_+ = \max(0, s)$. The parameter α here before is a positive constant used to ensure a margin between distances of points from positive and negative pairs. In order to prevent the system to tend to unsuitable states, where all points are mapped to 0, or where the map f takes arbitrary large values, we constraint the embedding to lie in the unit sphere by imposing $\|f(x)\|_2 = 1$.

3.2 Training and sampling strategies

We train our system using Stochastic Gradient Descent over batches of triplets. Since optimizing over triplets that are already well separated by the system (i.e. $\ell(X) = 0$) is unnecessary and costly, the system is fed only with triplets that are not yet separated. These are called *hard* triplets if the value of

$$\|f(x_a) - f(x_+)\|_2^2 - \|f(x_a) - f(x_-)\|_2^2$$

is negative, and *semi-hard* triplets if this value is in $[0, \alpha]$. We set $\alpha = 0.2$ as in [5, 18]. Notice that during triplets selection process the i -th state of the system is used to compute embeddings to filter data for the $(i + 1)$ -th parameters update.

³ <http://developer.7digital.com>

At each iteration, n samples are chosen from N different artists to form positive pairs. For each positive pair one negative sample from the $N - 1$ left artists is taken to form a triplet. This leave us with $Nn(n - 1)/2$ triplets per iteration that are given to the system for optimization only if they are labeled as hard or semi-hard.

Notice from the expression (1) that the gradients of ℓ are close to zero when the system maps all entries to very close points in the space, so in practice learning can fail with the system being stuck in a “collapsed” state where $f(x) = \hat{f}$ for every x . We observe in experiments that correct triplet sampling strategies are crucial to avoid this phenomenon: taking large batches and enforcing the presence of as many different artists as possible in each batch prevents the system from collapsing.

In order to strengthen the artist representations learned we propose to make use of side information related to music artists. Suppose that we are given tags for artists of the training database. The fact that two artists have a same tag t , indicates that these artists share some characteristic. If we want our system to distinguish between similar but not equal artists, an interesting possibility is to train the system to not rely on these characteristics. To implement this idea, we define a probability p to prefer a negative sample x_- with the same tag as the anchor sample x_a when creating a triplet X . This is done at each iteration after the first sampling of the $Nn(n - 1)/2$ triplets. If for some anchor x_a there is no negative sample x_- with a different tag, or if we do not dispose of tags for the anchor sample, then we fallback to the previous triplet method creation, that is, we choose as negative any sample from another artist. This allows us to make use of side information even if some may be missing in the database, setting thus a flexible sampling framework.

4. EXPERIMENTS AND EVALUATION

In this section, we first present our main artist clustering task, the two auxiliary tasks that we use to compare to previous works and validate our metric learning approach, and the datasets used for evaluation. Then, we describe the architecture of the neural network that we use to learn artist representations. Finally, we detail the datasets used during the training of the systems.

4.1 Tasks and Evaluations

The systems studied in this paper output vector embeddings from audio excerpts. In order to perform evaluations, we create track-level embeddings by simply averaging embeddings over 10 linearly spaced segments of the same track. For the metric learning based system, we also project back the mean embeddings on the unit sphere.

4.1.1 Artist classification and verification

Dataset: evaluation is made over a dataset of 467 artists not seen during training of the embedding systems. These artists are taken from the MSD as explained in Section 4.3.1. For each artist we extract 20 tracks, 15 tracks

are used as ground truth to build artist models and we report the results for 5 tracks as test cases.

Classification task: we attribute to each test case the artist identity of its nearest neighbor for the euclidean distance among all the ground truth artist models in the embedding space, and we report the classification accuracy obtained with this procedure.

Verification task: this is a binary classification task were given any two track-level embeddings (e_i, e_j) of a test case i and artist model j , we decide whether they have the same artist membership. This is achieved by thresholding the euclidean distance between the two embeddings. We may do two types of errors in this task: a false positive error when two embeddings from two different artists are incorrectly classified as sharing the same membership, and a false negative error when two embeddings from the same artist are classified as having different memberships. The higher the decision threshold is, the higher the false negative rate (FNR). Respectively, lowering the threshold results in an increase of the false positive rate (FPR). We report the Equal Error Rate (EER), i.e.: the value of FPR and FNR for the threshold at which they are equal.

4.1.2 Homonym artist clustering

Dataset: we built an homonym artists database gathering artists which share exactly the same name with manual cleaning. This results in a database of 122 groups of 2 to 4 homonym artists (102 groups of 2 artists, 17 groups of 3 artists and 3 groups of 4). Each artist has a total number of 2 to 14 albums and 8 to 168 tracks.

Task: the problem to solve is to discriminate between artists that share the same name. From a set of tracks by different artists (with the same name), the task is to retrieve the actual clusters of tracks having the same artist membership. It is worth noting that this task may be used in a real life scenario since there is no need of previous knowledge of the identities nor the number of artists present in a group of same named artists.

We use the Adjusted Rand Index (ARI) [17] and the Adjusted Mutual Information (AMI) [24] to measure the similarity of the obtained clusters with the ground truth clusters given by the artist memberships. We choose these corrected for chance information theoretic measures since the data size is relatively small compared to the number of clusters present therein [24].

We present the results on a 5-fold cross-validation of the homonym artists datasets. We used an agglomerative hierarchical clustering algorithm [15] to group tracks from the same named artists. During the linkage phase we use *ward* method [26] with euclidean metric for calculating the distance between newly formed clusters, and we apply *distance* criterion to get the flat clusters. We select the optimal parameters based only on the AMI performance on the development set, given that we observe a strong correlation with the ARI performance. Finally we report the ARI and the AMI, averaged over the test dataset.

layer	size-out	kernel	activation
dyn. comp.	1x388x128	-	$\log(1 + 10^4 \cdot)$
conv	16x388x128	3x3x16,1	relu
maxpool	16x194x64	3x3x16,2	-
conv	32x194x64	3x3x32,1	relu
maxpool	32x64x21	3x3x32,2	-
conv	64x64x21	3x3x64,1	relu
maxpool	64x21x7	3x3x64,2	-
flatten	1x64x147	-	-
average pool	1x1x147	-	-
g dropout	1x1x147	-	-
dense	1x1xd	-	tanh
ℓ_2 const.	1x1xd	-	$\cdot \times \ \cdot\ _2^{-1}$

Table 1: Neural network map details for 9s long audio segments. The output size is described in stacks x rows x columns. The kernel is specified as rows x columns x nb. filters, stride. The parameter d is the size of the embedding.

4.2 Embedding systems

Our embedding map f consists in a convolutional deep neural network that takes features computed from either 3s or 9s long audio samples as input x and outputs points in $S^1 \subset \mathbb{R}^d$. The audio is down-sampled to 22050 kHz and we use a mel-spectrogram as features, with 128 mel-filters and 46 ms long Hann window with 50% of overlapping. We apply dynamic compression at the first layer of the network. Temporal pooling is performed inside the network after the convolution and maxpooling layers, by flattening together the stack and features tensor dimensions and then performing a global average pooling in the time dimension. The output of the network is a d -dimensional vector with $d = 32$ or $d = 256$. Further details of the structure of the network are described in Table 1. We use the RMSProp optimizer [23] with a learning rate of 10^{-3} , $\rho = 0.9$, $\epsilon = 10^{-8}$ and no decay. We compare our metric learning based system to the system in [16] since it is the only previous work that deals with unknown artists. This system computes artist embeddings of dimension 256 using the last hidden layer of a 1D convolutional neural network trained as an artist classifier. We refer to [16] for further details. Both systems were implemented using Keras [6] framework with Tensorflow [1] as back-end.

4.3 Datasets for training embedding systems

For each of the datasets described in this section, audio was provided by *Deezer* music streaming company. At the web address⁴ are provided the files containing the ids for the tracks as well as the artists names.

We build 3 datasets with different characteristics in terms of available audio data per artist and statistical distribution of tracks per artist. Our main goal is to test on the homonym artists clustering task the embedding systems obtained in different scenarios. One of the datasets is created to match real life catalog conditions.

⁴ <https://github.com/deezer/Disambiguating-Music-Artists-at-Scale-with-Audio-Metric-Learning>

4.3.1 MSD small dataset.

The first dataset is a sub-sampling of the MSD. The interest of this dataset is to compare the two studied systems when a small amount of audio data is available for each artist. Following [16] for the dataset construction, we use the *7digitalid* artist labels delivered with the MSD, but in addition we take care of cleaning the dataset from potential duplicate artists: we drop *MSD* artist labels associated with more than one *7digitalid* artist labels and viceversa (6.3% and 5.8% of artist labels). From this cleaned dataset we use the *7digitalid* labels to choose a number of artists between 100 and 2000, and then select 17 tracks for each artist. These are respectively split into 15 for training and 2 to perform early stopping. We extract a 30s-long excerpts for each track (the position of the excerpts were sampled at random between the beginning and the end of each track) that are further subdivided in 3s long segments with no overlapping used to feed the system. This results in 7.5 minutes of audio per artist in the training sets.

4.3.2 Balanced dataset.

The interest of this second dataset is to compare the two studied systems when access is granted to larger amounts of audio. We choose artists among the 1000 most popular in the music streaming service that provided the audio. We keep artist that have at least 4 albums with more than 3 tracks. From the remaining artists, we first choose a number of them between 25 and 600, and then select 1000 9s long samples for each artist linearly spaced in each track. From these samples we pick 1/3 for training and 1/6 to perform early stopping. The split is done at the album level, meaning that two tracks from the same album cannot appear in the same split. This results in 100 minutes of audio per artist for the training set. We remark that this dataset is balanced in terms of number of samples per artist, a common approach to prevent bias towards common classes in classifier systems. Also, the samples are taken from any region of the track, resulting in dataset with more variability than the *MSD small* dataset.

4.3.3 Unbalanced dataset.

In this third dataset we reproduce the statistical distribution of large music catalogs. We use a match from the Discogs [8] dataset onto the music streaming company artist dataset and we keep the genre tags. We drop artists with less than 4 tracks. We do not perform any further cleaning, so the resulting dataset is heavily unbalanced and presents typical long tail behavior. Distributions of samples by artist for this dataset are shown in Figure 1: the unbalanced dataset exhibit a long-tailed distribution.

From this dataset we select 10 9s long samples for each track, which are respectively split into train, evaluation and test. The split is done at the artist level, meaning that two tracks from the same artist cannot appear in the same split. There are 1749 different artists in the training set, each of one has between 1.5 and 45 minutes of audio.

The interest of this dataset is to study the abilities of each system to make use of all the available audio. Classifi-

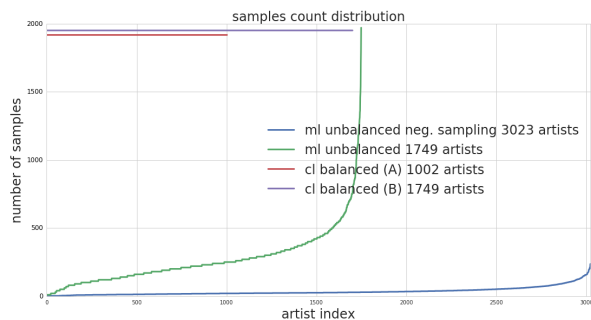


Figure 1: Count distribution of samples by artist in the *unbalanced side information negative sampling* and *unbalanced* datasets for metric learning, and the balanced versions for classifier systems of the *unbalanced* dataset.

cation systems are usually trained with a balanced dataset. If we have access to a dataset that is not already balanced in terms of classes, we have two options in order to balance it: (A) either cut down samples from the most represented classes or (B) repeat samples of the less represented ones. The former option implies losing data that could have potentially improved the training of the system, while in the second option there is a risk that the classification system over-fit the repeated samples. On the contrary, the triplet sampling of a metric learning system permits to make use of all accessible training data, since the system has the ability of dynamically choosing which data is more relevant for training. To study this, we train the metric learning system with the unbalanced dataset and the classifier system with two balanced versions of it as explained in here before. As we see in Figure 1, the (A) dataset contains 1002 artists and the (B) dataset contains 1749 artists, each with 2000 samples.

Finally, we study the influence of the new proposed negative sampling method using genre tags with a last *unbalanced side information negative sampling* dataset: from the raw match Discogs we retain 3023 artists, we keep the genre tags and we take only one 9s long sample for each track.

5. RESULTS

We first present in Figures 2 and 3 the results of the verification and classification tasks on the *MSD small* and *balanced* datasets. We observe that the classifier system performs better than the metric learning based system ($d = 256$) when few audio samples are available for each artist (*MSD small*), and that inversely, our system outperforms the classifier system when we are provided with larger amounts for each artist (*balanced*). Indeed, metric learning system are generally difficult to optimize, so large quantities of data are needed to make them learn correctly. When this data is accessible for each artist in train datasets, the metric learning system seems to learn better, which validates our approach.

In Figure 4 we present the results of the homonym artists clustering task. We first remark that both ARI and

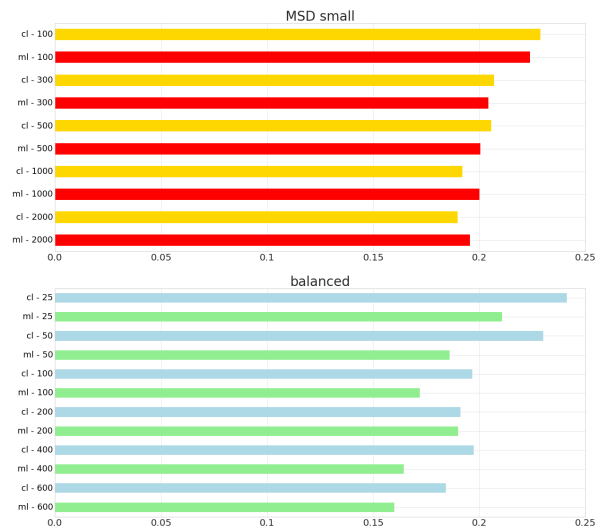


Figure 2: Equal Error Rate results of metric learning (ml - # artists) and classification (cl - # artists) embedding systems on the artist verification task for different training datasets and number of artists in the dataset. Lower is better.

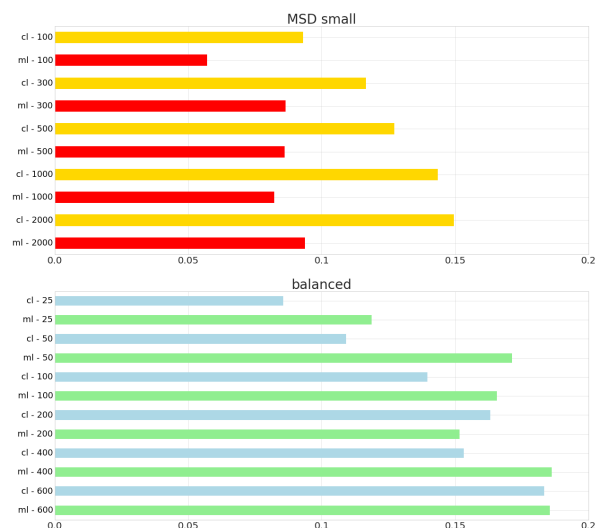


Figure 3: Accuracy results of metric learning (ml - # artists) and classification (cl - # artists) embedding systems on the artist verification task for different training datasets and number of artists in the dataset. Higher is better.

AMI measures take values between -1 and 1 , with 0 as expected value for a random clustering. The obtained results are thus satisfactory, showing the feasibility of the task and making it a compelling candidate to disambiguate unknown artists relying exclusively on audio, for large sized catalogs.

As we observed for the verification and classification tasks on the *MSD small* and *balanced* datasets, the metric learning system generally takes better advantage of larger training datasets. Moreover, the experiments with the *unbalanced* dataset and its balanced versions (A) and (B) in-

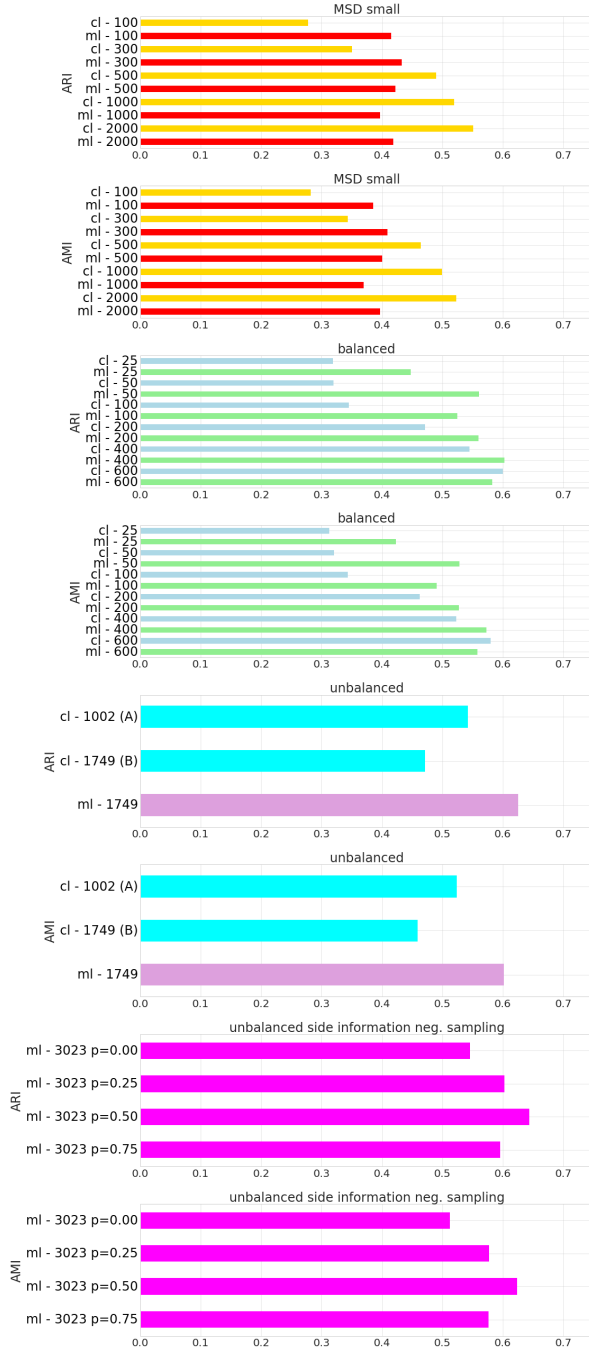


Figure 4: Mean AMI and mean ARI performances of the metric learning (ml - # artists) and classification (cl - # artists) embedding systems on the artist clustering task (5-fold cross-validation) for the different training datasets and number of artists in the dataset. Higher is better.

indicate that the metric learning system ($d = 32$) takes full advantage of all available data, at least when considering the balancing strategies that we proposed. This is of great interest since being able to use all the data in train databases could be beneficial in many other settings.

Finally, we study the results obtained with the side negative sampling strategy that use the genre tags in the *un-*

balanced side information negative sampling dataset. We experiment with setting the probability p as explained in Section 3.2 to different values. Although a linear hierarchy between different values of p is not completely observed, we remark that the best result is obtained with the probability $p = 0.5$. These promising results show the potential of using music side information to strengthen the learned artist representations.

6. CONCLUSION AND FUTURE WORK

6.1 Synthesis

We present a new task of unknown artists clustering to help disambiguating large scale catalogs, show the interest of it regarding the current problems of artists identification in the music industry, and demonstrate its feasibility with two different artist embeddings methods. Regarding different training datasets conditions (size, amount of audio available, distribution of tracks per artist) one or another could be of better help. We showed that the characteristics of artists learned by the system can generalize to other artists not seen during the learning phase. We prove that metric learning based method is an interesting choice for learning artist representations, in particular by the flexibility of the triplet loss mechanism that allows to better exploit available audio data or to incorporate music side information during training. To this extent, we proposed a new negative sampling method that takes advantage of side information during learning phase and show its relevance when using artist genre tags.

6.2 Future work

An interesting question for subsequent work is to understand the differences between the classification based artist representation and the metric learning based one, and if they learned similar high level characteristics of audio. To this extent a simple concatenation followed by a dimension reduction could be a first solution. More interestingly, we could try to train an embedding system with a linear combination of both losses. Since metric learning loss is difficult to optimize, for instance due to the collapsing problems, classification loss could act as a regularization term.

Another interesting research direction will be to explore other artist embeddings methods that also can incorporate side information, such as multi-task ranking losses from [28] or task driven non negative matrix factorization with group constraints as in [19].

Finally, we plan to further investigate the use of side information in negative sampling, and to explore the use of other kinds of sources such as mood or release date. Inversely, an interesting idea would be to investigate how to ameliorate genre representations learning by using artists labels as side information.

7. REFERENCES

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado,

- Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [2] Kamelia Aryafar and Ali Shokoufandeh. Multimodal music and lyrics fusion classifier for artist identification. In *ICMLA*, pages 506–509, 2014.
 - [3] A. Berenzweig, D. P. W. Ellis, and S. Lawrence. Anchor space for classification and similarity measurement of music. In *International Conference on Multimedia and Expo (ICME)*, volume 1, pages 1–29–32, 2003.
 - [4] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. *International Society for Music Information Retrieval Conference*, 2011.
 - [5] Hervé Bredin. TristouNet: Triplet Loss for Speaker Turn Embedding. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5430–5434, 2017.
 - [6] François Chollet et al. Keras. <https://github.com/fchollet/keras>, 2015.
 - [7] Sander Dieleman, Philémon Brakel, and Benjamin Schrauwen. Audio-based music classification with a pretrained convolutional network. In *ISMIR*, pages 669–674, 2011.
 - [8] Discogs. Monthly dumps of discogs release, artist, label, and master release data, 2018.
 - [9] Hamid Eghbal-zadeh, Bernhard Lehner, Markus Schedl, and Gerhard Widmer. I-vectors for timbre-based music similarity and music artist classification. In *ISMIR*, pages 554–560, 2015.
 - [10] Hamid Eghbal-Zadeh, Markus Schedl, and Gerhard Widmer. Timbral modeling for music artist recognition using i-vectors. In *European Signal Processing Conference (EUSIPCO)*, pages 1286–1290. IEEE, 2015.
 - [11] Hamid Eghbal-zadeh and Gerhard Widmer. Noise robust music artist recognition using i-vector. In *ISMIR*, pages 709–715, 2016.
 - [12] Aren Jansen, Manoj Plakal, Ratheet Pandya, Dan Ellis, Shawn Hershey, Jiayang Liu, Channing Moore, and Rif A. Saurous. Unsupervised learning of semantic audio representations. In *ICASSP*, 2018.
 - [13] Pavel P. Kuksa. Efficient multivariate sequence classification. *arXiv:1409.8211 [cs]*, 2014.
 - [14] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems*, pages 1096–1104. 2009.
 - [15] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *CoRR*, abs/1109.2378, 2011.
 - [16] Jiyoung Park, Jongpil Lee, Jangyeon Park, Jung-Woo Ha, and Juhan Nam. Representation learning of music using artist labels. *CoRR*, abs/1710.06648, 2017.
 - [17] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846, 1971.
 - [18] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823. IEEE Computer Society, 2015.
 - [19] Romain Serizel, Slim ESSID, and Gaël Richard. Group Non-negative Matrix Factorisation With Speaker And Session Similarity Constraints For Speaker Identification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Shanghai, China, March 2016.
 - [20] Sajad Shirali-Shahreza, Hassan Abolhassani, and M. Hassan Shirali-Shahreza. Fast and scalable system for automatic artist identification. *IEEE Transactions on Consumer Electronics*, 55(3), 2009.
 - [21] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.
 - [22] Li Su and Yi-Hsuan Yang. Sparse modeling for artist identification: Exploiting phase information and vocal separation. In *ISMIR*, pages 349–354, 2013.
 - [23] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.
 - [24] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of the American Statistical Association*, (301):236–244.
 - [25] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. 08 2017.
 - [26] Joe H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

- [27] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, 10:207–244, June 2009.
- [28] Jason Weston, Samy Bengio, and Philippe Hamel. Large-scale music annotation and retrieval: Learning to rank in joint semantic spaces. *CoRR*, abs/1105.5196, 2011.

DRIFTIN' DOWN THE SCALE: DYNAMIC TIME WARPING IN THE PRESENCE OF PITCH DRIFT AND TRANSPOSITIONS

Simon Waloschek, Aristotelis Hadjakos

Center of Music and Film Informatics, Detmold University of Music, Germany

{s.waloschek, a.hadjakos}@cemfi.de

ABSTRACT

Recordings of a cappella music often exhibit significant pitch drift. This drift may accumulate over time to a total transposition of several semitones, which renders the canonical 2-dimensional *Dynamic Time Warping* (DTW) useless. We propose *Transposition-Aware Dynamic Time Warping* (TA-DTW), an approach that introduces a 3rd dimension to DTW. Steps in this dimension represent changes in transposition. Paired with suitable input features, TA-DTW computes an optimal alignment path between a symbolic score and a corresponding audio recording in the presence of pitch drift or arbitrary transpositions.

1. INTRODUCTION

Existing audio-to-score alignment systems based on DTW are not yet able to handle performances appropriately if they exhibit significant pitch drift. However, pitch drift is rather common in a cappella singing and choir performances due to accumulation of intonation inaccuracies over time.

Absolute pitch, which is the ability to recognize and produce a given pitch without an external reference, is a rare ability of about 0.01% of the population [23]. Therefore, most singers have to rely on a combination of referencing with previous and simultaneous pitches together with muscle memory to intonate appropriately. In solo singing, the accuracy of a good singer has been reported to range from about 13 cents (standard deviation) [24] to about 22 cents [27] for very short melodies and intervals. Expert listeners judge a deviation of 20-25 cents to be still in tune [27]. The accuracy of note production can however be influenced adversely by various factors, e.g. by an unbalanced ratio of the sound pressure level of the reference sound in relation to the feedback sound of the singer's own voice [25]. Also the presence of vibrato, the absence of common partials between the voices and the absence of high partials make it more difficult to intonate correctly [24].

Due to the lack of an absolute reference, these minor de-

viations can lead to relatively large pitch deviations in the long run. It is widely reported that choirs have significant pitch drift, see [1]. Seaton et al. [20] surveyed amateur and professional choir singers and conductors regarding their experiences with pitch drift. Nearly half of the participants report that pitch drift occurs regularly while only 14% report that drift happens rarely or not at all. Nearly 80% of the participants say that the direction of the drift is usually downward while almost all other participants say that drift occurs in either direction similarly often.

However, pitch drift is not just an addition of small inaccuracies: Howard argues that pitch drift is almost inevitable when singing unaccompanied music that modulates from one key to another [12]. This arises mathematically from the observation that singers use non-tempered intonation based on the ratios of small integer numbers. Howard's measurements provide evidence that singers in fact use non-tempered intonation and that they consequentially shift their intonation as hypothesized. He even argues "that conductors who have a desire to correct overall intonation drift for its own sake in an a cappella performance [...] may be misguided" [12] if the piece contains considerable modulation.

This paper contributes a novel method called *Transposition-Aware Dynamic Time Warping* (TA-DTW) aiming at making an alignment between a symbolic score and a corresponding audio recording. TA-DTW is able to handle pitch drift (in contrast to a constant transposition), which makes it particularly useful to synchronize choir and singing recordings. Furthermore, it may be used as a drop-in replacement for existing solutions that can handle "only" fixed transpositions, which are commonly encountered in transcriptions of a piece for another instrument and historically informed performance practice.

The structure of this paper as follows: Section 2 describes the feature design, derived from Harmonic Pitch Class Profiles. Section 3 introduces TA-DTW as a 3-dimensional DTW based on the aforementioned features, followed by an evaluation, conclusions and future work in Sections 4 and 5. Related work is discussed in comparison to our approach within the individual sections.

2. ROBUST PITCH CLASS FEATURES IN THE PRESENCE OF PITCH DRIFT

Audio-to-Score Alignment algorithms that are based on DTW generally use pitch features such as chroma features [13, 15] as an intermediate data representation format



© Simon Waloschek, Aristotelis Hadjakos. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Simon Waloschek, Aristotelis Hadjakos. "Driftin' down the scale: Dynamic time warping in the presence of pitch drift and transpositions", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

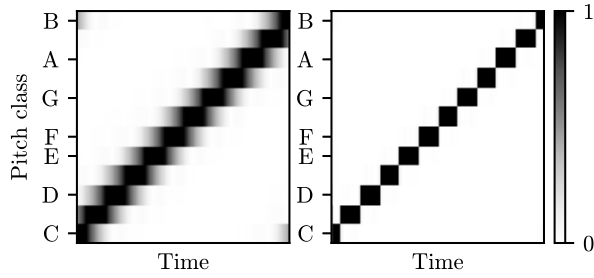


Figure 1. Chromagrams showing logarithmic sine sweeps from C4 to B4. *Left:* Canonical CQT chromas. *Right:* HPCPs with frame-wise tuning frequency estimation as used in this paper.

between the symbolic score and corresponding audio data. Chroma features are 12-dimensional vectors that describe, to a certain extent, tonality of a specific and usually very short sequence of music. They are obtained by measuring the relative intensity of each of the 12 pitch classes (C, C \sharp , D, ..., B) of the equal-tempered scale within an analysis frame. While this undoubtedly reduces the informational content in relation to tonal characteristics, this very reduction makes chroma features robust to changes in instrumentation and timbre. They still capture melodic and harmonic characteristics of music and thus provide a useful abstraction for various tasks within the music information retrieval research area.

For symbolic scores chroma features can be computed directly by mapping the pitch of the individual notes to their corresponding element in the chroma vector [13]. In case of audio data they are mostly computed using the *Fast Fourier Transform* (FFT) or the specialized *Constant Q Transform* (CQT) [3]. The latter is especially useful for western music since it allows for 1-to-1 mapping of filter bins to MIDI pitches. CQT can be expressed as a filter bank with fixed center frequencies for all filters, defined by a given reference pitch. In the presence of pitch drift, however, this reference pitch and thus the filter center frequencies have to be dynamically adapted to get high selectivity between adjacent semitones. Figure 1 (left) shows the effect of fixed center frequencies for a continuous sine sweep: The resulting chromagram appears to be “fuzzy” with leakage between semitones. For music that is more complex inaccurate center frequencies can lead to practically unusable chroma features.

2.1 Tuning Frequency Estimation

One way of dealing with these inaccuracies is the usage of multiple filter banks with slightly diverging reference pitch [18] and picking the best fit for each frame. A more general solution, however, is the use of tuning frequency estimation. Gnann et al. [8] proposed a real-time estimation algorithm, specifically addressing pitch drift in choir music. While their method of reducing the quadratic tuning deviation serves the purpose of having an active “pitch drift warning system” for rehearsals quite well, it does not allow for a time resolution down to a single analysis frame.

The same problem arises in an approach by Dressler [7] based on circular statistics: These methods calculate the tuning frequency iteratively resulting in an initial delay. Such behavior is unfavorable for DTW algorithms that rely on greatest feature accuracy possible in each frame. Both tuning estimation approaches were evaluated by Degani et al. [6] together with a third option that utilizes *Harmonic Pitch Class Profiles* (HPCP) [9, 29] and allows for the calculation of the deviation from reference pitch within a single analysis frame [11]. As all three tuning estimation methods demonstrated similar performance, we will focus on HPCPs and their superior time resolution.

HPCPs are closely related to chroma features but differ in one important aspect: They are tuning independent by definition, so that the reference frequency is not explicitly defined. The result of HPCP computation is an octave-independent histogram with 12, 24, 36, or even more bins, depending on the needed frequency resolution as shown in Figure 2. For a constant quality spectrum C with N bins in total and 36 per octave, the value of the k -th bin of a 36-bin HPCP H is calculated by

$$H_k = \sum_{n=0}^{N/36} |C_{k+36n}| \quad \forall k \in [1 : 36]. \quad (1)$$

In order to estimate the tuning deviation, each HPCP frame is processed with a peak detection algorithm. Multiple peaks might be found in such a frame and the global deviation from an assumed reference pitch can be averaged over the individual deviations of the peaks.

In this paper we decided to use quadratic interpolation as described by Smith in [22] with 36-bin HPCPs. However, we do not look for the peaks explicitly but rather accumulate the magnitudes of each semitone’s 3 corresponding bins:

$$m_k = \sum_{n=0}^{11} H_{3n+k} \quad \forall k \in \{1, 2, 3\}. \quad (2)$$

We assume that m_2 is the highest of these values, otherwise we would have shifted the values of m cyclically by a value $s \in \{-1, 0, 1\}$. To be consistent with [22] and increase readability, we define $\alpha = m_1$, $\beta = m_2$, and $\gamma = m_3$. Next, we fit a parabola to these magnitudes, i.e. through

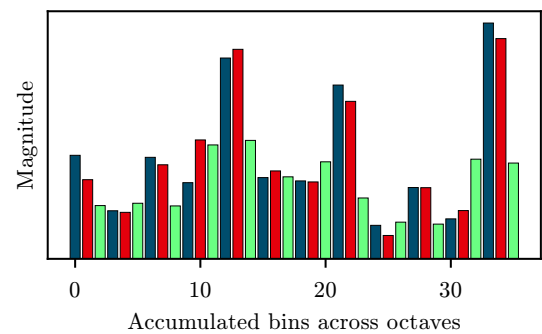


Figure 2. Harmonic Pitch Class Profile with 36 bins per octave for a single analysis frame.

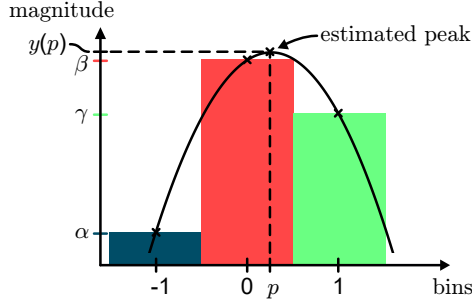


Figure 3. Parabola that is fitted to the bins’ magnitudes α , β , and γ .

$(-1, \alpha)$, $(0, \beta)$, and $(1, \gamma)$ as shown in Figure 3. (The bins have been arbitrarily renumbered about the estimated peak that is represented by the parabola’s vertex.)

Looking at the general formula for a parabola

$$y(x) = a(x - p)^2 + b \quad (3)$$

we can directly tell the location of the vertex: The center point p gives us the error offset of our actual pitch in bins, while the amplitude $y(p) = b$ equals the peak amplitude. All three magnitudes can be calculated as follows:

$$\begin{aligned} \alpha &= ap^2 + 2ap + a + b, \\ \beta &= ap^2 + b, \\ \gamma &= ap^2 - 2ap + a + b \end{aligned} \quad (4)$$

The peak location in (fractional) bins is given by

$$p = \frac{1}{2} \frac{\alpha - \gamma}{\alpha - 2\beta + \gamma} \in \left[-\frac{1}{2}, \frac{1}{2}\right] \quad (5)$$

and the estimated peak magnitude gets calculated by:

$$y(p) = \beta - \frac{1}{4}(\alpha - \gamma)p. \quad (6)$$

2.2 Feature Computation

Knowing the global tuning deviation p , we can calculate the estimated peak magnitude for every pitch class (from its 3 bins) of our HPCP H via Equation 6 with s being the potential cyclic shifting done after Equation 2 and

$$\begin{aligned} \alpha &= H_{3k-2-s} \\ \beta &= H_{3k-1-s} \quad \forall k \in [1 : 12]. \\ \gamma &= H_{3k-s} \end{aligned} \quad (7)$$

This step effectively reduces the 36 bins per octave to 12 bins per octave, which makes the resulting HPCPs comparable to standard chroma features. To decrease differences in dynamics between the features, each HPCP vector is normalized to have length 1. We obtain a chromagram as exemplary shown in Figure 1 (right) by repeating this for the entire audio recording.

If the estimated tuning is off by approximately -0.5 or $+0.5$ bins and the actual tuning of the recording fluctuates, the resulting features can be off by 1 semitone in either direction. This can be considered a local “unintended transposition” and will be handled in the next section.

3. TRANSPOSITION-AWARE DYNAMIC TIME WARPING

Support for changing transpositions over time is very limited in current alignment systems. In this context the term *transposition* covers intended alterations in pitch, e.g. “baroque pitch” or simply singing in a different key, as well as unintended pitch drift that exceeds the scope of a semitone. Most alignment systems, such as *Antescofo* [4] pass the problem of unknown transpositions on to the user and force them to adapt their symbolic score accordingly by themselves. Niedermayer [19] solves this step by computing all possible transpositions and picking the best fit. This is similar to a work by Müller [16] that determines the best transpositions for each individual pitch feature, though the results are not used for audio-to-score alignment. All these systems assume that the global tuning does not change over time and has to be estimated only once at the beginning, except Arzt [2]: He uses fingerprinting to determine a musical piece, the current position within that piece, and its transposition before the actual alignment. Hence, the system is theoretically able to “recover” from unforeseen pitch changes after some time but is (for now) restricted to piano music and does not allow for continuous alignment in such cases.

We propose an extended version of DTW called *Transposition-Aware Dynamic Time Warping* (TA-DTW) that allows for continuous changes in transposition during alignment. It shares conceptual ideas with the multidimensional DTW presented in [28] but focuses on special properties of chroma features and the nature of musical transpositions. The remainder of this section presupposes basic knowledge of the original DTW algorithm. A comprehensive overview can be found in [15].

3.1 Distance Calculation & Transpositions

In order to compute the alignment we need the distances between the vectors from the score and the audio HPCP vectors as computed in Section 2. Various metrics have been used to calculate these distances such as the Euclidean distance (2-norm distance) [13, 15] and Manhattan distance (1-norm distance) [2, 15]. For computational complexity reasons, however, we opted for the *Cosine Distance*¹ which is defined for two nonzero vectors \mathbf{a} and \mathbf{b} as

$$c(\mathbf{a}, \mathbf{b}) = 1 - \cos(\theta) = 1 - \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|_2 \cdot \|\mathbf{b}\|_2} \quad (8)$$

and represents the angular distance ranging from 0 (equal orientation) to 2 (diametrically opposed). Taking into account that our HPCP features already have the length 1, the denominator can be reduced to 1 and leaves us with

$$c(\mathbf{a}, \mathbf{b}) = 1 - \mathbf{a} \cdot \mathbf{b} = 1 - \sum_{i=1}^n a_i \cdot b_i. \quad (9)$$

This operation can be extended to two matrices with the same number of rows and unit length columns. It results in

¹ As the cosine distance does not obey the triangle inequality it is strictly speaking not a proper distance metric, see [21]. Nevertheless, it can be used as such in this particular context.

a matrix that contains distances between all combinations of column vectors of these matrices ($\mathbb{1}$ denotes the $N \times M$ matrix of ones):

$$c(\mathbf{A}, \mathbf{B}) = \mathbb{1} - \mathbf{A}^T \mathbf{B} \quad (10)$$

To make use of this in our context we will represent our sequence of HPCP vectors $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_N$ from the score as a matrix²:

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,N} \\ a_{2,1} & a_{2,2} & \dots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{12,1} & a_{12,2} & \dots & a_{12,N} \end{bmatrix} \quad (11)$$

The same applies to the sequence of HPCP vectors $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_M]$ from the audio recording.

Cyclically shifting the elements of a pitch class vector by a value t equals transposing by t semitones as pointed out by Goto [10]. We make use of this property and compute all 11 possible transpositions $t \in [1 : 11]$ for the score HPCP matrix \mathbf{A} . Shifting the rows of \mathbf{A} can be done by multiplying the cyclic permutation matrix $\mathbf{P} \in \mathbb{R}^{12 \times 12}$ with \mathbf{A} :

$$\mathbf{A}_t = (\mathbf{P}^t) \mathbf{A}, \mathbf{P} = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix} \quad (12)$$

Finally, we calculate all distances for all transpositions by

$$c(\mathbf{A}, \mathbf{B}, t) = \mathbb{1} - (\mathbf{A}_t)^T \mathbf{B}. \quad (13)$$

3.2 Accumulated Multidimensional Cost Matrix

Throughout this section we will express the results of Equation 13 as the cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M \times 12}$.

With \mathbf{C} we can compute the accumulated cost matrix $\mathbf{D} \in \mathbb{R}^{N \times M \times 12}$ by means of dynamic programming. Additionally to steps in the $n \times m$ plane, as performed in the canonical DTW, steps in the transposition dimension t need to be considered. Moving a semitone cyclically downwards and upwards along the $t \in [0 : 11]$ axis will be defined as

$$\begin{aligned} t_- &= (t - 1) \bmod 12 \\ t_+ &= (t + 1) \bmod 12 \end{aligned} \quad (14)$$

which allows arbitrary transposition changes that may even exceed one octave. We will restrict the possible transposition changes between two adjacent analysis windows to one semitone in order to keep the underlying math concise in this paper. While this seems reasonable in practice, too, it is not an inherent restriction of the algorithm. Figure 4 visualizes the resulting valid steps inside the accumulated cost matrix \mathbf{D} for a possible alignment path.

² Music analysis frameworks such as *librosa* or *madmom* already use such a representation anyway.

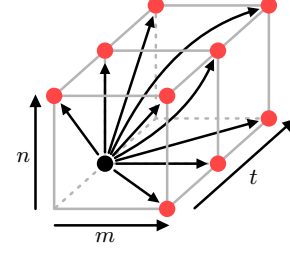


Figure 4. Valid steps inside the accumulated multidimensional cost matrix \mathbf{D} .

The accumulated multidimensional cost matrix will be calculated as follows:

$$\mathbf{D}_{n,m,t} = \begin{cases} \sum_{k=1}^m \mathbf{C}_{1,k,t} & \text{if } n = 1 \\ \sum_{k=1}^n \mathbf{C}_{k,1,t} & \text{if } m = 1 \\ \min(\text{steps}) + w(\Delta t) \mathbf{C}_{n,m,t} & \text{otherwise} \end{cases} \quad (15)$$

The (recursive) *steps* for computing \mathbf{D} are defined as

$$\text{steps} = \left\{ \begin{array}{l} \mathbf{D}_{n, \quad m-1, \quad t} \\ \mathbf{D}_{n-1, \quad m, \quad t} \\ \mathbf{D}_{n-1, \quad m-1, \quad t} \\ \mathbf{D}_{n, \quad m-1, \quad t_-} \\ \mathbf{D}_{n-1, \quad m, \quad t_-} \\ \mathbf{D}_{n-1, \quad m-1, \quad t_-} \\ \mathbf{D}_{n, \quad m-1, \quad t_+} \\ \mathbf{D}_{n-1, \quad m, \quad t_+} \\ \mathbf{D}_{n-1, \quad m-1, \quad t_+} \end{array} \right\}. \quad (16)$$

Steps along the t -axis alone are not allowed since it is impossible to calculate \mathbf{D} under these conditions. The factor $w(\Delta t)$ is a weighting factor for penalizing relative movements along the t axis. An increased weight has shown to stabilize the algorithm by reducing accidental transposition changes for special cases, e.g. monophonic passages, where regular pitch changes might look equivalent to transpositions.

3.3 Backtracking

In order to compute the warping path \mathbf{p} we use a backtracking algorithm. The starting point \mathbf{p}_L for the recursive computation is the point along the (N, M, t) -axis in \mathbf{D} with the least costs:

$$\begin{aligned} T &= \arg \min_t (\mathbf{D}_{N,M,t}) \\ \mathbf{p}_L &= (N, M, T). \end{aligned} \quad (17)$$

Feature	Algorithm	Drift	Percentage of events with absolute misalignment error						
			$\leq 0.15s$	$\leq 0.20s$	$\leq 0.25s$	$\leq 0.30s$	$\leq 0.40s$	$\leq 0.50s$	$\leq 1.00s$
Chroma	DTW		83.46%	87.75%	89.72%	90.87%	92.04%	92.67%	93.99%
HPCP	DTW		86.28%	91.09%	93.23%	94.39%	95.75%	96.36%	97.64%
HPCP	TA-DTW		86.52%	91.69%	93.96%	95.18%	96.40%	96.92%	97.89%
Chroma	DTW	✓	20.51%	23.00%	24.61%	25.98%	28.35%	30.23%	36.53%
HPCP	TA-DTW	✓	79.89%	88.35%	92.09%	93.97%	95.56%	96.28%	97.31%

Table 1. Results of the audio-to-score alignment evaluation. Best results are emphasized.

Now we move recursively through the accumulated cost matrix:

$$p_{\ell-1} = \begin{cases} \arg \min \begin{pmatrix} D_{1,m-1,t} \\ D_{1,m-1,t-} \\ D_{1,m-1,t+} \end{pmatrix} & \text{if } n = 1 \text{ and } m \geq 2 \\ \arg \min \begin{pmatrix} D_{n-1,1,t} \\ D_{n-1,1,t-} \\ D_{n-1,1,t+} \end{pmatrix} & \text{if } m = 1 \text{ and } n \geq 2 \\ \arg \min(\text{steps}) & \text{if } n, m \geq 2 \end{cases} \quad (18)$$

The resulting 3-dimensional warping path can be orthogonally projected onto different planes as shown in Figure 5:

$n \times m$ corresponds to the final alignment between the input feature matrices **A** and **B**.

$m \times t$ gives information about the location of transposition changes in the audio data in relation to the score. The “local unintended transpositions” as outlined in Section 2.2 are clearly visible.

$n \times t$ shows accordingly these transposition change positions in the score.

The transposition changes in the planes $n \times t$ and $m \times t$ can be refined by adding the center frequency offsets p as calculated in Section 2 for each audio HPCP vector. This allows for computing continuous pitch drift data.

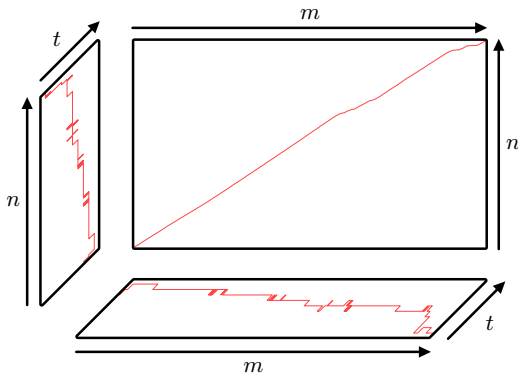


Figure 5. Orthogonal projections of the 3-dimensional warping path onto the $n \times t$, $m \times t$, and $n \times m$ plane.

4. EVALUATION

We evaluated the alignment accuracy of our TA-DTW with HPCPs in comparison with the canonical DTW and plain CQT-based chroma features. Due to the lack of datasets with choral music and corresponding beat-level annotations, we generated the evaluation data from the complete MusicNet dataset [26]. It consists of almost 1.3 million note events (that were manually verified by expert annotators) for approximately 34 hours of chamber music performances with various instrumentations. All material is available in 44.1 kHz sampling rate. Although a cappella music is not part of the dataset, we considered it meaningful for evaluation due to its substantial scope. Since the pieces of the dataset present no significant pitch drift, we extracted all recordings to raw PCM files and introduced continuously changing random artificial pitch drift.

This was done by loading each of the 330 pieces into a *Digital Audio Workstation* (DAW) and generating 100 equidistantly distributed random pitch change markers along the time axis for each piece. The amount of introduced pitch drift was kept within the range of ± 4 semitones and followed brownian motion to introduce correlation with previous markers. Between the markers, the pitch deviation was linearly interpolated. A randomization as such is a reasonably realistic simulation according to the pitch drift model for a cappella music of Mauch et al. in [14].

Based on the evaluation methodology of Cont et al. in [5] for audio-to-score alignment, the absolute alignment errors in seconds for note onsets were calculated. 1024 samples per window and no overlap for the audio data were used. This equals a feature rate of ~ 43 vectors per second or a window length of approximately 23ms. The results are shown in Table 1. We found $w(\Delta t) = 6.5$ to be a suitable penalty factor for changes along the t -axis in **D**, see Equation 15.

In the absence of any pitch drift, we found that using HPCPs showed superior performance in contrast to plain chroma features, regardless of whether the calculation of the alignment was done using DTW or our proposed TA-DTW. This can be explained by slight deviations in tuning frequency of the recordings that are compensated in the computation of HPCPs. Using them as calculated in this paper introduces occasional errors in the form of “local transpositions” as explained in Section 2.2. Such errors can be minimized by switching from DTW to TA-DTW, which further improves the alignment.

For music with drifting pitch our proposed method shows comparable results while the “classic” approach failed for the majority of note onsets. We assumed that the remaining $\sim 30\%$ are based on the limited maximum pitch drift in our test data, resulting in this amount of data effectively being not or only marginally pitched. This hypothesis was validated by exemplarily computing the alignment with plain DTW using audio files that exhibit constant pitch drift >1 semitone. These cases showed results $<1\%$ for all shown time intervals.

The drawbacks of this approach are the increased memory and computation requirements. TA-DTW requires a cost matrix of dimension $N \times M \times 12$, which is 12 times larger compared to the 2-dimensional cost matrix for DTW. Computing the warping path with TA-DTW involves computing $N \times M \times 12 \times 9$ path scores. This is 36 times greater in contrast to $M \times N \times 3$ in DTW. We have empirically verified these numbers. For music recordings with a length that exceeds several minutes, the algorithm demands well-equipped hardware in terms of memory.

5. CONCLUSIONS & FUTURE WORK

In this paper we presented a DTW-based method to compute audio-to-score alignment for audio data that suffers from drift in global pitch throughout the recording. To this end we explained the computation of suitable pitch features that allow for “sharper” distinction between adjacent notes on the pitch scale. We used these features in conjunction with a DTW algorithm that we extended to support static and dynamically changing transpositions. A final evaluation proved the robustness and effectiveness of our approach.

Apart from normalizing our pitch features to have length 1, we did not process them any further. This ensures that they can be used as basis for additional feature enhancements [17]. Similarly, this applies to our DTW extension: Since the *Transposition-Aware DTW* is conceptually very close to the original DTW algorithm, many variations and improvements such as varying step size conditions, local weights, or global constraints [15] can be adapted easily.

Potential on-line variants of TA-DTW could greatly reduce the computational complexity by only calculating cost values for $t \pm 1$ for the current audio window.

6. REFERENCES

- [1] Per-Gunnar Alldahl. *Choral intonation*. Gehrman Musikförlag, 2008.
- [2] Andreas Arzt. *Flexible and Robust Music Tracking*. PhD thesis, Johannes Kepler University, Linz, Austria, 2016.
- [3] Judith C Brown. Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.
- [4] Arshia Cont. Antescofo: Anticipatory synchronization and control of interactive parameters in computer music. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 33–40, 2008.
- [5] Arshia Cont, Diemo Schwarz, Norbert Schnell, and Christopher Raphael. Evaluation of real-time audio-to-score alignment. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, 2007.
- [6] Alessio Degani, Marco Dalai, Riccardo Leonardi, and Pierangelo Migliorati. Comparison of tuning frequency estimation methods. *Multimedia Tools and Applications*, 74(15):5917–5934, 2015.
- [7] Karin Dressler and Sebastian Streich. Tuning frequency estimation using circular statistics. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, 2007.
- [8] Volker Gnann, Markus Kitz, Julian Becker, and Martin Spiertz. Least-squares local tuning frequency estimation for choir music. In *Audio Engineering Society Convention 131*. Audio Engineering Society, 2011.
- [9] Emilia Gómez. *Tonal description of music audio signals*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2006.
- [10] Masataka Goto. A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1783–1794, 2006.
- [11] Christopher Harte and Mark Sandler. Automatic chord recognition using quantised chroma and harmonic change segmentation. *Centre for Digital Music, Queen Mary University of London*, 2009.
- [12] David M Howard. Intonation drift in a capella soprano, alto, tenor, bass quartet singing with key modulation. *Journal of Voice*, 21(3):300–315, 2007.
- [13] Ning Hu, Roger B Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 185–188, 2003.
- [14] Matthias Mauch, Klaus Frieler, and Simon Dixon. Intonation in unaccompanied singing: Accuracy, drift, and a model of reference pitch memory. *The Journal of the Acoustical Society of America*, 136(1):401–411, 2014.
- [15] Meinard Müller. *Information retrieval for music and motion*, volume 2. Springer, 2007.
- [16] Meinard Müller and Michael Clausen. Transposition-invariant self-similarity matrices. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, 2007.

- [17] Meinard Müller and Sebastian Ewert. Chroma toolbox: Matlab implementations for extracting variants of chroma-based audio features. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011.
- [18] Meinard Müller, Peter Grosche, and Frans Wiering. Automated analysis of performance variations in folk song recordings. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010.
- [19] Bernhard Niedermayer. *Accurate audio-to-score alignment: data acquisition in the context of computational musicology*. PhD thesis, Johannes Kepler University, Linz, Austria, 2012.
- [20] Richard Seaton, Dennis Pim, and David Sharp. Pitch drift in a cappella choral singing – work in progress. In *Proceedings of the Institute for Acoustics Annual Spring Conference*, volume 35, 2013.
- [21] John R Smith. *Integrated spatial and feature image systems: Retrieval, analysis and compression*. PhD thesis, Columbia University, New York, USA, 1997.
- [22] Julius O. Smith. *Spectral audio signal processing*. W3K publishing, 2011.
- [23] Annie H Takeuchi and Stewart H Hulse. Absolute pitch. *Psychological bulletin*, 113(2):345–361, 1993.
- [24] Sten Ternström and Johan Sundberg. Acoustical factors related to pitch precision in choir singing. *Speech Transmission Laboratory Quarterly Progress and Status Report (STL-QPSR)*, 2(3):1982, 1982.
- [25] Sten Ternström and Johan Sundberg. Intonation precision of choir singers. *The Journal of the Acoustical Society of America*, 84(1):59–69, 1988.
- [26] John Thickstun, Zaid Harchaoui, and Sham Kakade. Learning features of music from scratch. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
- [27] Allan Vurma and Jaan Ross. Production and perception of musical intervals. *Music Perception: An Interdisciplinary Journal*, 23(4):331–344, 2006.
- [28] Martin Wöllmer, Marc Al-Hames, Florian Eyben, Björn Schuller, and Gerhard Rigoll. A multidimensional dynamic time warping algorithm for efficient multimodal fusion of asynchronous data streams. *Neurocomputing*, 73(1-3):366–380, 2009.
- [29] Yongwei Zhu, Mohan S Kankanhalli, and Sheng Gao. Music key detection for musical audio. In *Proceedings of the 11th International Multimedia Modelling Conference (MMM)*, 2005.

END-TO-END LEARNING FOR MUSIC AUDIO TAGGING AT SCALE

Jordi Pons^{*†} Oriol Nieto[†] Matthew Prockup[†] Erik Schmidt[†] Andreas Ehmann[†] Xavier Serra^{*}

^{*} Music Technology Group, Universitat Pompeu Fabra, Barcelona

[†] Pandora Media Inc., Oakland, CA

ABSTRACT

The lack of data tends to limit the outcomes of deep learning research, particularly when dealing with end-to-end learning stacks processing raw data such as waveforms. In this study, 1.2M tracks annotated with musical labels are available to train our end-to-end models. This large amount of data allows us to unrestrictedly explore two different design paradigms for music auto-tagging: assumption-free models – using waveforms as input with very small convolutional filters; and models that rely on domain knowledge – log-mel spectrograms with a convolutional neural network designed to learn timbral and temporal features. Our work focuses on studying how these two types of deep architectures perform when datasets of variable size are available for training: the MagnaTagATune (25k songs), the Million Song Dataset (240k songs), and a private dataset of 1.2M songs. Our experiments suggest that music domain assumptions are relevant when not enough training data are available, thus showing how waveform-based models outperform spectrogram-based ones in large-scale data scenarios.

1. INTRODUCTION

One fundamental goal in music informatics research is to automatically structure large music collections. The music audio tagging task consists of automatically estimating the musical attributes of a song – including: moods, language of the lyrics, year of composition, genres, instruments, harmony, or rhythmic traits. Thus, tag estimates may be useful to define a semantic space that can be advantageous for automatically organizing musical libraries.

Many approaches have been considered for this task (mostly based on feature extraction + model [1, 22, 26]), with recent publications showing promising results using deep architectures [5, 9, 14, 21]. In this work we confirm this trend by studying how two deep architectures conceived considering opposite design strategies (using domain knowledge or not) perform for several datasets – with one of the datasets being of an unprecedented size: 1.2M songs. Provided that a sizable amount of data is available for that study, we investigate the learning capabilities

of these two architectures. Specifically, we investigate whether the architectures based on domain knowledge overly constrain the solution space for cases where large training data are available – in essence, we study if certain architectural choices (e.g., using log-mel spectrograms as input) can limit the model’s capabilities to learn from data. The main contribution of this work is to show that little to no model assumptions are required for music auto-tagging when operating with large amounts of data.

Section 2 discusses the main deep architectures we identified in the audio literature, section 3 describes the datasets used for this work, section 4 presents the architectures we study, and section 5 provides discussion about the results with conclusions drawn in section 6.

2. CURRENT DEEP ARCHITECTURES

In order to facilitate the discussion around the current audio architectures, we divide deep learning models into two parts: front-end and back-end – see Figure 1. The front-end is the part of the model that interacts with the input signal in order to map it into a latent-space, and the back-end predicts the output given the representation obtained by the front-end. In the following, we present the main front- and back-ends we identified in the literature.

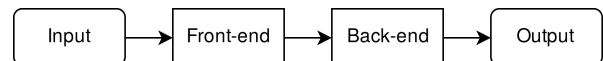


Figure 1. Deep learning pipeline.

Front-ends. These are generally comprised of convolutional neural networks (CNNs) [5, 9, 20, 21, 27], since these can learn efficient representations by sharing weights¹ along the signal. Front-ends can be divided into two groups depending on the used input signal: waveforms [9, 14, 27] or spectrograms [5, 20, 21]. Further, the design of the filters can be either based on domain knowledge or not. For example, one leverages domain knowledge when a front-end for waveforms is designed so that the length of the filter is set to be as the window length of a STFT [9]. Or for a spectrogram front-end, it is used vertical filters to learn timbral representations [12] or horizontal filters to learn longer temporal cues [25]. Generally, a single filter shape is used in the first CNN layer [5, 9, 12, 25], but some recent works report performance gains when using several filter shapes in the first layer [4, 18–21, 27]. Using many filters promotes a richer feature extraction in the first layer, and facilitates leveraging domain knowledge for designing the filters’ shape. For example: a

¹ Which determine the learned feature representations.



waveform front-end using many long filters (of different lengths) can be motivated from the perspective of a multi-resolution time-frequency transform² [27]; or since it is known that some patterns in spectrograms are occurring at different time-frequency scales, one can intuitively incorporate many (different) vertical and/or horizontal filters in a spectrogram front-end [18–21]. To summarize, using domain knowledge when designing models allows us to naturally connect the deep learning literature with previous signal processing work. On the other hand, when domain knowledge is not used, it is common to employ a deep stack of small filters, e.g.: 3×1 as in the sample-level front-end used for waveforms [14], or 3×3 filters used for spectrograms [5]. These models based on small filters make minimal assumptions over the local stationarities of the signal, so that any structure can be learned via hierarchically combining small-context representations. These architectures with small filters are flexible models able to potentially learn any structure given enough depth and data.

Back-ends. Among the different back-ends used in the audio literature, we identified two main groups: (i) fixed-length input back-end, and (ii) variable-length input back-end. The generally convolutional nature of the front-end allows it to process different input lengths. Therefore, the back-end unit can adapt a variable-length feature map to a fix-sized output. The former group of models (i) assume that the input will be kept constant – examples of those are front-ends based on feed-forward neural-networks or fully-convolutional stacks [5, 9]. The second group (ii) can deal with different input-lengths since the model is flexible in at least one of its input dimensions – examples of those are back-ends using temporal-aggregation strategies such as max-pooling, average-pooling, attention models or recurrent neural networks [23]. Given that songs are generally of different lengths, these types of back-ends are ideal candidates for music processing. However, despite the different-length nature of music, many works employ fixed-length input back-ends (group i) since these architectures tend to be simpler and perform well [5, 9, 21].

3. DATASETS

We study how different deep architectures for music auto-tagging perform for 3 music collections of different sizes:

1) The MagnaTagATune (MTT) dataset is of $\approx 26k$ music audio clips of 30s [11]. Predicting the top-50 tags of this dataset is a popular benchmark for auto-tagging.

2) Although the Million Song Dataset (MSD) name indicates that 1M songs are available [2], audio files with proper tag annotations (top-50 tags) are only available for $\approx 240k$ previews of 30s. This dataset constitutes the biggest public dataset available for music auto-tagging, making these data highly appropriate for benchmarking.

3) A private dataset consisting of 1M songs for training, 100k for validation, and 100k for test³ is available for this study. The 1.2M-songs dataset has 139 track-level human-expert annotations that can be summarized as follows:

- *Meter tags* denote different sorts of musical meters (e.g., triple-meter, cut-time, compound-duple, odd).
- *Rhythmic feel tags* denote rhythmic interpretation (e.g., swing, shuffle, back-beat strength) and elements of rhythmic perception (e.g., syncopation, danceability).
- *Harmonic tags*: major, minor, chromatic, etc.
- *Mood tags* express the sentiment of a music audio clip (e.g., if the music is angry, sad, joyful).
- *Vocal tags* denote the presence of vocals and timbral characteristics of it (e.g., male, female, vocal grittiness).
- *Instrumentation tags* denote the presence of instruments (e.g., piano) and their timbre (e.g., guitar distortion).
- *Sonority tags* detail production techniques (e.g., studio, live) and overall sound (e.g., acoustic, synthesized).
- *Basic genre tags*: jazz, rock, rap, latin, disco, etc.
- *Subgenre tags*: jazz (e.g., cool, fusion, hard bop), rock (e.g., light, hard, punk), rap (e.g., east coast, old school), world music (e.g., cajun, indian), classical music (e.g., baroque period, classical period), etc.

Other large (music) audio datasets exist: the Free Music Archive (FMA: $\approx 106k$ songs) [8] and Audioset ($\approx 2.1M$ audios) [10]. Since previous works mainly used the MTT and MSD [5, 14], we employ these datasets to assess the studied models with public data. Despite our interest in using FMA, for brevity, we restrict our study to 3 datasets that already cover a wide range of different sizes. Finally, Audioset is not used since most of its content is not music.

4. THE ARCHITECTURES UNDER STUDY

After an initial exploration of the different architectures introduced in section 2, we select two models based on opposite design paradigms: one for processing waveforms, with a design that does minimal assumptions over the task at hand; and another for spectrograms, with a design that heavily relies on musical domain knowledge. Our goal is to compare these two models for providing insights in whether domain knowledge is required (or not) for designing deep learning models. This section provides discussion around our architectural choices and introduces the *basic* configuration setup – which is also accessible online.⁴

The waveform model was selected after observing that the sample-level front-end (using a deep stack of 3×1 filters) was remarkably superior to the other waveform-based front ends – as shown in the original paper [14]. This result is particularly compelling because this front-end does not rely on domain-knowledge for its design. Note that raw waveforms are fed to the model without any pre-processing, and the small filters considered for its design make no strong assumptions over the most informative local stationarities in waveforms. Therefore, the sample-level can be seen as a problem agnostic front-end that has the potential to learn any audio task provided that enough depth and data are available. Given that a large amount data is available for this study, the sample-level front-end is of particular interest due to its strong learning potential: its solution space is not constrained by severe architectural choices relying on domain knowledge.

² The Constant-Q Transform [3] is an example of such transform.

³ Test & validation sets are kept the same throughout the experiments for a fair evaluation. All used partitions are stratified and artist-filtered.

⁴ <https://github.com/jordipons/music-audio-tagging-at-scale-models>

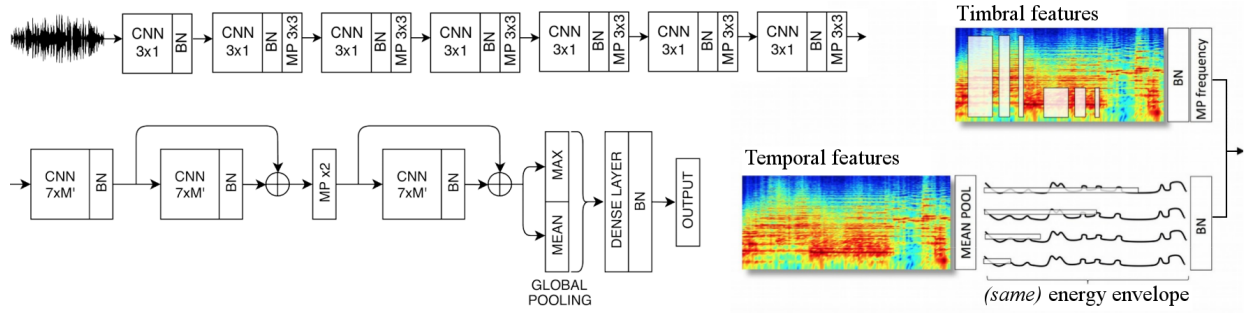


Figure 2. *Bottom-left* – back-end. *Top-left* – waveform front-end. *Right* – spectrogram front-end. *Definitions* – M' stands for the feature map’s vertical axis, BN for batch norm, and MP for max-pool.

On the other hand, when experimenting with spectrogram front-ends, we found domain knowledge intuitions to be valid guides for designing deep architectures. For example, front-ends based on (i) many vertical and horizontal filters in the first layer were consistently superior to front-ends based on (ii) a single vertical filter – as shown in recent publications [4, 18–20]. Note that the former front-ends (i) can learn spectral and (long) temporal representations already in the first layer – which are known to be important musical cues; while the latter (ii) can only learn spectral representations. Moreover, we observed that front-ends based on a deep stack of 3×3 filters were achieving equivalent performances to the former front-end (i) when input segments were shorter than 10s – as noted in the literature [21]. But when considering longer inputs (which yielded better performance), the computational price of this deeper model increases: longer inputs implies having larger feature maps in every layer and therefore, more GPU memory consumption. For that reason, we refrained from using a deep stack of 3×3 filters as a front-end – because our 12GBs of VRAM were not enough to input 15s of audio when using a back-end. Hence, making use of domain knowledge also provides guidance for minimizing the computational cost of the model – since by using a single layer with many vertical and horizontal filters, one can efficiently capture the same receptive field without paying the cost of going deep. Finally, note that front-ends using many vertical and horizontal filters in the first layer are an example of deep architectures relying on (musical) domain knowledge for their design.

After considering the previous discussion, we select the sample-level front-end as main part of our assumption-free model for waveforms; and we use a spectrogram front-end with many vertical and horizontal (first-layer) filters for the model designed considering domain knowledge. Experiments below share the same back-end, which enables a fair comparison among the previously selected front-ends. Unless otherwise stated, the following specifications are the ones used for the experiments – throughout the document, we refer to these specifications as the *basic* configuration:

Shared back-end. It consists of three CNN layers (with 512 filters each and two residual connections), two pooling layers and a dense layer – see Figure 2 (*Bottom-left*). We introduced residual connections in our model to explore very deep architectures, such that we can take advantage

of the large data available. Although adding more residual layers did not drastically improve our results, we observed that adding these residual connections stabilized learning while slightly improving performance [16]. The used 1D-CNN filters [9] are computationally efficient and shaped such that all extracted features are considered across a reasonable amount of temporal context (note the $7 \times M'$ filter shapes, representing *time* \times *all features*). We also make a drastic use of temporal pooling: firstly, down-sampling $\times 2$ the temporal dimensionality of the feature maps; and secondly, by making use of global pooling with mean and max statistics. The global pooling strategy allows for variable length inputs to the network and therefore, such a model can be classified as a “variable-length input” back-end. Finally, a dense layer with 500 units connects the pooled features to a sigmoidal output.

Waveform front-end. It is based on a sample-level front-end [14] composed of seven: 1D-CNN (3×1 filters), batch norm, and max pool layers – see Figure 2 (*Top-left*). Each layer has 64, 64, 64, 128, 128, 128 and 256 filters. For the 1.2M-songs dataset, we use a model with more capacity having nine layers with 64, 64, 64, 128, 128, 128, 128, 128, 256 filters. By hierarchically combining small-context representations and making use of max pooling, the sample-level front-end yields a feature map for an audio segment of 15s (down-sampled to 16kHz) which is further processed by the previously described back-end.

Spectrogram front-end. Firstly, audio segments are converted to log-mel magnitude spectrograms (15 seconds and 96 mel bins [17]) and normalized to have zero-mean and unit-var. Secondly, we use vertical and horizontal filters explicitly designed to facilitate learning the timbral and temporal patterns present in spectrograms [19–21]. Note in Figure 2 (*Right*) that the spectrogram front-end is a single-layer CNN with many filter shapes that are grouped into two branches [19]: (i) top branch – timbral features [21]; and (ii) lower branch – temporal features [20]. The top branch is designed to capture pitch-invariant timbral features that are occurring at different time-frequency scales in the spectrogram. Pitch invariance is enforced via enabling CNN filters to convolve through the frequency domain, and via max-pooling the feature map across its vertical axis [21]. Note that several filter shapes are used to efficiently capture many different time-frequency patterns: 7×86 , 3×86 , 1×86 , 7×38 , 3×38

1.2M-songs Models	train size	ROC AUC	PR AUC	\sqrt{MSE}
Baseline	1.2M	91.61%	54.27%	0.1569
Waveform	1M	92.50%	61.20%	0.1465
Spectrogram	1M	92.17%	59.92%	0.1473
Waveform	500k	91.16%	56.42%	0.1504
Spectrogram	500k	91.61%	58.18%	0.1493
Waveform	100k	90.27%	52.76%	0.1554
Spectrogram	100k	90.14%	52.67%	0.1542

Table 1. 1.2M-songs average results (3 runs) when using different training-set sizes. Baseline: GBTs+features [22].

and 1×38^5 – to facilitate learning, e.g.: kick-drums (with small-rectangular filters of 7×38 capturing sub-band information for a short period of time), or string ensemble instruments (with long vertical filters of 1×86 which are capturing timbral patterns spread in the frequency axis). The lower branch is meant to learn temporal features, and is designed to efficiently capture different time-scale representations by using several long filter shapes [20]: 165×1 , 128×1 , 64×1 and 32×1 .⁶ These filters operate over an energy envelope (not directly over the spectrogram) obtained via mean-pooling the frequency-axis of the spectrogram. By computing the energy envelope in that way, we are considering high and low frequencies together while minimizing the computations of the model – note that no frequency/vertical convolutions are performed, but 1D (temporal) convolutions. Thus, domain knowledge is also providing guidance to minimize the computational cost of the model. The output of these two branches is merged, and the previously described back-end is used for going deeper. For further details, see its online implementation.⁴

Parameters. 50% dropout before every dense layer, ReLUs as non-linearities, and our models are trained with SGD employing Adam (with an initial learning rate of 0.001) as optimizer. We minimize the MSE for the 1.2M-songs dataset, but we minimize the cross entropy for the other datasets. During training our data are converted to audio patches of 15s, but during prediction one aims to consider the whole song. To this end, several predictions are computed for a song (by a moving window of 15s) and then averaged. Although our models are capable of predicting tags for variable-length inputs, we use fixed length patches since in preliminary experiments we observed that predicting the whole song at once yielded worse results than averaging several patch predictions. In future work we aim to further study this behavior, to find ways to exploit the fact that the whole song is generally available.

5. EXPERIMENTAL RESULTS

5.1 1.2M-songs dataset

Experimental setup. As a baseline, we use a system consisting of a music feature extractor (in essence: timbre, rhythm, and harmony descriptors) and a model based on gradient boosted trees (GBT) for predicting each of the tags [22]. By predicting each tag individually, one aims

⁵ Each filter shape has 16, 32, 64, 16, 32 and 64 filters, respectively.

⁶ Each filter shape has 16, 32, 64 and 128 filters, respectively.

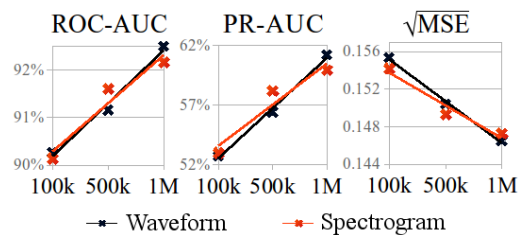


Figure 3. Linear regression fit on the 1.2M-songs results.

to turn a hard problem into multiple (hopefully simpler) problems. A careful inspection of the dataset reveals that, among tags, two different data distributions dominate the annotations: (i) tags with bi-modal distributions, where most of the annotations are zero, which can be classified; and (ii) tags with pseudo-uniform distributions that can be regressed.⁷ A regression tag example is *acoustic*, which indicates how acoustic a song is – from zero to one, zero being an electronic music song and one a string quartet. And a classification tag example can be any genre – for example, most songs will not be cataloged as *rap* since the dataset is large and its taxonomy contains dozens of genres. We use two sets of performance measurements: ROC-AUC⁸ and PR-AUC⁸ for the classification tags, and error (\sqrt{MSE})⁸ for the regression tags. ROC-AUC can lead to over-optimistic scores in cases where data are unbalanced [7]; given that classification tags are highly unbalanced, we also consider the PR-AUC metric since it is more indicative than ROC-AUC in these cases [7]. For ROC-AUC and PR-AUC, the higher the score the better – but for \sqrt{MSE} , the lower the better. Studied spectrogram and waveform models are set following the *basic* configuration – and are composed of 5.9M and 5.5M parameters, respectively. Given the unprecedented size of the dataset, we focus on how these models scale when trained with different amounts of data: 100k, 500k, or 1M songs. Average results (across 3 runs) are shown in Table 1 and Figure 3.

Quantitative results. Training the models with 100k songs took a few days, with 500k songs one week, and with 1M songs less than two weeks. The deep learning models trained with 1M tracks achieve better results than the baseline in every metric. However, the deep learning models trained with 100k tracks perform worse than the baseline. This result confirms that deep learning models require large datasets to clearly outperform strong methods based on feature-design – although note that large datasets are generally not available for most audio tasks. Moreover, the biggest performance improvement *w.r.t.* the baseline is seen for PR-AUC, which provides a more informative picture of the performance when the dataset is unbalanced [7]. In addition, the best performing model is based on the waveform front-end – being capable of outperforming the spectrogram model in every metric when trained with 1M songs. This result confirms that waveform sample-level front-ends have a great potential to learn from large data, since their solution space is not constrained by any se-

⁷ Note that all output nodes are sigmoidal – i.e., we treat classification tags as regression tags for simplicity’s sake.

⁸ ROC: Receiver Operating Characteristic. PR: Precision Recall. AUC: Area Under the Curve. MSE: Mean Squared Error.

vere architectural choice. On the other hand, the architectural choices defining the spectrogram front-end might be constraining the solution space. While these architectural constraints are not harmful when training data are scarce (as for the 100k/500k songs results or in prior works [24]), such a strong regularization of the solution space may limit the learning capacity of the model in scenarios where large training data are available – as for the 1M songs results. One can observe this in Figure 3, where we fit linear models to the obtained results to further study this behavior. When 100k training songs are available: trend lines show that spectrogram models tend to perform better. However, when 1M training songs are available: the lines show that waveform models outperform the spectrogram ones. It is worth mentioning that the observed trends are consistent throughout metrics: ROC-AUC, PR-AUC, and \sqrt{MSE} . Finally, note that there is room for improving the models under study – e.g.: one could address the data imbalance problem during training, or improve the back-end via exploring alternative temporal aggregation strategies.

Qualitative results. Since it is the first report of a deep music tagging model trained with such a large dataset, we also perceptually assess the quality of the estimates. To this end, we compared the predictions of one of our best performing models to the predictions of the baseline, and to the human-annotated ground-truth tags. Some interesting examples identified during this qualitative experiment are available online.⁹ First, we observed that the deep learning model is biased towards predicting the popular tags (such as *lead vocals*, *English* or *male vocals*). Note that this is expected since we are not addressing the data unbalancing issue during training. And second, we observe that the baseline model (which predicts the probability of each tag with an independent GBT model) predicts mutually exclusive tags with high confidence – e.g., it predicted with high scores: *East Coast* and *West Coast* for an East Coast rap song, or *baroque period* and *classic period* for a Bach aria. However, the deep learning model (predicting the probability of all tags together) was able to better differentiate these similar but mutually exclusive tags. This suggests that deep learning has an advantage when compared to traditional approaches, since these mutually exclusive relations can be jointly encoded within the model.

5.2 MagnaTagATune (MTT) dataset

Experimental setup. State-of-the-art models are set as baselines, and we use the same (classification) performance metrics as for the 1.2M-songs dataset: ROC-AUC and PR-AUC – note that the MTT labels are binary. One of the baseline results (the SampleCNN [14] with 90.55 ROC-AUC) was computed using a slightly different version of the MTT dataset – which only includes songs having more than 1 tag and lasting more than 29.1 seconds. As a result, this cleaner version of the MTT dataset is of $\approx 21k$ songs instead of $\approx 26k$. Although this dataset cleans out potential noisy annotations, we decided to use the original dataset to easily compare our results with former works. Thus, to fairly compare our models with

the SampleCNN, we reproduce their work considering the original dataset – achieving a score of 88.56 ROC-AUC. Given that less noise is present in the SampleCNN dataset, it seems reasonable that their performance is higher than the one obtained by our implementation.

The MTT experiments can be divided in two parts: waveform and spectrogram models – see Tables 2 and 3. Due to the amenable size of the dataset (every MTT experiment lasts $< 5h$), it is feasible to run a comprehensive study investigating different architectural configurations. Specifically, we study how waveform and spectrogram architectures behave when modifying the capacity of their front- and back-ends. For example, the experiment “# filters $\times 1/2$ ” in Table 2 consists of dividing the number of filters available in the waveform front-end by two. This means having 32, 32, 32, 64, 64, 64 and 128 filters, instead of the 64, 64, 64, 128, 128, 128 and 256 filters in the *basic* configuration. We also apply this methodology to the spectrogram front-ends, and we add/remove capacity to them by increasing/decreasing the number of available filters. After running the front-end experiments with a fixed back-end (following the *basic* configuration: 512 CNN filters, 500 output units), we select the most promising ones to proceed with the back-end study – for waveforms: “# filters $\times 2$ ”,¹⁰ and for spectrograms: “# filters $\times 1/2$ ”. Having now a fixed front-end for every experiment, we modify the capacity of the back-end via changing the number of filters in every CNN layer (512, 256, 128, 64) and changing the number of output units (500, 200). Since the *basic* configuration leads to relatively big models for the size of the dataset, these experiments explore smaller back-ends. The inputs for the MTT are set to be of 3s, since longer inputs yield worse results [15, 21].

Quantitative results. The waveform and spectrogram models we study outperform the proposed baselines – which represent the current state-of-the-art. Further, performance is quite robust to the number of parameters of the model. Although the best results are achieved by models having higher capacity, the performance difference between small and large models is minor – what means that relatively small models (which are easier to deploy) can do a reasonable job when tagging the MTT music. Finally: spectrogram models perform better than waveform models for this small public dataset – which aligns with previous works using datasets of similar size [20, 21]. Consequently, these results confirm that domain knowledge intuitions are valid guides for designing deep architectures in scenarios where training data are scarce.

5.3 Million Song Dataset (MSD)

Experimental setup. State-of-the-art models are set as baselines, and we use the same (classification) performance metrics as for the 1.2M-songs dataset: ROC-AUC and PR-AUC – note that the MSD labels are binary. These experiments aim to validate the studied models with the biggest public dataset available. Models are set following the *basic* configuration, and results are shown in Table 4.

⁹ <http://www.jordipons.me/apps/music-audio-tagging-at-scale-demo>

¹⁰ “# filters $\times 2$ ” front-end was selected instead of “# filters $\times 4$ ”, because it performs similarly with less parameters.

MTT dataset	ROC	PR	#
Waveform models	AUC	AUC	param

State-of-the-art results – with our own implementations

SampleCNN [14] ¹³	90.55	-	2.4M
SampleCNN (reproduced)	88.56	34.38	2.4M
Dieleman et al. [9]	84.87	-	-
Dieleman et al. (reproduced)	85.58	29.59	194k

How much capacity is required for the front-end?

# filters × 4	89.05	34.92	11.8M
# filters × 2 (selected)	88.96	34.74	7M
# filters × 1	88.9	34.18	5.3M
# filters × 1/2	88.69	33.97	4.7M
# filters × 1/4	88.47	33.89	4.4M

How much capacity is required for the back-end?

# filters in every CNN layer - # units in dense layer			
64 CNN filters - 500 units	88.57	33.99	1.3M
- 200 units	<u>88.94</u>	<u>34.47</u>	1.3M
128 CNN filters - 500 units	88.82	34.62	1.8M
- 200 units	88.81	34.6	1.7M
256 CNN filters - 500 units	88.95	34.27	3.1 M
- 200 units	88.59	34.39	2.9M
512 CNN filters - 500 units	<u>88.96</u>	<u>34.74</u>	7M
- 200 units	88.3	34.05	6.7M

Table 2. MTT results: waveform models.

Quantitative results. The spectrogram model outperforms the waveform model for this public dataset – having $\approx 200k$ training songs. Furthermore, the spectrogram model performs equivalently to ‘Multi-level & multi-scale’ [13], which is the best performing method in the literature – denoting that musical knowledge can be of utility to design models for the MSD. Additionally, the waveform model performs worse than other waveform-based models that also employ sample-level front-ends. Such performance decrease could be caused because (i) SampleCNN methods [14, 15] average ten¹¹ estimates for the same song to compensate for possible faults in song-level predictions, while our method only averages two – predicting consecutive patches of 15s; or (ii) because the major difference between SampleCNN and the waveform model is that the latter employs a global pooling strategy that could remove potentially useful information for the model. Besides, the best performing waveform-based model (‘SampleCNN multi-level & multi-scale’ [15]) also achieves lower scores than the best performing spectrogram-based ones. Considering the outstanding results we report when the waveform model is trained with 1M songs, one could argue that the lack of larger public datasets is limiting the outcomes of deep learning research for music auto-tagging – particularly when dealing with end-to-end learning stacks processing raw data such as waveforms.

¹¹ Since MSD audios are of 30s, ten tag estimates per song can be obtained via running the model with consecutive patches of 3s.

¹³ Result computed with a different MTT version, see section 5.2.

¹⁴ Reproduced using 96 mel bands instead of 128 as in [21].

MTT dataset	ROC	PR	#
Spectrogram models	AUC	AUC	param

State-of-the-art results – with our own implementations

VGG - Choi et al. [5]	89.40	-	22M
VGG (reproduced)	89.99	37.56	450k
Timbre CNN [21]	89.30	-	191k
Timbre CNN (reproduced) ¹⁴	89.07	34.92	220k

How much capacity is required for the front-end?

# filters × 1/8	90.08	37.18	4.4M
# filters × 1/4	90.12	37.69	4.6M
# filters × 1/2 (selected)	90.40	38.11	5M
# filters × 1	90.31	37.79	5.9
# filters × 2	90.07	37.29	7.6M

How much capacity is required for the back-end?

# filters in every CNN layer - # units in dense layer			
64 CNN filters - 500 units	90.03	36.98	277k
- 200 units	<u>90.28</u>	<u>37.55</u>	222k
128 CNN filters - 500 units	90.16	37.61	617k
- 200 units	<u>90.28</u>	<u>37.69</u>	524k
256 CNN filters - 500 units	90.18	37.98	1.6M
- 200 units	90.06	37.16	1.4M
512 CNN filters - 500 units	<u>90.40</u>	<u>38.11</u>	5M
- 200 units	89.98	37.05	4.7M

Table 3. MTT results: spectrogram models.

6. CONCLUSIONS

This study presents the first work describing how different deep music auto-tagging architectures perform depending on the amount of available training data. We also present two architectures that yield results on par with the state-of-the-art. These architectures are based on two conceptually different design principles: one is based on a waveform front-end, and no domain knowledge inspired its design; and the other, with a spectrogram front-end, makes use of (musical) domain knowledge to justify its architectural choices. While our results suggest that models relying on domain knowledge play a relevant role in scenarios where no sizable datasets are available, we have shown that, given enough data, assumption-free models processing waveforms outperform those that rely on musical domain knowledge.

MSD Models	ROC AUC	PR AUC	# param
Waveform (<i>ours</i>)	87.41	28.53	5.3M
SampleCNN [14]	88.12	-	2.4M
SampleCNN multi-level & multi-scale [15]	88.42	-	-
Spectrogram (<i>ours</i>)	88.75	31.24	5.9M
VGG + RNN [6]	86.2	-	3M
Multi-level & multi-scale [13]	88.78	-	-

Table 4. MSD results. *Top* – waveform-based models. *Bottom* – spectrogram-based models.

7. ACKNOWLEDGMENTS

This work was partially supported by the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502) – and we are grateful for the GPUs donated by NVidia.

8. REFERENCES

- [1] Yann Bayle, Pierre Hanna, and Matthias Robine. Revisiting autotagging toward faultless instrumental playlists generation. *arXiv preprint arXiv:1706.07613*, 2017.
- [2] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2011.
- [3] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.
- [4] Ning Chen and Shijun Wang. High-level music descriptor extraction algorithm based on combination of multi-channel cnns and lstm. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- [5] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. *International Society for Music Information Retrieval Conference (ISMIR)*, 2016.
- [6] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.
- [7] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *International Conference on Machine Learning (ICML)*. ACM, 2006.
- [8] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- [9] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.
- [10] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.
- [11] Edith Law, Kris West, Michael I Mandel, Mert Bay, and J Stephen Downie. Evaluation of algorithms using games: The case of music tagging. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2009.
- [12] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems*, 2009.
- [13] Jongpil Lee and Juhan Nam. Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging. *IEEE signal processing letters*, 24(8):1208–1212, 2017.
- [14] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. *Sound and Music Computing Conference (SMC)*, 2017.
- [15] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification. *Applied Sciences*, 8(1):150, 2018.
- [16] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.
- [17] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. 2004.
- [18] Huy Phan, Lars Hertel, Marco Maass, and Alfred Mertins. Robust audio event recognition with 1-max pooling convolutional neural networks. *arXiv preprint arXiv:1604.06338*, 2016.
- [19] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6. IEEE, 2016.
- [20] Jordi Pons and Xavier Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.
- [21] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra. Timbre analysis of music audio signals with convolutional neural networks. *European Signal Processing Conference (EUSIPCO2017)*, 2017.
- [22] Matthew Prockup, Andrew J. Asman, Fabian Gouyon, Erik M. Schmidt, Oscar Celma, and Youngmoo E. Kim. Modeling rhythm using tree ensembles and the music genome project. *Machine Learning for Music*

Discovery Workshop at the International Conference on Machine Learning (ICML), 2015.

- [23] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, 2016.
- [24] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals. Learning the speech front-end with raw waveform cldnns. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [25] Jan Schluter and Sebastian Bock. Improved musical onset detection with convolutional neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.
- [26] Mohamed Sordo, Cyril Laurier, and Oscar Celma. Annotating music collections: How content-based similarity helps to propagate labels. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2007.
- [27] Zhenyao Zhu, Jesse H Engel, and Awni Hannun. Learning multiscale features directly from waveforms. *arXiv preprint arXiv:1603.09509*, 2016.

AUDIO BASED DISAMBIGUATION OF MUSIC GENRE TAGS

Romain Hennequin, Jimena Royo-Letelier, Manuel Moussallam

Deezer R&D, Paris
research@deezer.com

ABSTRACT

In this paper, we propose to infer music genre embeddings from audio datasets carrying semantic information about genres. We show that such embeddings can be used for disambiguating genre tags (identification of different labels for the same genre, tag translation from a tag system to another, inference of hierarchical taxonomies on these genre tags). These embeddings are built by training a deep convolutional neural network genre classifier with large audio datasets annotated with a flat tag system. We show empirically that they make it possible to retrieve the original taxonomy of a tag system, spot duplicate tags and translate tags from a tag system to another.

1. INTRODUCTION

Large genre annotated databases have been made available lately: the Google Audio Set (GAS) [12], the MuMu dataset [20], Discogs [1] or the Free Music Archive (FMA) dataset [6] all contain hundreds of genre tags and hundreds of thousands multi-label genre track annotations.

Every dataset with genre annotations has its own genre representation: usually it is a tag set which is sometimes organized with a basic taxonomy (Discogs, MuMu, FMA) or a basic ontology (GAS).

However these representations usually suffer from ambiguity issues. First, tag definition may not be explicit: for the same tag name, definition may not be coherent from a dataset to another which prevents from doing correct translation from one tag set to another with a simple string matching. Second, there may be duplicated tags *i.e.* tag with different names but referring to the exact same genre such as *Bossa Nova* and *Bossanova* (without space) in Discogs. Thirdly, there may be polysemy issues for some tags: it happens that a single tag refers to different concepts. In Discogs, the tag *hardcore* may refer to *hardcore punk* or to *hardcore electronic music* which are quite different genres. Finally, while a tag set may be structured in a taxonomy or an ontology, those have limitation for expressing all relations between tags: for instance the tag *Blues Rock* in the MuMu taxonomy is a subgenre of *Rock* and is not related to *Blues*, which makes it as close to *Elec-*

tric Blues as to *Drum & Bass* according to the taxonomy. Moreover taxonomy and ontology are generally designed with a particular purpose in mind [21], possibly clarity for the customer for the MuMu taxonomy (which is the Amazon taxonomy), while it may be musicologic precision for DBpedia¹, which may result in different meaning for tags and different relationship between them.

Building a genre representation from these tag systems in order to deal with these ambiguity issues can be done using a top-down approach, using an expert-level ontology such as the DBpedia ontology and trying to project the tag system into this ontology [7]. Mapping tags to an external expert ontology is not trivial, as a genre can have several different name and some tags may have several meanings: the tag *funk* for instance may refer to a genre born in the 60s derived from soul and jazz, or, in Brazil, to Funk carioca which is a totally different style inspired by gangsta rap music. It also can be done using a bottom up approach, inferring relations between entities from data. The latter was mainly done using the genre tag distribution of a dataset with Latent Semantic Analysis (LSA) [28] or with a straight use of cooccurrences [26, 27] which all rely on the distributional hypothesis (similar tags are tags that cooccur a lot with same other tags). However, it is sometimes not possible to rely only on tag distributions: the MuMu dataset has no overlap with the GAS, which prevents from using tags cooccurrences to infer relationship between MuMu genre tags and GAS ones.

So far, the literature has been mainly focusing on music genre classification on flat tag systems from audio [4, 8, 9, 23, 24, 30], text such as reviews [13, 19] or lyrics [3, 17], album covers [16] or combinations of the previous modalities [18, 20, 25], while rarely addressing the actual semantic relationships that exist between genres. In [29], the authors pointed out that focusing on classification metrics was not sufficient and suggested a deeper results analysis such as explanation of the confusion of the classifiers in term of musicological aspects. In this paper, we suggest going deeper in this direction and seeing how the confusion of the classifier is able to generate a structured genre representation: if the classifier is good enough, the confusion it makes should be able to encode the relation of proximity between genres. Showing this property has two implications: it shows in a qualitative way that the classifier performs well and allows generation of a structured representation of a tag system using audio.

In this paper, we thus aim to disambiguate genre tags



© Romain Hennequin, Jimena Royo-Letelier, Manuel Moussallam. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Romain Hennequin, Jimena Royo-Letelier, Manuel Moussallam. "Audio based disambiguation of music genre tags", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ wiki.dbpedia.org

and relations between them using audio as an alternative to the distributional hypothesis: we propose a method able to spot inconsistencies, help reducing them and relate tags between themselves, possibly across different non overlapping datasets with different tag systems. We enforce that the representation is based on audio only information and not on tag distribution using a monolabel learning scheme. While extracting a semantic representation from audio annotated with a flat tag representation was already sparsely addressed (in [14], basic ontological relations between a few instruments are learnt back from isolated music instruments sounds and in [15] a simple music genre taxonomy is learnt with a few genre concepts), in this paper, we propose to learn representations at a large scale for tag systems with several hundreds of genre tags and with datasets of several hundreds of thousands of songs.

In Section 2, we explain how we compute genre tag embeddings using an audio-based genre classifier and use them to define an audio-based similarity between genre tags. In Section 3, we validate the learnt similarity by showing that it performs fairly on two artificial tasks (Discogs taxonomy learning and artificial deduplication). In Section 4, we show how we can use the learnt similarity to translate tags from a dataset tag system to another. Finally, we draw conclusions in Section 5.

2. LEARNING A GENRE REPRESENTATION

In this section, we explain how we build embeddings of genre tags using a genre classifier with audio input. We associate to each genre tag t_i in the genre tag set $T = \{t_1, \dots, t_{N_c}\}$ an embedding vector $\mathbf{f}(t_i) = \mathbf{v}_{t_i} \in \mathbb{R}^n$, such that $d(\mathbf{v}_{t_1}, \mathbf{v}_{t_2})$ should correspond to an audio similarity between genre tag t_1 and genre tag t_2 .

2.1 Datasets

We use two large-scale genre annotated datasets for our experiments: The MuMu dataset [20], and the genre provided by the Discogs website². We matched both datasets to Deezer track IDs using song metadata (album and artist names, and track titles). We extracted a 30s-long excerpt for each track (the position of the excerpt was sampled at random between the beginning and the end of the track). For tags with too few occurrences, we extracted several excerpts for balancing (as explained in Section 2.2). To avoid overlap between datasets we removed the 7260 tracks that belong to both datasets (in order to not affect the translation experiment of Section 4).

While each dataset provides a simple genre taxonomy, we do not rely on it in the classification stage and consider the genre annotations as flat tag systems with no links between tags. The provided taxonomies are used afterwards for evaluation of the built genre representation.

2.1.1 Discogs

Discogs is referred as the “*largest open database containing explicit crowd-sourced genre annotations*” in [1]. It

contains genre annotations at the album level for hundreds of thousands of albums. Genre tags in Discogs are organized in a two-level hierarchy: the first level, referred as *genre*, includes generic genre categories (*genre:Rock*³, *genre:Jazz*, etc...) and the second level, referred as *style*, corresponds to subgenres (*Psychedelic Rock*, *Cool Jazz*, etc...). It contains a total of more than 500 genre/style tags. Only the 250 most common tags were kept in our experiments (235 style tags and 15 genre tags).

After cleaning, balancing (see Section 2.2) and matching, the Discogs dataset we used contained 418184 tracks.

2.1.2 MuMu dataset

The MuMu dataset [20] has genre annotation based on the Amazon 4-level genre taxonomy. It contains genre annotations at the album level for 31471 albums which contain a total of 147295 tracks. It contains a total of 446 different genres. Only the top 211 tags (those with less than 300 annotated tracks are discarded) are kept. After cleaning, balancing (see Section 2.2) and matching, the final MuMu dataset we used contained 122014 tracks.

2.1.3 Dataset split

When training the system described in Section 2.3, we split the datasets into a training dataset (70%), a validation dataset (10%) used for early stopping, and a test dataset (20%) used for building genre representations (Section 2.4). The split was done at the artist level meaning two tracks by the same artist are in the same part of the split in order to avoid overfitting on variables such as album or artist as advised in [11, 22].

2.2 Monolabel learning

The annotations in a multilabel dataset carry information of popularity (through number of occurrences of a tag) and of similarity (through cooccurrences of tags). This information was already used in several papers to build genre taxonomies from a flat tag system [26–28] or to build a target representation to improve classification results [20].

The goal of the paper is to learn a genre representation only through audio and to avoid using non-audio information such as the one provided by the tag distribution. As this distribution can be easily learnt as a side information in the last layer of a neural network, where bias can encode popularity (higher bias for more popular genre) while weights can encode similarity between genres (important value of dot product between weights corresponding to similar genres and vice versa), simply training a multilabel audio classifier based on a neural net will result in taking advantage of this information, and it may be difficult to assess at what point the actual audio information is relevant in building the representation from this classifier.

In order to avoid influence of these non-audio information in the built genre representation, we propose to turn the multilabel classification problem into a monolabel one using the following learning scheme:

² <https://discogs.com>

³ We prefix Discogs genre by “*genre:*” to distinguish them from style

- To remove cooccurrences information, we transform the multilabel dataset into a monolabel one by sampling a tag among the multilabel tag annotation of every track.
- To remove the popularity information, we balance equally all classes using a sampling probability inversely proportional to the global popularity of a tag (note, that it does not enforce perfect balancing).

For instance, if *Rock* appears 1000 times in the dataset and *Punk* appears 100 times, a song with (multi-)labels $\{Rock, Punk\}$ will get as monolabel *Rock* with probability $1/11$ and *Punk* with probability $10/11$. This ensures that rare genre tags have a high probability of being drawn, and that we keep the maximum of available information for rare tags while discarding somewhat redundant information for very common tags.

To enforce balancing, tags with too many occurrences are downsampled to keep a maximum of 2000 occurrences per tag. Genre with not enough occurrences are upsampled to 2000 occurrences by duplicating tracks (different 30s excerpts are chosen for each track).

In order to avoid fitting independent variables, the sampling is done at the album level, which means that every track from the same album gets the same label. It also ensures that different excerpts of the same track have the same label. Using this learning scheme, the confusion between genres should result only from similarities in audio.

2.3 Classification system

We use a convolutional neural network with a recurrent layer on top of it as a monolabel classifier. We feed it with Mel-spectrograms computed with 1024 samples long Hann windows without overlap, with 96 Mel filters. Audio is first downsampled to 22050Hz and stereo channels are summed up. Mel-spectrograms were log compressed using the function $f(x) = \log(1 + Cx)$ where we chose $C = 10000$. It results in 646×96 input matrices.

The architecture of the neural network is quite similar to the one used in [4] for automatic tagging, but with half as many filters in the convolutional layers (we noticed that it resulted in less overfitting) and a Gated Recurrent Unit [2] on top of the conv layer (which improved overall classification accuracy). The gated linear unit was used for temporal pooling (only last temporal output is forwarded to the last layer which removes the time dimension) and was used in conjunction with dropout to reduce overfitting. The architecture is summed up in Table 1.

The network was trained with a categorical cross-entropy loss with mini-batch stochastic gradient descent using Adadelta [32] and early stopping on the validation loss. The system was implemented with Keras [5] using the Tensorflow [10] backend.

As the main goal of the paper is not to perform in terms of classification results, we did not try to optimize thoroughly the architecture and we just checked that our proposed system had similar classification results as in [20].

Layer	output shape	N param.
Log-comp Mel-spec	646×96×1	0
Conv 3×3×64 - MP 2×2	323×48×64	640
Conv 3×3×128 - MP 3×4	107×12×128	1280
Conv 3×3×256 - MP 2×3	53×4×256	2560
Conv 3×3×512 - MP 3×4	17×1×512	5120
GRU 512	512	1574400
Dense Softmax	N_c	$512 \times N_c$

Table 1. Architecture of the Neural Network. MP stands for Max Pooling.

2.4 Genre embeddings from classification

There are several ways of extracting an embedding from a neural net based classification system. We describe the three kinds of genre embeddings we generated from the audio classifier in the following subsections. Whereas the first embedding only uses parameters of the classifiers, the other two make use of the test set.

2.4.1 Last hidden layer weights

The weights of the last hidden layer \mathbf{W} are a $512 \times N_c$ matrix. The i -th column of this matrix is then chosen as the embedding of genre tag t_i :

$$\mathbf{f}_w(t_i) = \mathbf{v}_{t_i} = \mathbf{W}_{:,i}. \quad (1)$$

This is a straightforward representation of a genre tag in the network: the output of the last hidden layer for a track annotated with some genre should be similar (in terms of dot product) to the weight vector of this genre. However, it necessitates retraining to incorporate new genre tags in the embedding.

2.4.2 Columns of output

We can also build an embedding using the test dataset: for every track s in the test dataset, we denote T_s the set of tags associated to s . We note the test dataset $S = \{s_1, s_2 \dots s_{N_s}\}$ where s_k are the track excerpts. The output of the network when fed with track excerpt s_i is a vector $\mathbf{p}_k \in [0, 1]^{N_c}$ (with $\sum_{j=1}^{N_c} [\mathbf{p}_k]_j = 1$). We note \mathbf{P} the matrix in $\mathbb{R}^{N_s \times N_c}$ with \mathbf{p}_k as k -th row. The embedding of tag t_i is then defined as the i -th column of matrix \mathbf{P} :

$$\mathbf{f}_c(t_i) = \mathbf{v}_{t_i} = \mathbf{P}_{:,i}. \quad (2)$$

This embedding does not require annotation information about the tracks of the test set and the cosine similarity matrix between embeddings of all pairs of genre can be understood as a normalized confusion matrix and is the audio counterpart of the occurrence based representation defined in (4). However it has very large dimension (that may be reduced using dimension reduction techniques such as LSA) and it is quite difficult to add extra genres without retraining the whole system.

2.4.3 Mean of output

This third embedding type also uses the test dataset and takes advantage of the annotations. We note $S_{t_i} = \{s_{k_1}, s_{k_2} \dots s_{k_{N_{t_i}}}\}$ the set of tracks annotated with genre tag t_i . We then associate to each t_i the set of outputs of the

classifier $\{\mathbf{p}_k | s_k \in S_{t_i}\}$. Ideally each genre tag t_i would be represented by the distribution of all possible outputs for this genre. In practice, we compute statistics on these distributions. We then define the third type of genre tag embeddings as the mean of the output:

$$\mathbf{f}_m(t_i) = \mathbf{v}_{t_i} = \frac{1}{|S_{t_i}|} \sum_{s_k \in S_{t_i}} \mathbf{p}_k. \quad (3)$$

As \mathbf{p}_k is a categorical probability distribution, $\mathbf{f}_m(t_i)$ is too. Embedding \mathbf{f}_m makes it possible to incorporate new tags without retraining the whole system, by simply adding tracks annotated with the new genre tag in the dataset (the only constraints would be that the classifier was trained with similar genres): this is an important property of the embedding since it makes it much easier to incorporate new knowledge from another tag system.

2.4.4 Occurrence based representation

In order to compare the audio-based representation we also define the following representation which is not based on audio but on tag distribution only. We note $\mathbf{M} \in \{\mathbf{0}, \mathbf{1}\}^{N_s \times N_c}$ the multilabel tag occurrence matrix with coefficient $M_{ij} = 1$ iff track s_i is annotated with tag t_j . The cocurrence embedding of tag t_i is then defined as the i -th column of matrix \mathbf{M} :

$$\mathbf{f}_{\text{dist}}(t_i) = \mathbf{M}_{:,i}. \quad (4)$$

This definition then shares similarity with the audio-based representation \mathbf{f}_c .

2.4.5 Similarity measure

To compare tags, we use the cosine similarity applied to the four types of genre tag embeddings defined in Equations (1), (2), (3) and (4).

3. MODEL VALIDATION

In this section, we validate that the audio-based similarities learnt in Section 2 have a semantic meaning by showing that the original Discogs taxonomical relations can be inferred from the similarities and that they make it possible to spot duplicate tags in a dataset. In order to reproduce the results, we make available the embeddings, the similarity matrices we obtained for the different representation⁴ as well as dataset files (as lists of Deezer song IDs).

3.1 Taxonomy Learning

In this section, we use similarity obtained from the genre embeddings described in Section 2, to infer hierarchical links between genres. We trained the classification system with the Discogs dataset and the purpose of the experiment is to infer the genre/style links of the two-level Discogs taxonomy from audio.

The cosine similarity computed between genre tag embeddings provides a measure of similarity between genre

	\mathbf{f}_w	\mathbf{f}_c	\mathbf{f}_m	\mathbf{f}_{dist}
HR@1	85.1±4.6	89.4±3.9	87.7±5.2	96.2±2.5
HR@2	91.9±3.5	98.3±1.7	96.2±3.2	100.0±0
MAP	90.6±2.9	94.2±2.2	93.1±3.0	98.1±1.2

Table 2. Average ranking metrics (in %) for the Discogs taxonomy learning task with 95% confidence intervals.

tags. This can be used to rank for each style the similarity with each of the 15 genres. The ground truth is the actual genre associated to the style in the Discogs taxonomy (note that some rare style are associated to 2 music genres, such as *hardcore* and *noise* which are associated to both *rock* and *electronic*). We measure the quality of this ranking with classic ranking metrics: Hit Rate (HR)@k which is the percentage of style for which the associated genre is in the top-k according to the similarity score. (HR@1 can be considered as a classification accuracy) and Mean Average Precision (MAP) as defined in [33]. MAP takes into account the rank of the related genre in the similarity list.

Results are presented in Table 2. As a reference, we report results for the occurrence based embedding \mathbf{f}_{dist} . As style tags are always present together with their parent genre tag in the annotations, the performance of the occurrence based representation should be interpreted as an upper-bound for the results of the other representations, the errors being likely due to incoherences in the Discogs taxonomy (which is confirmed by the perfect HR@2 score of \mathbf{f}_{dist}). Among the audio-based representations, \mathbf{f}_c performs better than the two others. Despite being smaller than the occurrence based representation, we can see that the metrics for the audio representations are quite high, notably for \mathbf{f}_c which has a near perfect HR@2. This is noteworthy, since only audio information is used to infer the relations.

A qualitative analysis of the error shows that most of the “errors” (in the sense that the most similar genre to a style is not its related genre) actually make sense: for instance *blues rock* which is a subgenre of *genre:rock* in Discogs taxonomy has the greatest similarity (for \mathbf{f}_c) with *genre:blues* which makes as much sense as the other (the same phenomena with hybrid subgenre appears with *jazz-funk* and *genre:funk / soul* instead of *genre:jazz*, *pop rock* and *genre:rock* instead of *genre:pop* and *soul-jazz* and *genre:funk / soul* instead of *genre:jazz*). Other noteworthy examples are *bossa nova* (subgenre of *genre:jazz*) associated with *genre:latin*, *musique concrète* (subgenre of *genre:electronic*) associated to *genre:non-music* or *rnb/swing* (subgenre of *genre:hip hop*) associated to *genre:funk / soul*. These qualitative results confirm that most of the “errors” are actually due to limitations of the original taxonomy and that HR@2 may be the most revealing metric. In Figure 1, we plot a 2D t-distributed stochastic neighbor embedding (t-SNE) [31] of the learnt audio representation \mathbf{f}_w in order to get visual insights about it: most music style tags are gathered in coherent clusters and are most of the time close to their related genre tag. A noteworthy exception is the style tags related to *folk*, *world*, & *country* that form several clusters, one of which being next to *latin*, another one being next to *blues* and an-

⁴ github.com/deezer/audio_based_disambiguation_of_music_genre_tags.git

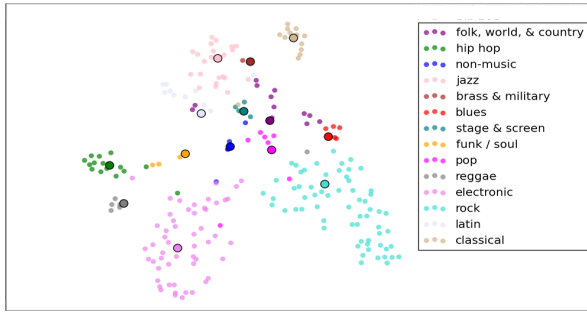


Figure 1. 2D t-SNE of f_w for the Discogs tags. Each *style* is colored with the same color as its related *genre*. Main genres are depicted with bigger circle and black edges.

other one next to *pop*. This is pretty coherent since the tag *folk, world, & country* is supposed to gather several very different styles that may be closely related to other genres.

3.2 Tag deduplication

In this section, we show how the audio-based similarities learnt in Section 2 can be used to spot duplicates in a tag system. To do that we rely on the ability of a classifier based on audio data to discriminate between two genre tags. If two genre tags cannot be discriminated, they probably have some strong relation (even if they have very dissimilar names). There may be several reasons for two tags having high confusion similarity: First, they may represent the exact same genre. Second, genre related audio characteristics may be very similar (the genre may be very similar with respect to audio). Thirdly, there may be differences of distribution in the datasets: datasets are usually an imperfect sample of the set of all music. Some genre may be biased toward a subgenre in a dataset while not in another one, which may result in strong differences in the meaning of some genres. Last, the classifier may not be able to distinguish them while there exists difference in some audio characteristics (that the classifier is not able to handle).

As it is very difficult to assess a ground truth for such a deduplication experiment, we propose the following artificial tag duplication: we use the Discogs genre dataset. We artificially duplicate every genre tag by creating two duplicate tags: for instance, *Rock* is duplicated into *Rock1* and *Rock2*, which means that half of the tracks originally annotated with *Rock* get the annotation *Rock1* instead while the other half get the annotation *Rock2*. To avoid learning the similarity through artist specific characteristics, we perform the split at the artist level, meaning that tracks of the same artist annotated with *Rock* will get all the same subtag (either *Rock1* or *Rock2*). Note that a subtag of group 1 cannot cooccur with a subtag of group 2, which results in two separate tag systems (that we will refer as system 1 and system 2), with no overlap. While all tags from system 1 having a semantically equivalent counterpart in system 2 is quite artificial, the total separation between the tag systems in term of cooccurrences is realistic. There is, for instance, no overlap between the GAS and the MuMu dataset which means we can only rely on audio for linking them.

In a similar way as in the experiment of Section 3.1,

	f_w	f_c	f_m
HR@1	92.0 \pm 2.4	92.8 \pm 2.3	74.8 \pm 3.8
HR@2	95.8 \pm 1.8	97.0 \pm 1.5	83.0 \pm 3.3
MAP	98.1 \pm 0.6	98.4 \pm 0.5	93.3 \pm 1.1

Table 3. Average ranking metrics (in %) for the Discogs deduplication task with 95% confidence intervals.

we use the similarity between genre tags embeddings as a *duplication score*. The task is then for each genre tag, to retrieve its duplicated tag. Once again, we present quantitative results in terms of HR@k and MAP in Table 3. As opposed to the taxonomy learning task, it does not make sense to compare the audio based representations to the occurrence-based representation since the sampling scheme we use avoid a tag of group 1 cooccurring with a tag of group 2 which means that the cosine similarity between any tag of group 1 with any tag of group 2 is 0. f_w and f_c performs similarly, both performing significantly better than f_m . Once again the score seems reasonably high for a representation based on audio information only.

It is interesting to look at the “errors” (when the most similar tag is not the actual duplicate) done by the system using f_c . Some errors were actual duplicates in Discogs: *bossa nova* was associated to *bossanova* (without a space) which is clearly a duplicate issue in Discogs. Other example are *style:reggae* and *genre:reggae* (where a *style* tag as the same name as its related *genre* tag) or *thug rap* and *gangsta* (considered as the same genre in Wikipedia). This shows that the genre similarity computed from the embeddings is able to spot actual duplicates and that HR@2 may be again the most revealing metric. Some errors are matching between quite different concepts but with very similar audio, such as *field recording/musique concrète*, *poetry/spoken word*, *spoken word/genre:non-music* and *conscious/genre:hip hop*. Other errors are with very similar genres: *bop/hard bop*, *honky tonk/country blues*, *space rock/post rock*. A few errors are more difficult to explain such as *ragtime/tango* which may have some audio similarities (the use of piano is quite common in both genres, and both are intended for dancing). These errors may come from the classification system we use or from a strong bias or annotation noise in the Discogs annotations.

4. TAGS TRANSLATION

In this section, we perform another experiment that aims at translating tags from MuMu dataset to Discogs dataset. For sack of clarity Discogs tags are prefixed with “D:” and MuMu tags with “M:”. When there are no or few overlaps between two datasets, we cannot rely on cooccurrences of tags to model relation between the tag systems. The only media we can rely on is then audio.

To train the classifier (see Section 2.3), we used the concatenation of the tags from the MuMu dataset and the Discogs dataset. Tags of each dataset were considered different even if they had the exact same name: e.g., there were a *M:jazz* tag that was considered different from the *D:jazz* tag. The experiment of translation is then very similar to the deduplication task presented in 3.2:

Audio-based translation \mathbf{f}_c		Cooccurrence-based translation \mathbf{f}_{dist}	
Mumu tag	Discogs tag	Mumu tag	Discogs tag
bebop	bop	irish folk	celtic
movie scores	score	contemporary big band	big band
indie & lo-fi	lo-fi	latin music	genre:latin
electric blues	modern elec. blues	rap & hip-hop	genre:hip hop
electronica	leftfield	vocal blues	ragtime
punk-pop	pop punk	dance & electronic	genre:electronic
modern postbebop	genre:jazz	today's country	country
special interest	avantgarde	electric blues	genre:blues
singer-songwriters	folk rock	children's music	genre:children's
r&b	rnb/swing	comedy & spoken word	comedy

Table 4. Top 10 most similar tags between MuMu and Discogs according to \mathbf{f}_c (left columns) and \mathbf{f}_{dist} (right columns), removing string matched tags.

the translation task consists of deduplicating the whole MuMu/Discogs tag set, focusing on pairs of duplicates for which the first element is a MuMu tag and the second element is a Discogs tag.

This allows to translate tags from one tag system to another, but also to spot possible genre definition differences between datasets: if two genre tags from two different datasets, with the exact same name can be discriminated with audio, this is probably because they do not carry the exact same meaning (provided we can move apart overfitting of the audio classifier used to build the representation).

We only consider here simple one-to-one tag mappings between MuMu and Discogs although it is restrictive since there may exist one-to-many mapping (e.g. between *M:avant garde & free jazz* and *D:avant-garde jazz/D:free jazz*) or even more complex relationships.

As the Discogs and MuMu datasets have some common tracks, we can compare the audio-based similarities with the cooccurrence-based one derived from \mathbf{f}_{dist} .

There are two aspects that may be qualitatively assessed: why would two tags with different names be associated? and why would two tags with same name have a very low audio similarity.

In the two first columns of Table 4, we present the 10 Discogs tags that are most similar (according to \mathbf{f}_c) to MuMu tags while not having the same normalized name. As can be seen, when the names are different, it can be due to the following reasons:

- Two different names are used for the exact same concept: *M:bebop/D:bop*, *M:punk-pop/D:pop punk*, *M:movie scores/D:score*.
- Some genres were considered sufficiently similar to be grouped under the same tag name in one of the tag system while they were not in the other one e.g. *M:indie & lo-fi/D:lo-fi*, *M:r&b/D:rnb/swing*.
- One genre is a subgenre of the other: *M:electric blues/D:modern electric blues*, *M:modern postbebop/D:genre:jazz*.

The association between *M:singer-songwriters* (a subgenre of *M:rock*) with *D:folk rock* (a subgenre of *D:genre:rock*) seems to link quite similar concepts (which seems to be confirmed by the cooccurrence based similarity that is quite high). *M:electronica* and *D:leftfield* seem to be quite broad electronic genres: the span of the former and the lack of precise definition of the latter while both seem not intended for dancing could explain the asso-

ciation. The association *M:special interest/D:avantgarde* remains quite unclear, while the tags are quite vague.

In the two last columns of Table 4 are presented top 10 most similar tags between MuMu and Discogs according to the cooccurrence based similarity (excluding string matched pairs with basic normalization as in [26]). It can be seen that the top 10 for cooccurrences and the top 10 for audio similarity contains mostly different tags, with the exception of *M:electric blues* which is not mapped to the same Discogs tag: this tends to show that cooccurrence similarity is complementary to the audio-based similarity, and when cooccurrence information is available (overlap between dataset), using both similarities should provide the best analysis. This is confirmed with some MuMu/Discogs pairs such as *M:bebop/D:bop* and *M:post hardcore/D:post-hardcore* which seems to be perfect mapping and have very high audio similarity but very low (less than 0.1) cooccurrence similarity. The low cooccurrence similarity may be explained by a lack of data for these tags.

On the other hand, it is also interesting to check tags with the exact same name in both datasets, but with quite low similarity score: the tags *electronic*, *instrumental* have very low similarity (according to both \mathbf{f}_c and \mathbf{f}_{dist}) from one database to another. *D:electronic* refers to a generic term for describing all electronic music while this exact same concept seemed to be carried by *M:dance & electronic* in MuMu. *M:electronic* is actually a subgenre of *M:progressive* which is a subgenre of *M:rock* and then has a very different meaning than the one in Discogs. *instrumental* (which is not a genre by itself) is considered a subgenre of *M:new age* and *M:country* in the MuMu taxonomy and a subgenre of *D:hip hop* in Discogs (while a large number of non-hip-hop songs seems to have the *D:instrumental* tag).

Thus, audio made it possible to spot significantly different genres that were represented by the exact same string. This highlights that the meaning of some genre may vary significantly from a database to another and that string matching can result in wrongly matched concepts.

5. CONCLUSION

In this paper we presented a way of learning genre embeddings from audio and showed that they are able to encode semantic similarities between genre tags: we showed that these embeddings were able to build genre taxonomies, to spot duplicates in a dataset or to translate genre from one tag set to another one. In future works, we plan to explore extraction of structured representation of other tag types than genre (mood, instruments, country...) from audio and to exploit other datasets such as the FMA dataset or GAS to learn a more global representation. We also plan to explore in more details how we can use several sources (audio, expert based ontology, string matching, cooccurrences) to build richer representation from flat tag systems.

6. ACKNOWLEDGEMENTS

The authors would like to thank Guillaume Salha for fruitful conversations and Matt Mould for proof-reading.

7. REFERENCES

- [1] Dmitry Bogdanov and Xavier Serra. Quantifying Music Trends and Facts Using Editorial Metadata From the Discogs Database. In *International Society for Music Information Retrieval Conference*, pages 89–95, 2017.
- [2] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Empirical Methods in Natural Language Processing*, 2014.
- [3] Kahyun Choi, Jin Ha Lee, and J. Stephen Downie. What is this song about anyway?: Automatic classification of subject using user interpretations and lyrics. In *ACM/IEEE Joint Conference on Digital Libraries*, pages 453–454. IEEE, sep 2014.
- [4] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. In *International Society for Music Information Retrieval Conference*, pages 805–811, 2016.
- [5] François Chollet. Keras: Deep learning library for theano and tensorflow, 2015.
- [6] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. FMA: A Dataset For Music Analysis. In *International Society for Music Information Retrieval Conference*, 2017.
- [7] Dennis Diefenbach, Pierre René Lhérisson, Fabrice Muhlenbach, and Pierre Maret. Computing the semantic relatedness of music genres using semantic web data. In *CEUR Workshop*, volume 1695, 2016.
- [8] Sander Dieleman, Philémon Brakel, and Benjamin Schrauwen. Audio-based Music Classification with a Pretrained Convolutional Network. In *International Society for Music Information Retrieval Conference*, pages 669–674, 2011.
- [9] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6964–6968, 2014.
- [10] Martín Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2015.
- [11] Arthur Flexer. a Closer Look on Artist Filters for Musical Genre Classification. In *IsMir 07*, pages 341–344, 2007.
- [12] Jort F Gemmeke, Daniel P.W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio Set: An ontology and human-labeled dataset for audio events. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 776–780, 2017.
- [13] Xiao Hu, JS Downie, Kris West, and AF Ehmann. Mining Music Reviews: Promising Preliminary Results. In *International Society for Music Information Retrieval Conference*, pages 536–539, 2005.
- [14] Şefki Koložali, Mathieu Barthet, György Fazekas, and Mark Sandler. Automatic ontology generation for musical instruments based on audio analysis. *IEEE Transactions on Audio, Speech and Language Processing*, 21(10):1–14, 2013.
- [15] Tao Li and Midsunori Ogihara. Music genre classification with taxonomy. In *Acoustics, Speech, and Signal Processing*, pages 197–200, 2005.
- [16] Janis Libeks and Douglas Turnbull. You can judge an artist by an album cover: Using images for music annotation. *IEEE Multimedia*, 18(4):30–37, apr 2011.
- [17] Rudolf Mayer, Robert Neumayer, and Andreas Rauber. Rhyme and Style Features for Musical Genre Classification By Song Lyrics. In *International Society for Music Information Retrieval Conference*, pages 337–342, 2008.
- [18] Robert Neumayer and Andreas Rauber. Integration of Text and Audio Features for Genre Classification in Music Information Retrieval. In *Advances in Information Retrieval*, pages 724–727. 2007.
- [19] Sergio Oramas, Luis Espinosa-anke, Aonghus Lawlor, Xavier Serra, Horacio Saggion, Music Technology Group, Universitat Pompeu Fabra, and Universitat Pompeu Fabra. Exploring Customer Reviews for Music Genre Classification and Evolutionary Studies. In *International Society for Music Information Retrieval Conference*, 2016.
- [20] Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra. Multi-label Music Genre Classification from Audio, Text, and Images Using Deep Features. In *International Society for Music Information Retrieval Conference*, 2017.
- [21] François Pachet and Daniel Cazaly. A Taxonomy of Musical Genres. In *Content-Based Multimedia Information Access Conference*, pages 1238–1245, 2000.
- [22] Elias Pampalk, Arthur Flexer, and Gerhard Widmer. Improvements of Audio-Based Music Similarity and Genre Classification. In *IsMir*, volume 5, pages 634–637, 2005.
- [23] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *International Workshop on Content-Based Multimedia Indexing*, volume 2016-June, pages 1–6. IEEE, jun 2016.
- [24] Chris Sanden and John Z. Zhang. Enhancing Multi-label Music Genre Classification Through Ensemble Techniques. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 705–714, New York, New York, USA, 2011. ACM Press.

- [25] Alexander Schindler and Andreas Rauber. An Audio-Visual Approach to Music Genre Classification through Affective Color Features. In *European Conference on Information Retrieval*, pages 61–67, 2015.
- [26] Hendrik Schreiber. Improving genre annotations for the million song dataset. In *16th International Society for Music Information Retrieval Conference*, pages 241–247, 2015.
- [27] Hendrik Schreiber. Genre Ontology Learning: Comparing Curated With Crowd-Sourced Ontologies. *17th International Society for Music Information Retrieval Conference*, pages 400–406, 2016.
- [28] Mohamed Sordo, Oscar Celma, Martín Blech, and Enric Guaus. The Quest for Musical Genres: Do the Experts and the Wisdom of Crowds Agree? In *9th International Conference on Music Information Retrieval*, pages 255–260, 2008.
- [29] Bob L. Sturm. Aalborg Universitet Classification Accuracy Is Not Enough Classification Accuracy Is Not Enough On the Analysis of Music Genre Recognition Systems. *Journal of Intelligent Information Systems*, 41(3):371–406, 2013.
- [30] George Tzanetakis and Perry Cook. Musical Genre Classification of Audio Signals. *IEEE Transactions On Speech And Audio Processing*, 10(5), 2002.
- [31] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.
- [32] Matthew D Zeiler. ADADELTA: An Adaptive Learning Rate Method, 2012.
- [33] Mu Zhu. Recall, precision and average precision, 2004.

LEARNING DOMAIN-ADAPTIVE LATENT REPRESENTATIONS OF MUSIC SIGNALS USING VARIATIONAL AUTOENCODERS

Yin-Jyun Luo and Li Su

Institute of Information Science, Academia Sinica, Taiwan

{freedomluo, lisu}@iis.sinica.edu.tw

ABSTRACT

In this paper, we tackle the problem of domain-adaptive representation learning for music processing. Domain adaptation is an approach aiming to eliminate the distributional discrepancy of the modeling data, so as to transfer learnable knowledge from one domain to another. With its great success in the fields of computer vision and natural language processing, domain adaptation also shows great potential in music processing, for music is essentially a highly-structured semantic system having domain-dependent information. Our proposed model contains a Variational Autoencoder (VAE) that encodes the training data into a latent space, and the resulting latent representations along with its model parameters are then reused to regularize the representation learning of the downstream task where the data are in the other domain. The experiments on cross-domain music alignment, namely an audio-to-MIDI alignment, and a monophonic-to-polyphonic music alignment of singing voice show that the learned representations lead to better higher alignment accuracy than that using conventional features. Furthermore, a preliminary experiment on singing voice source separation, by regarding the mixture and the voice as two distinct domains, also demonstrates the capability to solve music processing problems from the perspective of domain-adaptive representation learning.

1. INTRODUCTION

Music is composed, arranged, and performed in various forms residing in different data modalities and domains, yet sharing some common underlying information with each other. Almost all of the music processing tasks essentially extract such commonality as a protocol that enables the transferring or communication among various domains. For example, a piece of music can be either written as a musical score, or rendered as an audio recording; though the later encompasses much more information such as intonation, articulation, emotion, and others not found in the former, they still share common information such

as note, pitch, and meter with each other. In light of this property, we devise a framework that aims at eliminating domain-dependent information, to achieve a feature representation that is semantically shared across domains.

In this paper, we study learning representations that embed shared semantic information across different domains, specifically in applications of music signal processing. In order to achieve domain-invariant feature representations, we are essentially considering a domain adaptation problem [28]. Take audio-to-MIDI alignment [30] as an example, while audio and MIDI data are drawn from distinct domains of representation, they share pitch information in common. We explore the transfer learning technique [25] to tackle the problem. Specifically, in addition to transferring model parameters, we also transfer latent representations from one domain to the other.

With its success in computer vision [24, 28, 34] and natural language processing [16, 19, 32], transfer learning has also shown great potential in music information retrieval (MIR). In [6], a linear transformation is learned to project data into a shared latent representation that captures semantic similarity of music. Choi *et al.* uses feature maps of multiple layers derived from a pre-trained convolutional neural network (CNN) for music classification and regression tasks [2], and Park *et al.* exploits the deep model trained for artist recognition as a general feature extractor used for various tasks [26].

Our proposed framework¹ is different from the above-mentioned works. With pairwise training data² from two distinct domains, our framework first utilizes a VAE [12], a state-of-the-art unsupervised generative model shown to be effective in representation learning [9, 14], to embed information of data from one domain (the *source domain*) which contains mostly shared semantics into latent representations. Data from the other domain (i.e., the *target domain*) is then mapped to the learned embeddings through a separate neural network, in order to eliminate domain-dependent information. Therefore, the novelty of this paper is a unified framework that combines representation learning and transfer learning altogether, which learns domain-adaptive representations with VAEs that are then transferred from source to target domain. In particular, we empirically validate the framework through three



© Yin-Jyun Luo and Li Su. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Yin-Jyun Luo and Li Su. "Learning Domain-adaptive Latent Representations of Music Signals Using Variational Autoencoders", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ <https://github.com/yjlolo/Domain-Adaptive-VAE>

² Pairwise data in the context means parallel music events in different domains, e.g., a piece of music written as a score or rendered as an audio, and a recording of singing voice with or without accompaniment.

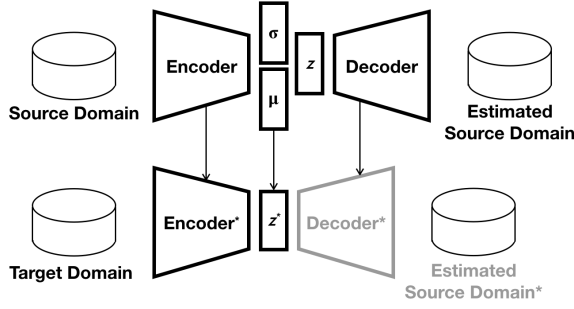


Figure 1. The general architecture of the proposed training framework.

well-known tasks in music signal processing that have not been considered from the perspective of domain adaptation: audio-to-MIDI alignment [1, 13], audio-to-audio alignment [17, 22], and singing voice separation [5, 10].

The rest of the paper is organized as follows. In Section 2, we describe our proposed architecture. The experiments and results are detailed in Section 3 and Section 4, respectively. Conclusions and future work are presented in Section 5.

2. ARCHITECTURE

2.1 Overview

Figure 1 shows our proposed framework in the training phase, which is divided into two modules: the first is a VAE which models the data in source domain, and the second, which can be either an autoencoder (AE) or simply an encoder depending on tasks, models the data in target domain. To facilitate the discussion, we refer *Encoder* (or *Decoder*) and *Encoder** (or *Decoder**) to the *source-domain encoder* (or *source-domain decoder*) and *target-domain encoder* (or *target-domain decoder*), respectively.

The two models are trained sequentially in two steps. First, we train the VAE, using the source-domain data as inputs, and obtain the source-domain latent representations $z := z(\mu, \sigma)$. More specifically, given the observation data x in the source domain, and $z \sim p(z)$ the latent representation, the posterior distribution $p(z|x)$ is modeled as a Gaussian distribution parameterized by the estimated mean and standard deviation of the posterior distribution, namely μ and σ , respectively. In other words, we have $p(z) = \mathcal{N}(z; \mu, \sigma^2 I)$ in practice.

Second, after the source-domain model is trained, we train the target-domain model, with the following two transfer learning schemes: 1) the source-domain model parameters are used to initialize the target-domain model parameters, and 2) the source-domain latent representation z is used to regularize the target-domain latent representation z^* with a regularization term $\mathcal{L}(z, z^*)$. The intuition behind this is to leverage knowledge learned by the source-domain VAE to reduce the distributional discrepancy between the source and target domain.

The target-domain decoder, colored in gray in Figure 1, is optional. For example, in the task of music alignment,

	Conv1	Conv2	Conv3	Fc1	Gauss
#filters/units	64	128	256	512	L
filter size	$1 \times F$	3×1	2×1	-	-
stride	(1,1)	(2,1)	(2,1)	-	-

Table 1. Encoder network architecture. *Conv* refers to convolutional layers, *Fc* refers to fully connected layers, and *Gauss* refers to the Gaussian parametric layer modeling z .

our purpose is to learn the domain-adaptive features by mapping the data in target domain into the feature distribution of data in source domain, without the need to reconstruct the input data from the latent representation.

It should be noticed that in the inference phase, shown in the left-hand side of Figure 2, the parameter μ is regarded as z . That is, when encoding the source-domain data, μ , the center of a Gaussian distribution, is the representative of z . Therefore, μ is the true latent representation that is transferred to the target domain. More details about the models and experiments are in Section 3.

2.2 Source-domain Model: Variational Autoencoder

Since the source-domain VAE is task-independent, we introduce its detailed architecture first in this subsection, and the task-dependent target-domain model will be introduced later in Section 3. We adopt the VAE architecture proposed in [9], which learns the latent representations and models the generative process of speech segments for voice conversion. In our work, the source-domain input representation is either a segment of singing voice in the tasks of singing voice alignment and separation, or a piano roll for audio-to-MIDI alignment.

The input x of the source-domain VAE is a two-dimensional image of size $T \times F$, where T is the number of time steps and F is the number of frequency bands. The encoder network of this VAE is a CNN with 3 convolution (*Conv*) layers and 1 fully-connected (*Fc*) layer that outputs the latent representation z with dimension L at the *Gauss* layer. The parameters of this CNN are summarized in Table 1. The decoder network is symmetric to the encoder network; it takes z as the input to reconstruct x . Batch normalization followed by the activation function \tanh are used for every layer except for the *Gauss* and the output layers. The objective function for training the VAE is expressed as (1):

$$\mathcal{L}_{vae} = \mathcal{L}_{rec} + \mathcal{L}_{KL}, \quad (1)$$

where the total loss function of the VAE, \mathcal{L}_{vae} , contains two terms: the reconstruction loss function $\mathcal{L}_{rec} = -\mathbb{E}_{q(z|x)}[\log p(x|z)]$, the negative expected log-likelihood of x , and the KL-Divergence loss $\mathcal{L}_{KL} = \text{KL}[q(z|x) || p(z)]$, which regularizes the distance between the posterior and the Gaussian distribution. In variational inference, the true posterior $p(z|x)$ is approximated by $q(z|x)$. For more implementation details of the VAE, we refer the readers to [3, 12].

3. EXPERIMENTS

We discuss the following three tasks: 1) audio-to-MIDI alignment, 2) audio-to-audio alignment, and 3) singing voice separation. In this section, we elaborate the goals, the datasets, the task-dependent target-domain models, the input data representations, and the evaluation processes for each of these three tasks. Experiment results will be discussed in Section 4.

All of the models discussed in the following are implemented with `PyTorch` [27], and are trained using stochastic gradient descent with the Adam optimizer [11]. The optimizer is parametrized by: learning rate = 10^{-3} , $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The mini-batch size is set to 128 instances of input segments.

3.1 Task 1: Audio-to-MIDI Alignment

The first experiment we consider is to align an audio recording of piano to its corresponding MIDI file. Although this is a rather well-studied task [13, 23, 30], we re-investigate this task from the perspective of domain adaptation: using the learned latent representations for the feature on which dynamic time warping (DTW) is performed.

3.1.1 Dataset

We use a subset of the MAPS dataset [4], *ENSTD-kCI*, which contains 30 piano recordings performed by a Yamaha Disklavier auto-piano together with MIDI files that generate the recordings. We use 24 and 6 pieces of the subset for training and validation, respectively.

3.1.2 Model

The goal of our framework is to map a frame of audio feature and piano roll into the same representation if they are of the same music event. To do this, we first define the MIDI pieces as the source domain data, and the audio pieces as the target-domain data. Then, we train the source-domain VAE and obtain the learned source-domain latent representation z . We then use this representation z as the learning target to train the target-domain model, a single encoder taking audio data as input, with its architecture the same as the source-domain encoder. To be more specific: given a pair of MIDI-audio input data that are of the same event in the music, the source-domain VAE maps the MIDI into a representation in a low-dimensional Gaussian distribution, and the target-domain encoder is then trained to map the audio input data to that distribution.

The learning task in the target domain is essentially a regression task. The training objective function for source domain is the same as (1), while the objective function for the target domain $\mathcal{L}_{encoder}$ is

$$\mathcal{L}_{encoder} = \mathcal{L}_{MSE}(z, z^*), \quad (2)$$

which is the mean squared error between the encoded latent representations of the source-domain encoder z and the ones of the target-domain encoder z^* . Notice that (2) is only applicable when we have parallel source-target pairs.

Instead of audio, MIDI is regarded as the source-domain data because the latent representation we obtain should be more related to MIDI which contains mostly the shared semantics with audio, i.e., pitch; we let the target-domain encoder eliminate the information residing in audio while unrelated to MIDI (e.g., spectral-related information) in order to get succinct representations for alignment.

3.1.3 Data Representation

MIDI files are represented as piano-roll representation with 128 pitch classes, while its associated audio recordings are represented using Mel-scaled spectrogram with 128 filter banks, derived from power magnitude spectrum of 1024-point short-time Fourier transform (STFT). To compute the STFT, we use Hanning window with window size of 64 ms and hop size of 20 ms. An input data for the source-domain VAE (or the target-domain encoder) is a segment of a piano-roll (or Mel-scaled spectrogram) with 21 frames, or equivalently, 400 ms, leading to the input dimensions of $T = 21$ and $F = 128$. To reduce the memory load, we only collect segments every 10 frames of each clip for training.

3.1.4 Evaluation

To evaluate our proposed feature representation for audio-to-MIDI alignment, we apply non-linear time-stretching to the audio recordings so as to see if the features are robust against the distortion and can still be aligned to the original MIDI well. We follow the methodology in [17] for non-linear time-stretch.

The proposed feature representation of MIDI can be derived as follows: we express MIDI as piano roll and use it as the input to the source-domain encoder to obtain the encoded latent representation as our proposed feature; the process is illustrated in Figure 3 with the solid blue line. On the other hand, in the target domain, we firstly apply time-stretching distortion to the audio recordings, represent the audio stream with Mel-scaled spectrogram described in Section 3.1.3, and utilize the outputs of the target-domain encoder as our final feature representation; the green solid line in Figure 3 describes the process. Overall, the derivation of the proposed feature representation during inference is illustrated in the left panel of Figure 2.

For comparison, we consider chroma as a baseline to represent both domains, illustrated in Figure 3 with the loose dash lines colored in blue and green, respectively. Regarding the implementation of chroma, we use `chroma_stft` in the `librosa` library [18] for audio and `get_chroma` in the `pretty_midi` library [29] for MIDI. The other baseline is to use the piano roll for MIDI and Mel-scaled spectrogram for audio, illustrated in Figure 3 with the dense dash lines colored in blue and green, respectively.

We utilize DTW to align the feature representations and compute the alignment accuracy. The accuracy is calculated by an error measure e which compares the discrepancy between the estimated warping path and the ground-truth one [36] instead of the conventional note-level alignment accuracy, because the error measure e allows more subtle comparison on frame-level evaluation.

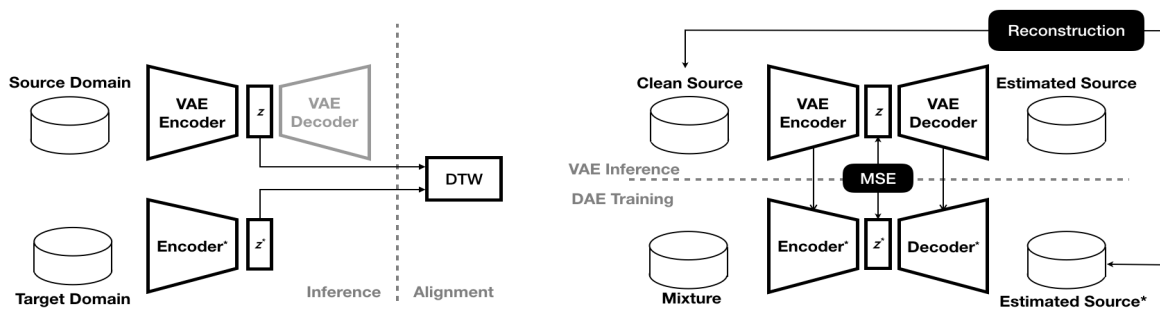


Figure 2. Left: the derivation of the proposed feature representation in *task 1* and *task 2*. Right: the training scheme of the target-domain DAE in *task 3*.

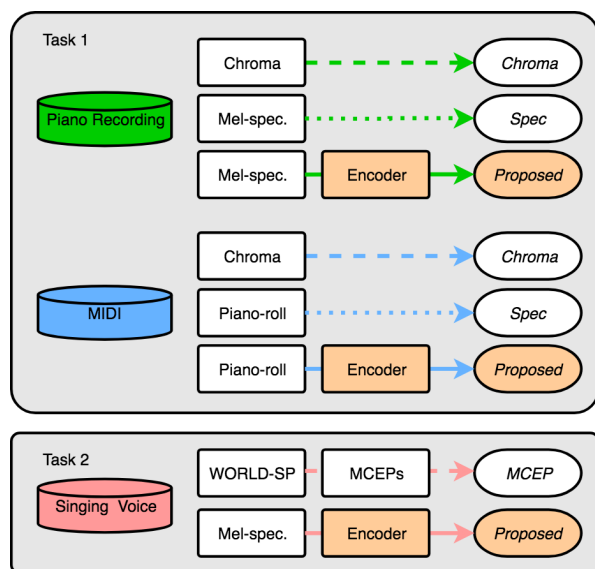


Figure 3. Extraction pipelines of feature representations for the two alignment tasks, i.e., *task 1* and *task 2*.

3.2 Task 2: Audio-to-Audio Alignment

In this experiment, we consider singing voice alignment, in particular the alignment of song recordings performed by singers with their artificially-distorted versions. Specifically, two subtasks are considered: 1) aligning the distorted monophonic singing recordings to the original version, denoted as *mono-to-mono*, and 2) the distorted monophonic singing recordings to the original singing recordings mixed with the corresponding background music, denoted as *mono-to-poly*. The goal is to demonstrate the robustness of our proposed feature representation against the artificial distortion effects, i.e., pitch-shift and time-stretch, as well as interference of the background music.

3.2.1 Dataset

We adopt the MIR-1k dataset [8] which contains the 1,000 Chinese karaoke excerpts with separated voice and accompaniment tracks, clipped from 110 songs. We then divided the 110 songs into two subsets, one containing 88 songs for training, and the other containing 22 songs for validation.

3.2.2 Model

The training procedure resembles the one mentioned in Section 3.1.2. The difference is that the source domain refers to monophonic singing, and the target domain refers to its polyphonic version; the shared information is the singing voice. Notice that, similar to Section 3.1, the synthetic dataset with artificial distortion is not used for training. The target-domain encoder for modeling polyphonic music learns not only to output features that are comparable to monophonic singing voice, but also features that are more robust to artificial distortion.

3.2.3 Data Representation

The inputs are represented the same way as the audio data described in Section 3.1.1. As suggested by the preliminary experiments, the number of filter banks is set to $F = 256$ instead of 128.

3.2.4 Evaluation

We evaluate our proposed feature representation under both time-stretching and pitch-shifting distortion. The settings of the distortion follow the one in [17].

As shown in Figure 3, for the subtask *mono-to-mono*, we first apply the artificial distortion to the monophonic singing, followed by a Z-score normalization. The Mel-scaled spectrogram is then extracted as the input to the source-domain encoder, which gives the proposed feature representation of monophonic singing after a post Z-score normalization. We align the distorted monophonic singing to the intact version. For the subtask *mono-to-poly*, we adopt the identical process to the monophonic singing. While for polyphonic singing, the target-domain encoder takes the input as the Mel-spectrogram to output the proposed feature representation. We align the distorted monophonic to the original polyphonic version. The overview of derivation of the proposed feature representation and alignment are illustrated in the left panel of Figure 2.

We compare our proposed feature representation with the 24-ordered Mel-cepstral coefficients (MCEPs) [33], a widely used features regarding speech alignment for voice conversion [20], in terms of the error measure e , as in Section 3.1.4. We use the spectral envelope which is extracted by WORLD [21] to derive MCEPs. DTW in this

task searches for the optimal alignment path according to squared Euclidean distance, as suggested by preliminary experimental results.

3.3 Task 3: Singing Voice Separation

Singing voice separation is an essential yet notoriously challenging problem in music signal processing; the goal is to separate singing voice from music mixture. We investigate the potential of domain adaptation on this problem.

3.3.1 Dataset

We again adopt the MIR-1k dataset for experiment, and split the dataset in a way identical to that in Section 3.2.1.

3.3.2 Model

The basic idea is a follow-up of the *mono-to-poly* scheme in Section 3.2: given the fact that we have obtained domain-adaptive latent representations shared across monophonic singing and its polyphonic version with accompaniment, one step further is to consider decoding the outputs of the target-domain encoder in order to reconstruct the monophonic singing voice in the target domain. Therefore, we adopt a Denoising Autoencoder (DAE) [35] in the target domain.

The training scheme of the target domain is illustrated in the right panel of Figure 2. It is important to note that, different from the vanilla DAE for source separation [5], we regularize the bottleneck layer with the learned latent representation encoded by the VAE along with the model parameters for weight initialization. The training objective function for the target domain therefore becomes:

$$\mathcal{L}_{DAE} = \mathcal{L}_{l_1}(\mathbf{x}, \tilde{\mathbf{x}}) + \alpha \mathcal{L}_{MSE}(\mathbf{z}, \mathbf{z}^*), \quad (3)$$

where the reconstruction loss \mathcal{L}_{l_1} denotes the l_1 -norm; \mathbf{x} and $\tilde{\mathbf{x}}$ are the clean source of singing voice and estimated one, respectively. α is the weight of the regularization term which is set to 1 without further investigation in this preliminary work.

3.3.3 Data Representation

For audio representation, the magnitude spectrogram instead of the Mel-scaled spectrogram is used as the input; the parameters for computation of STFT remain the same as in Section 3.1.3.

3.3.4 Evaluation

The music mixture which contains the ground-truth source of singing voice \mathbf{x} and background music is firstly normalized with a Z-score normalization, and is represented as the magnitude spectrogram. The trained DAE then takes as the input the magnitude spectrogram, and outputs the estimated source of singing voice $\tilde{\mathbf{x}}$.

For evaluation, we use `mir_eval` [31] to calculate and report source-to-distortion ratio (SDR), source-to-inference ratio (SIR), and source-to-artifact (SAR) ratio together with normalized SDR (NSDR). All scores are weighted by number of frames of each song. We compare the performance among vanilla DAE with or without

	error measure
<i>Proposed</i> ($L = 128$)	2.48
<i>Proposed</i> ($L = 12$)	4.08
<i>Chroma</i>	6.71
<i>Spec</i>	39.24

Table 2. The error measure e of audio-to-MIDI alignment using different feature representations.

our proposed regularization term and weight initialization during training phase.

4. RESULTS

In this section, we report the performance evaluated on the validation sets for each experiment.

4.1 Task 1: Audio-to-MIDI Alignment

Table 2 lists the median value of e , the alignment error measure, over the 6 audio-MIDI pairs in the validation set using four different feature representations: two of them are the proposed latent representations with dimensions $L = 128$ and 12 (*Proposed*), one is the 12-dimensional chroma (*Chroma*), and the other uses Mel-scaled spectrogram for audio and piano-roll representation for MIDI, both are 128-dimension (*Spec*). One can see our proposed domain-adaptive features outperform with both $L = 128$ and 12. This implies that the plane pitch information of MIDI domain is properly modeled in the latent representations by the source-domain encoder, and is efficiently transferred to audio domain by treating the latent representations as learning targets for the target-domain encoder.

4.2 Task 2: Audio-to-Audio Alignment

We evaluate on the validation set of 22 songs and report the alignment error measure e of different feature representations under the *mono-to-mono* and *mono-to-poly* sub-tasks, along with the artificial distortion in pitch-shift and linear/non-linear time-stretch. Figure 4 shows the median of the error measure e using different feature representations; the baseline feature and proposed one are denoted as *MCEP* and *Proposed*, respectively. Each individual plot shows the error measure along pitch-shift steps of -2, -1, 0, 1, and 2. The top panel and bottom panel refer to *mono-to-mono* and *mono-to-poly*, respectively. The leftmost to the fifth column correspond to linear time-stretching rates of 0.8, 0.9, 1.0, 1.1 and 1.2, respectively, while the rightmost column corresponds to the non-linear time-stretch.

The results of *mono-to-mono* in the top panel suggest that our proposed feature representation encoded by the source-domain encoder is more robust to the artificial distortion than the baseline feature. The bottom panel, which corresponds to *mono-to-poly*, shows that by transferring the latent representations from source to target domain, the target-domain encoder indeed learns to output features that are robust against both the artificial distortion and the interference of background music.

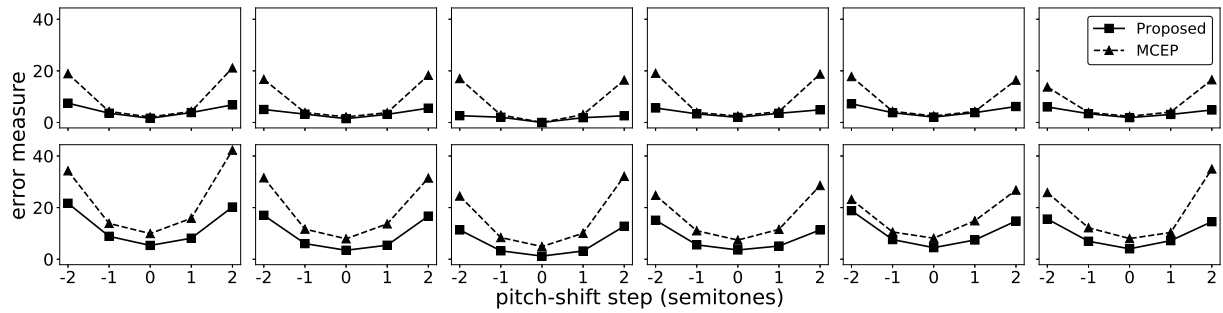


Figure 4. The error measure e of singing voice alignment using our proposed features or MCEPs. Top panel: *mono-to-mono*; bottom panel: *mono-to-poly*. The leftmost column to the fifth column refer to time-stretch rate $r = 0.8, 0.9, 1.0, 1.1$, and 1.2 , respectively; the rightmost column refers to non-linear time-stretch.

	SDR	SIR	SAR	NSDR
DAE	4.73	16.13	5.35	3.16
$DAE + wt$	6.50	20.40	6.85	4.93
$rDAE$	4.97	14.96	5.74	3.40
$rDAE + wt$	7.20	18.98	7.74	5.63

Table 3. The source-to-distortion ratio (SDR), source-to-inference ratio (SIR), and source-to-artifact ratio (SAR) and normalized SDR (NSDR) of different models.

4.3 Task 3: Singing Voice Separation

Table 3 demonstrates the SDR, SIR, and SAR together with NSDR of different models in the task of singing voice separation. Four models are compared: 1) DAE referring to the vanilla DAE, the baseline model, 2) $DAE + wt$ denoting the DAE trained with weight initialization using the source-domain model parameters, 3) $rDAE$ referring to the DAE trained with the objective function whose weight of the regularization term $\alpha = 1$ in (3), and 4) $rDAE + wt$, the DAE trained with both the weight initialization and regularization term.

From the SDR in Table 3, one can observe that $DAE + wt$ outperforms DAE by 1.77 dB, while $rDAE$ outperforms DAE by only 0.24 dB. However, by combining weight initialization and regularization together, $rDAE + wt$ achieves an improvement of 2.47 dB over DAE . This implies that the effect of transferring the latent representation from the source to target domain as a regularization term can be optimized by the transfer of the source-domain model parameters.

Notice that though the reported performance is not on par with the state-of-the-art method [10], our model still show potentials in solving the singing voice separation problem from the perspective of domain adaptation. Meanwhile, as a preliminary work, we evaluate the framework on a relatively small dataset without data augmentation and fine-tuning parameters.

5. CONCLUSION AND FUTURE WORK

In this paper, we re-investigate three well-known tasks of music signal processing from the perspective of domain adaptation, namely *task 1*: audio-to-MIDI alignment, *task 2*: audio-to-audio alignment and *task 3*: singing voice separation. To this end, we devise a unified framework that achieve both representation learning and transfer learning at once. Specifically, we use a VAE to learn latent representation of source-domain data, which is then transferred to train a separate model that maps target-domain data to the representation.

We empirically validate our idea by demonstrating the superiority of our proposed feature representations over baseline ones across all the tasks. In both *task 1* and 2, the proposed features are shown to properly model the source-domain data and are efficiently transferred to the target domain; they are more robust against various settings of artificial distortion compared to baseline features. In *task 3*, it is shown that transferring of both model parameters and latent representations, used for weight initialization and as a regularization term, respectively, can benefit the performance of singing voice separation, which indicates the potential of the framework for such a challenging problem.

As a preliminary work, though we share most of the parameters and model architectures across all the tasks without tailoring for each individual task, the proposed framework consistently outperforms the baselines. For future work, we would like to include larger datasets and optimize the system architectures and their parameters. Moreover, expanding the framework for classification is of particular interest. For example, it is possible to transfer the latent representation from source to target domain by directly leveraging it as the classifying feature [15] or intermediate condition to models in target domain [7].

6. ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work is partially supported by MOST Taiwan, under the contract MOST 106-2218-E-001-003-MY3.

7. REFERENCES

- [1] J. J. Carabias-Orti, F. J. Rodríguez-Serrano, P. Vera-Candeas, N. Ruiz-Reyes, and F. J. Cañadas-Quesada. An Audio to Score Alignment Framework Using Spectral Factorization and Dynamic Time Warping. In *ISMIR*, pages 742–748, 2015.
- [2] K. Choi, G. Fazekas, M. Sandler, and K. Cho. Transfer Learning for Music Classification and Regression Tasks. In *ISMIR*, 2017.
- [3] C. Doersch. Tutorial on Variational Autoencoders. *ArXiv e-prints*, June 2016.
- [4] V. Emiya, R. Badeau, and B. David. Multipitch Estimation of Piano Sounds using a New Probabilistic Spectral Smoothness Principle. *IEEE Trans. on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.
- [5] E. M. Grais and M. D. Plumbley. Single Channel Audio Source Separation using Convolutional Denoising Autoencoders. *ArXiv e-prints*, March 2017.
- [6] P. Hamel, M. E. Davies, K. Yoshii, and M. Goto. Transfer Learning in MIR: Sharing Learned Latent Representations for Music Audio Classification and Similarity. In *ISMIR*, 2013.
- [7] J. A. Hennig, A. Umakantha, and R. C. Williamson. A Classifying Variational Autoencoder with Application to Polyphonic Music Generation. *ArXiv e-prints*, November 2017.
- [8] C.-L. Hsu and J.-S. R. Jang. On the Improvement of Singing Voice Separation for Monaural Recordings using MIR-1k Dataset. *TASLP*, 18(2):310–319, 2010.
- [9] W.-N. Hsu, Y. Zhang, and J. Glass. Learning Latent Representations for Speech Generation and Transformation. In *Interspeech*, pages 1273–1277, 2017.
- [10] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde. Singing Voice Separation With Deep U-Net Convolutional Networks. In *ISMIR*, 2017.
- [11] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, December 2014.
- [12] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, December 2013.
- [13] T. Kwon, D. Jeong, and J. Nam. Audio-to-score Alignment of Piano Music Using RNN-based Automatic Music Transcription. *ArXiv e-prints*, November 2017.
- [14] S. Latif, R. Rana, J. Qadir, and J. Epps. Variational Autoencoders for Learning Latent Representations of Speech Emotion: A Preliminary Study. *ArXiv e-prints*, December 2017.
- [15] S. Latif, R. Rana, J. Qadir, and J. Epps. Variational Autoencoders for Learning Latent Representations of Speech Emotion: A Preliminary Study. *ArXiv e-prints*, December 2017.
- [16] Q. Li. Literature Survey: Domain Adaptation Algorithms for Natural Language Processing. Technical report, Department of Computer Science The Graduate Center, The City University of New York, 2012.
- [17] Y.-J. Luo, M.-T. Chen, T.-S. Chi, and L. Su. Singing Voice Correction using Canonical Time Warping. *ArXiv e-prints*, November 2017.
- [18] B. McFee, M. McVicar, O. Nieto, S. Balke, C. Thome, D. Liang, E. Battenberg, J. Moore, R. Bittner, R. Yamamoto, and et al. librosa 0.5.0, February 2017.
- [19] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *ArXiv e-prints*, January 2013.
- [20] S. H. Mohammadi and A. Kain. An Overview of Voice Conversion Systems. *Speech Communication*, 2017.
- [21] M. Morise, F. Yokomori, and K. Ozawa. WORLD: A Vocoder-based High-quality Speech Synthesis System for Real-time Applications. *IEICE Trans. Information and Systems*, 99(7):1877–1884, 2016.
- [22] M. Müller, F. Kurth, and M. Clausen. Audio Matching via Chroma-Based Statistical Features. In *ISMIR*, page 6th, 2005.
- [23] M. Müller, F. Kurth, and T. Röder. Towards an Efficient Algorithm for Automatic Score-to-Audio Synchronization. In *ISMIR*, 2004.
- [24] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and Transferring Mid-level Image Representations using Convolutional Neural Networks. In *CVPR*, pages 1717–1724. IEEE, 2014.
- [25] S. J. Pan and Q. Yang. A Survey on Transfer Learning. *IEEE Trans. on knowledge and data engineering*, 22(10):1345–1359, 2010.
- [26] J. Park, J. Lee, J. Park, J.-W. Ha, and J. Nam. Representation Learning of Music Using Artist Labels. *ArXiv e-prints*, October 2017.
- [27] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic Differentiation in PyTorch. In *NIPS-W*, 2017.
- [28] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa. Visual Domain Adaptation: A survey of Recent Advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.
- [29] C. Raffel and D. P. W. Ellis. Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty_midi. In *ISMIR Late Breaking and Demo Papers*, 2014.

- [30] C. Raffel and D. P. W. Ellis. Optimizing DTW-based Audio-to-MIDI Alignment and Matching. In *ICASSP*, pages 81–85. IEEE, 2016.
- [31] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. mir_eval: A Transparent Implementation of Common MIR Metrics. In *ISMIR*, 2014.
- [32] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML*, pages 759–766. ACM, 2007.
- [33] K. Tokuda, T. Kobayashi, T. Masuko, and S. Imai. Mel-generalized Cepstral Analysis-A Unified Approach to Speech Spectral Estimation. In *International Conference on Spoken Language Processing*, 1994.
- [34] T. Tommasi, N. Quadrianto, B. Caputo, and C. H. Lampert. Beyond Dataset Bias: Multi-task Unaligned Shared Knowledge Transfer. In *Asian Conference on Computer Vision*, pages 1–15. Springer, 2012.
- [35] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.
- [36] F. Zhou and F. De la Torre. Generalized Time Warping for Multi-modal Alignment of Human Motion. In *CVPR*, pages 1282–1289, 2012.

LEARNING TRANSPOSITION-INVARIANT INTERVAL FEATURES FROM SYMBOLIC MUSIC AND AUDIO

Stefan Lattner^{1,2}, Maarten Grachten^{1,2}, Gerhard Widmer¹

¹ Institute of Computational Perception, JKU Linz

² Sony Computer Science Laboratories (CSL), Paris, France

ABSTRACT

Many music theoretical constructs (such as scale types, modes, cadences, and chord types) are defined in terms of pitch intervals—relative distances between pitches. Therefore, when computer models are employed in music tasks, it can be useful to operate on interval representations rather than on the raw musical surface. Moreover, interval representations are transposition-invariant, valuable for tasks like audio alignment, cover song detection and music structure analysis. We employ a gated autoencoder to learn fixed-length, invertible and transposition-invariant interval representations from polyphonic music in the symbolic domain and in audio. An unsupervised training method is proposed yielding an organization of intervals in the representation space which is musically plausible. Based on the representations, a transposition-invariant self-similarity matrix is constructed and used to determine repeated sections in symbolic music and in audio, yielding competitive results in the MIREX task “Discovery of Repeated Themes and Sections”.

1. INTRODUCTION

The notion of relative pitch is important in music understanding. Many music theoretical concepts, such as scale types, modes, chord types and cadences, are defined in terms of relations between pitches or pitch classes. But relative pitch is not only a music theoretical construct. It is common for people to perceive and memorize melodies in terms of pitch intervals (or in terms of *contours*, the upward or downward *direction* of pitch intervals) rather than sequences of absolute pitches. This characteristic of music perception also has ramifications for the perception of form in musical works, since it implies that transposition of some musical fragment along the pitch dimension (such that the relative distances between pitches remain the same) does not alter the perceived identity of the musical material, or at least establishes a sense of similarity between the original and the transposed material. As such, adequate detection of musical form in terms of (approx-

mately) repeated structures presupposes the ability to account for pitch transposition—one of the most common types of transformations found in music.

Relative pitch perception in humans is currently not well-understood [13]. For example there are no established theories on how the human brain derives a relative representation of pitch from the tonotopic representations formed in the cochlea, neither is it clear whether there is a connection between the perception of pitch relations in simultaneous versus consecutive pitches.

Computational approaches to address tasks of music understanding (such as detecting patterns and form in music) often circumvent this issue by representing musical stimuli as sequences of monophonic pitches, after which simply differencing consecutive pitches yields a relative pitch representation. This approach also works for polyphonic music, to the extent that the music can be meaningfully segregated into monophonic pitch streams. A drawback of this approach is that it presupposes the ability to segregate musical streams, which is often far from trivial due to the ambiguity of musical contexts. To take an analogous approach on acoustical representations of musical stimuli is even more challenging, since it further depends on the ability to detect pitches and onsets in sound.

In this paper we take a different approach altogether. We train a neural network model to learn representations that represent the relation between the music at some time point t and the preceding musical context. During training, these representations are adapted to minimize the reconstruction error of the music at t given the preceding context and the representation itself.

A crucial aspect of the model is its bilinear architecture (more specifically, a *gated autoencoder*, or GAE architecture) involving multiplicative connections, which facilitates the formation of relative pitch representations. We stimulate such representations more explicitly using an altered training procedure in which we transpose the training data using arbitrary transpositions.

The result are two models (for symbolic music and audio) that can map both monophonic and polyphonic music to a sequence of points in a vector space—the *mapping space*—in a way that is invariant to pitch transpositions. This means that a musical fragment will be projected to the same mapping space trajectory independently of how it is transposed.

We validate our approach experimentally in several ways. First we show that musical fragments that are nearest neigh-



© Stefan Lattner, Maarten Grachten, Gerhard Widmer. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Stefan Lattner, Maarten Grachten, Gerhard Widmer. “Learning transposition-invariant interval features from symbolic music and audio”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

bors in the mapping space have many pitch intervals in common (as opposed to nearest neighbors in the input space). Then we show that the topology of the learned mapping space reflects musically meaningful relations between intervals (such as the tritone being dissimilar to other intervals). Lastly we use mapping space representations to detect musical form both for symbolic and audio representations of music, showing that it yields competitive results, and in the case of audio even improves the state of the art. A re-implementation of the transposition-invariant GAE for audio is publicly available¹.

The paper is structured as follows. Section 2 provides an overview of relation learning using GAEs, and reviews work on creating interval representations from music. In Section 3, the used architecture is described and in Section 4, data is introduced on which the GAE is trained. The training procedure, including the novel method to support the emergence of transposition-invariance, is proposed in Section 5. The experiments conducted to examine the properties of learned mappings are described in Section 6, and results are presented and discussed in Section 7. Section 8 wraps the paper up with conclusions and prospects of future work.

2. RELATED WORK

GAEs utilize *multiplicative interactions* to learn correlations between or within data instances. The method was inspired by the correlation theory of the brain [32], where it was pointed out that some cognitive phenomena cannot be explained with the conventional brain theory and an extension was proposed which involves the correlation of neural patterns.

In machine learning, this principle was deployed in *bi-linear* models, for example to separate person and pose in face images [30]. Bi-linear models, like the GAE, are two-factor models whose outputs are linear in either factor when the other is held constant. [26] proposed another variant of a bi-linear model in order to learn objects and their optical flow. Due to its similar architecture, the gated Boltzmann machine (GBM) [17, 18] can be seen as a direct predecessor of the GAE. The GAE was introduced by [14] as a derivative of the GBM, as standard learning criteria became applicable through the development of denoising autoencoders [31].

GAEs have been further used to learn transformation-invariant representations for classification tasks [15], for parent-offspring resemblance [5], for learning to negate adjectives in linguistics [27], for activity recognition with the Kinect sensor [22], in robotics to learn to write numbers [6], and for learning multi-modal mappings between action, sound, and visual stimuli [7].

In music, bi-linear models have been applied to learn co-variances within spectrogram data for music similarity estimation [28], and for learning musical transformations in the symbolic domain [9]. In sequence modeling, the GAE has been utilized to learn co-variances between sub-

sequent frames in movies of rotated 3D objects [16] and to predict accelerated motion by stacking more layers in order to learn higher-order derivatives [21], which uses a method similar to the one proposed here.

Transposition-invariance in music is achieved in [20] by transforming symbolic pitch-time representations into point-sets, in which translatable patterns are identified. Another method in the symbolic domain is that in [2], where a general interval representation for polyphonic music is put forward, in [24], where specific pitch-class intervals in polyphonic music are used for characterizing music styles and in [23] where transposition-invariant self-similarity matrices are computed. In [12], an approach to calculating transposition-invariant mid-level representations from audio is introduced, based on the 2-D power spectrum of melodic fragments. Similarly, a method to calculate interpretable interval representations from audio is proposed in [33], where chromagrams that are close in time are cross-correlated to obtain local pitch-invariance.

3. MODEL

Let \mathbf{x}_j be a vector representing pitches of currently sounding notes (in the symbolic domain) or the energy distributed over frequency bands (in the audio domain), in a fixed-length time interval. Given a temporal context $\mathbf{x}_{t-n}^t = \mathbf{x}_{t-n} \dots \mathbf{x}_t$ as the input and the next time step \mathbf{x}_{t+1} as the target, the goal is to learn a mapping \mathbf{m}_t which does not change when shifting \mathbf{x}_{t-n}^{t+1} up- or downwards in the pitch dimension. A gated autoencoder (GAE, depicted in Figure 1) is well-suited for this task, modeling the intervals between reference pitches in the input and pitches in the target, encoded in the latent variables of the GAE as mapping codes \mathbf{m}_j . Unlike in common prediction tasks, the targets are known when training a GAE. The goal of the training is to find a mapping \mathbf{m}_j for any input/target pair which transforms the input into the given target. The mapping at time t is calculated as

$$\mathbf{m}_t = \sigma_h(\mathbf{W}_1 \sigma_h(\mathbf{W}_0(\mathbf{U}\mathbf{x}_{t-n}^t \cdot \mathbf{V}\mathbf{x}_{t+1}))), \quad (1)$$

where \mathbf{U} , \mathbf{V} and \mathbf{W}_k are weight matrices, σ_h is the hyperbolic tangent non-linearity, and we will refer to the learnt mappings \mathbf{m}_j as the *mapping space* of the input/target pairs. The operator \cdot (depicted as a triangle in Figure 1) depicts the Hadamard (or element-wise) product of the filter responses $\mathbf{U}\mathbf{x}_{t-n}^t$ and $\mathbf{V}\mathbf{x}_{t+1}$, denoted as *factors*. This operation allows the model to *relate* its inputs, making it possible to learn interval representations.

The target of the GAE can be reconstructed as a function of the input \mathbf{x}_{t-n}^t and a mapping \mathbf{m}_t :

$$\tilde{\mathbf{x}}_{t+1} = \sigma_g(\mathbf{V}^\top (\mathbf{W}_0^\top \mathbf{W}_1^\top \mathbf{m}_t \cdot \mathbf{U}\mathbf{x}_{t-n}^t)), \quad (2)$$

where σ_g is the sigmoid non-linearity for binary input and the identity function for real-valued input.

The cost function is defined to penalize the error of reconstructing the target \mathbf{x}_{t+1} given the input \mathbf{x}_{t-n}^t and the

¹ see <https://github.com/SonyCSLParis/cgae-invar>

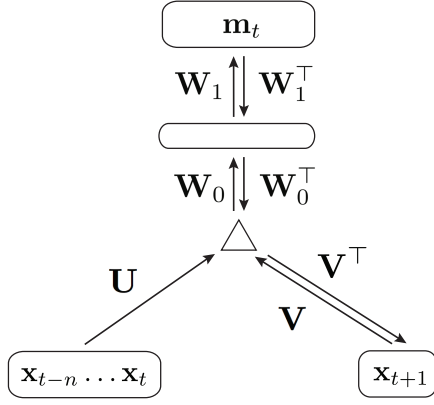


Figure 1: Schematic illustration of the gated autoencoder architecture used in the experiments.

mapping \mathbf{m}_t as

$$\mathcal{L}_c = c(\mathbf{x}_{t+1}, \tilde{\mathbf{x}}_{t+1}), \quad (3)$$

where $c(\cdot)$ is the mean-square error for real-valued sequences and the cross-entropy loss for binary sequences.

4. DATA

We train the model both on symbolic music representations and on audio spectrograms. For the symbolic data, the Mozart/Batik data set [35] is used, consisting of 13 piano sonatas containing more than 106,000 notes. The dataset is encoded as successive 60 dimensional binary vectors (encoding MIDI note number 36 to 96), each representing a single time step of 1/16th note duration. The pitch of an active note is encoded as a corresponding on-bit, and as multiple voices are encoded simultaneously, a vector may have multiple active bits. The result is a pianoroll-like representation.

The audio dataset consists of 100 random piano pieces of the MAPS dataset [8] (subset MUS), at a sampling rate of 22.05 kHz. We choose a constant-Q transformed spectrogram using a hop size of 1984, and Hann windows with different sizes depending on the frequency bin. The range comprises 120 frequency bins (24 per octave), starting from a minimal frequency of 65.4 Hz. Each time step is contrast-normalized to zero mean and unit variance.

5. TRAINING

The model is trained with stochastic gradient descent in order to minimize the cost function (cf. Equation 3) using the data described in Section 4. However, rather than using the data as is, we use data-augmentation in combination with an altered training procedure to explicitly aim at transposition invariance of the mapping codes.

5.1 Enforcing Transposition-Invariance

As described in Section 3 the classical GAE training procedure derives a mapping code from an input/target pair,

and subsequently penalizes the reconstruction error of the target given the input and the derived mapping code. Although this procedure naturally tends to lead to similar mapping codes for input target pairs that have the same interval relationships, the training does not explicitly enforce such similarities and consequently the mappings may not be maximally transposition invariant.

Under ideal transposition invariance, by definition the mappings would be identical across different pitch transpositions of an input/target pair. Suppose that a pair $(\mathbf{x}_{t-n}^t, \mathbf{x}_{t+1}^t)$ leads to a mapping \mathbf{m} (by Equation 1). Transposition invariance implies that reconstructing a target \mathbf{x}_{t+1}' from the pair $(\mathbf{x}_{t-n}^t, \mathbf{m})$ should be as successful as reconstructing \mathbf{x}_{t+1} from the pair $(\mathbf{x}_{t-n}^t, \mathbf{m})$ when $(\mathbf{x}_{t-n}^t, \mathbf{x}_{t+1}^t)$ can be obtained from $(\mathbf{x}_{t-n}^t, \mathbf{x}_{t+1}^t)$ by a single pitch transposition.

Our altered training procedure explicitly aims to achieve this characteristic of the mapping codes by penalizing the reconstruction error using mappings obtained from transposed input/target pairs. More formally, we define a transposition function $shift(\mathbf{x}, \delta)$, shifting the values of a vector \mathbf{x} of length M by δ steps (MIDI note numbers and CQT frequency bins for symbolic and audio data, respectively):

$$shift(\mathbf{x}, \delta) = (x_{(0+\delta) \bmod M}, \dots, x_{(M-1+\delta) \bmod M})^T, \quad (4)$$

and $shift(\mathbf{x}_{t-n}^t, \delta)$ denotes the transposition of each single time step vector *before* concatenation and linearization.

The training procedure is then as follows. First, the mapping code \mathbf{m}_t of an input/target pair is inferred as shown in Equation 1. Then, \mathbf{m}_t is used to reconstruct a *transposed* version of the target, from an equally *transposed* input (modifying Equation 2) as

$$\tilde{\mathbf{x}}_{t+1} = \sigma_g(\mathbf{V}^T (\mathbf{W}_0^T \mathbf{W}_1^T \mathbf{m}_t \cdot \mathbf{U} shift(\mathbf{x}_{t-n}^t, \delta))), \quad (5)$$

with $\delta \in [-30, 30]$ for the symbolic, and $\delta \in [-60, 60]$ for the audio data. Finally, we penalize the error between the reconstruction of the transposed target and the actual transposed target (i.e., employing Equation 3) as

$$\mathcal{L}(shift(\mathbf{x}_{t+1}, \delta), \tilde{\mathbf{x}}_{t+1}). \quad (6)$$

The transposition distance δ is randomly chosen for each training batch. This method amounts to both, a form of guided training and data augmentation. Some weights (i.e., filters) in \mathbf{U} and \mathbf{V} resulting from that training are depicted in Figure 2.

5.2 Architecture and Training Details

The architecture and training details of the GAE are as follows: A temporal context length of $n = 8$ is used (the choice of $n > 1$ leads to higher robustness of the mapping codes to diatonic transposition). The factor layer has 1024 units for the symbolic data, and 512 units for the spectrogram data. Furthermore, for all datasets, there are 128 neurons in the first mapping layer and 64 neurons in the second mapping layer (resulting in $\mathbf{m}_t \in \mathbb{R}^{64}$).

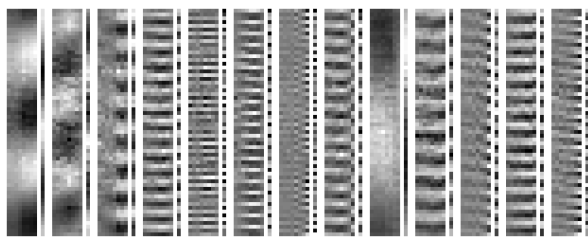


Figure 2: Some filter pairs $\in \{U, V\}$ of a GAE trained on polyphonic Mozart piano pieces.

L2 weight regularization for weights U and V is applied, as well as sparsity regularization [11] on the top-most mapping layer. The deviation of the norms of the columns of both weight matrices U and V from their average norm is penalized. Furthermore, we restrict these norms to a maximum value. We apply 50% dropout on the input and no dropout on the target, as proposed in [14]. The learning rate ($1e-3$) is gradually decremented to zero over the course of training.

6. EXPERIMENTS

In this Section we describe several experimental analyses to validate the proposed approach. They are intended to test the degree of transposition-invariance of the learned mappings, as well as assess their musical relevance (Sections 6.1 and 6.3). Finally, we put the learned representations to practice in a repeated section discovery task for symbolic music and audio (Section 6.2).

6.1 Classification and Cluster Analysis

Our hypothesis is that the model learns relative pitch representations (i.e. intervals) from polyphonic absolute pitch sequences. In order to test this hypothesis, we conduct two experiments using the symbolic data.

In the first experiment a ten-fold k-nn classification of intervals is performed (where $k = 10$), where the task is to identify all pitch intervals between notes in the input and the target of an input/target pair. If the learned mappings actually represent intervals, the classifier will perform substantially better on the mappings than on the input space. As intervals in music are transposition-invariant, the interval labels do not change when performing transposition in the input space. Thus, we perform the classification on the mappings of the original data and of randomly transposed data, to test if the mappings are indeed transposition-invariant.

We label the symbolic train data input/target pairs according to all intervals which occur between them, independent of the temporal distance of the notes exhibiting the intervals. Thus, each pair can have multiple labels. For each pair in the test set the k-nn classifier predicts the set of interval labels that are present in the k neighbors of that pair. The classification is performed in the input space (using concatenated pairs) and in the mapping space. Using these predictions we determine the precision, recall, and

Data	Precision	Recall	F1
Original input			
Mapping space	91.27	70.25	76.66
Input space	65.58	46.05	50.59
Transposed input			
Mapping space	90.78	71.44	77.31
Input space	51.81	32.99	37.43
All	26.40	100.0	40.05
None	0.0	0.0	0.0

Table 1: Results of the k-nn classification in the mapping space and in the input space for the original symbolic data and data randomly transposed by $[-24, 24]$ semitones. “All” is a lower bound (always predict all intervals), “None” returns the empty set.

F-score over the test set (cf. Table 1). For example, when a pair contains 6 intervals and the classifier estimate yield 4 true-positive and 4 false-positive interval occurrences, that pair is assigned a precision of 0.5 and a recall of 0.67.

In the second part of the experiment, the cluster centers of all intervals in the mapping space are determined. Again, each pair projected into the mapping space accounts for all intervals it exhibits and can therefore participate in more than one cluster. The mutual Euclidean distances between all cluster centers are displayed as a matrix (cf. Figure 3). An interpretation of the results follows in Section 7.

6.2 Discovery of Repeated Themes and Sections

The MIREX Task for Discovery of Repeated Themes and Sections for Symbolic Music and Audio² tests algorithms for their ability to identify repeated patterns in music. The commonly used JKUPDD dataset [3] contains 26 motifs, themes, and repeated sections annotated in 5 pieces by J. S. Bach, L. v. Beethoven, F. Chopin, O. Gibbons and W. A. Mozart. We use the MIDI and the audio versions of the dataset and preprocess them as described in Section 4.

We calculate the reciprocal of the Euclidean distances between all representations \mathbf{m}_t of a song, resulting in a transposition-invariant similarity matrix X . Then, the values of the main diagonal are set to the minimal value of the matrix. Subsequently, the matrix is normalized and convolved with an identity matrix of size 15×15 to emphasize and smooth diagonals (Figure 4 shows a resulting matrix). The method used to determine repeated parts based on diagonals of high values in the self-similarity matrix is adopted from [25], with a different method to identify diagonals, as described below.

The function

$$s(i, j, N) = \sum_{k=N-m}^N \frac{X(i+k, j+k)w_k}{m} \quad (7)$$

returns the score for a diagonal starting at $X(i, j)$ with

²http://www.music-ir.org/mirex/wiki/2017:Discovery_of_Repeated_Themes_&_Sections

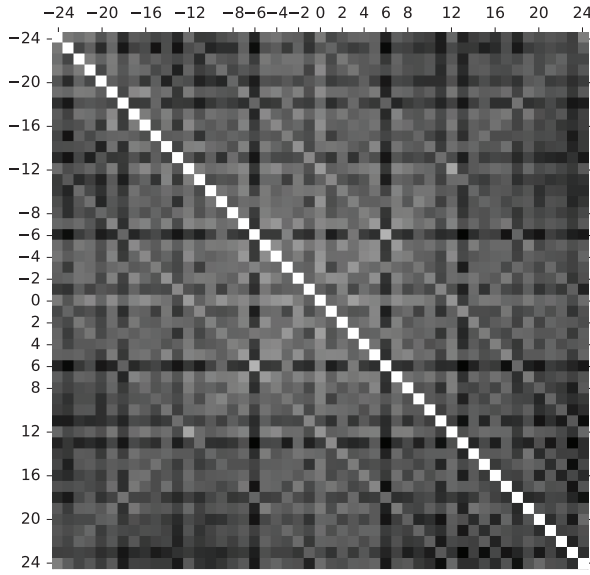


Figure 3: Distance matrix of cluster centers of intervals represented in mapping space. Darker cells indicate higher distances between respective clusters, brighter cells indicate closeness.

length N , and diagonals with high score are considered to be repeated sections. For each i, j , we iteratively evaluate the score with N increasing from 1 in integer steps, until the score undercuts a threshold γ . Only the last m values, $m = \min(10, N)$, of the diagonal are taken into account, because those values indicate when to stop tracing. The factor

$$w_k = \frac{1 + k + m - N}{m} \quad (8)$$

linearly weights the last m values of the diagonal so that later values have more impact on the overall score.

Three empirically determined parameters influence the functioning of the method: (1) from the diagonals found, we only keep those spanning more than 2 *whole notes*, (2) all sections whose common boundaries start and end within the length of a *half note* are considered to be repetitions of each other, (3) the thresholds γ determining if a diagonal should be considered a repetition in the symbolic and the audio data are set to 0.9 and 0.81, respectively. The results are shown in Table 2 and are discussed in Section 7.

6.3 Sensitivity Analysis

The sensitivity of the model to specific context information provides important insights into the functioning of the model. A common way of determining a networks sensitivity is by calculating the absolute value of the gradients of the networks predictions with respect to the input, holding the network parameters fixed [29]. Figure 5 shows the sensitivity of the model with respect to the temporal context. The model is particularly sensitive to note occurrences at $t \in \{0, -3, -7\}$. This shows that the most informative notes for a prediction are direct predecessors ($t = 0$), and notes which occur a quarter ($t = -3$) and a

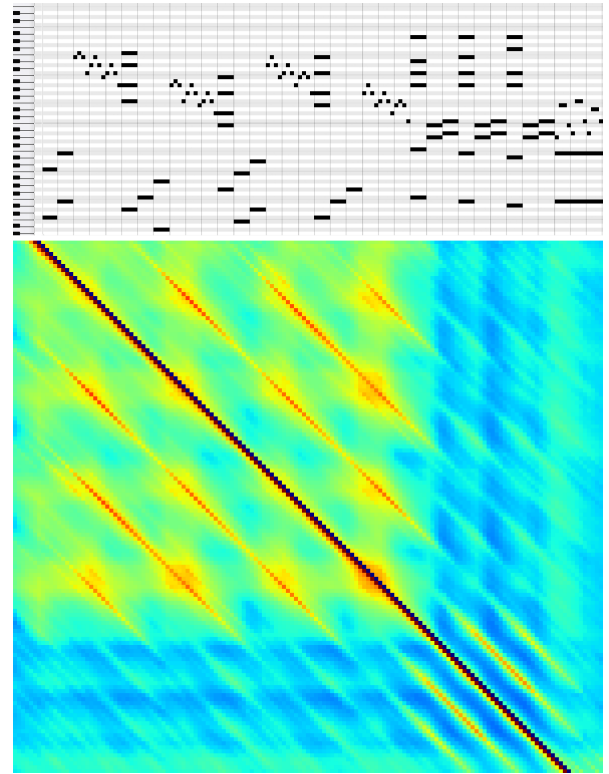


Figure 4: Symbolic music and corresponding self-similarity matrix calculated from transposition-invariant mapping codes. Warmer colors indicate similarity, colder colors indicate dissimilarity.

half note ($t = -7$, i.e., eight sixteenth notes) before the prediction.

7. RESULTS AND DISCUSSION

The results of the k-nn classification on the raw data and on representations learnt by the model are shown in Table 1. Classification in the mapping space appreciably outperforms classification in the input space, and obtains similar values for mappings of the original data and the randomly transposed data. In contrast, when performing classification in the input space the results deteriorate for the randomly transposed input and do not exceed the theoretical lower bound (i.e., always predict all intervals). As the register and keys of the original data are limited, correlations between absolute and relative pitch exist. When transposing the input, the classifier cannot make use of these absolute cues for relative pitch any more and performs weakly in the input space.

Figure 3 indicates which intervals are close to each other in the mapping space. An obvious regularity are the slightly brighter k-diagonals (i.e. parallels to the main diagonal) with $k \in \{-24, -12, 12, 24\}$, showing that two pitch intervals lead to similar mapping codes when they result in the same pitch class, such as the intervals +8 and -4 semitones, or -7 and -19 semitones. This is an indication that

Algorithm	F_{est}	P_{est}	R_{est}	$F_{o(.5)}$	$P_{o(.5)}$	$R_{o(.5)}$	$F_{o(.75)}$	$P_{o(.75)}$	$R_{o(.75)}$	F_3	P_3	R_3	Time (s)
Symbolic													
GAE intervals (ours)	59.07	77.60	58.30	68.92	80.24	67.46	77.51	91.38	73.29	50.44	60.36	53.23	127
VMO symbolic [34]	60.79	74.57	56.94	71.92	79.54	68.78	75.98	75.98	75.99	56.68	68.98	53.56	4333
SIARCT-CFP [4]	33.70	21.50	78.00	76.50	78.30	74.70	-	-	-	-	-	-	-
COSIATEC [19]	50.20	43.60	63.80	63.20	57.00	71.60	68.40	65.40	76.40	44.20	40.40	54.40	7297
Audio													
GAE intervals (ours)	57.67	67.46	59.52	58.85	61.89	56.54	68.44	72.62	64.86	51.61	59.60	55.13	194
VMO deadpan [34]	56.15	66.80	57.83	67.78	72.93	64.30	70.58	72.81	68.66	50.60	61.36	52.25	96
SIARCT-CFP [4]	23.94	14.90	60.90	56.87	62.90	51.90	-	-	-	-	-	-	-
Nieto [25]	49.80	54.96	51.73	38.73	34.98	45.17	31.79	37.58	27.61	32.01	35.12	35.28	454

Table 2: Different precision, recall and f-scores (adopted from [34], details on the measures are given in [3]) of different methods in the Discovery of Repeated Themes and Sections MIREX task, for symbolic music and audio. The F_3 score constitutes a summarization of all measures.



Figure 5: Absolute sensitivity of the model when looking backwards on the temporal context, averaged over the whole dataset.

the model learns the phenomenon of octave equivalence, even if the input to the model represents only absolute pitch. Another distinct feature is the stripe which is orthogonal to the main diagonal (i.e. where $y = -x$). This indicates that the model develops some notion of relative distances, by positioning intervals of the same distance (but different signs) close to each other.

Note also that the mappings of certain intervals, notably 6 and -6 , are distant to those of most other intervals (dark horizontal and vertical lines). This likely reflects the fact that tritone intervals are rare in diatonic music, and is further evidence of the musical significance of the learned mappings.

Table 2 shows results of the repeated themes and section discovery task, where the F_3 score is a good indicator for the overall performance of the models (see [3] for a thorough explanation on the respective measures). For the audio data, the current state-of-the-art F_3 score was raised from 50.60 to 51.61 by our proposed method. The method performs slightly worse on the symbolic data, which is counterintuitive at first sight, given that results of other models suggest that this task is easier. Our hypothesis is that for discovery of repeated sections, approx-

imate matching leads to better results than exact comparison, simply because musical variation goes beyond chromatic transposition (towards which our model is invariant). For approximate matching, a spectrogram representation is better suited than symbolic vectors, as notes are blurred over more than one frequency bin, and harmonics may provide additional cues for a similarity estimation. The proposed approach is computationally efficient, because the diagonal detector (cf. Equations 7 and 8) is rather simple and the transposition-invariance of the representations does not require explicit comparison of mutually transposed musical textures.

8. CONCLUSION AND FUTURE WORK

In this paper we have presented a computational approach to deriving (pitch) transposition-invariant vector space representations of music both in the symbolic and the audio domain. The representations encode pitch intervals that occur in the music in a musically meaningful way, with tritone intervals—a rare interval in diatonic music—leading to more distinct representations, and octaves leading to more similar representations. Furthermore, the temporal sensitivity of the model reveals a beat pattern that shows increased sensitivity to pitch intervals occurring at beat multiples of each other.

The transposition-invariance of the representations makes it possible to detect transposed repetitions of musical sections in the symbolic and in the spectral domain of audio. We have demonstrated that this is beneficial in tasks such as the MIREX task *Discovery of Repeated Themes and Sections*. A simple diagonal finding approach on a transposition-invariant self-similarity matrix produced by our model is sufficient to outperform the state of the art in the audio version of the task.

We believe it is worthwhile to further explore the utility of transposition-invariant music representations for other applications, including speech recognition, music summarization, music classification, transposition-invariant music alignment (including a cappella voices with pitch drift), query by humming, fast melody-based retrieval in large audio collections, and music generation. First results show that the proposed representations are useful for audio-to-score alignment [1] and for music prediction tasks [10].

9. ACKNOWLEDGMENTS

This research was supported by the EU FP7 (project Lrn2Cre8, FET grant number 610859), and the European Research Council (project CON ESPRESSIONE, ERC grant number 670035). We thank Oriol Nieto for providing us with the source code of his experiments [25].

10. REFERENCES

- [1] Andreas Arzt and Stefan Lattner. Audio-to-score alignment using transposition-invariant features. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*.
- [2] Emiliós Cambouropoulos. A general pitch interval representation: Theory and applications. *Journal of New Music Research*, 25(3):231–251, 1996.
- [3] Tom Collins. Discovery of repeated themes and sections. http://www.music-ir.org/mirex/wiki/2017:Discovery_of_Repeated_Themes_%26_Sections, 2017.
- [4] Tom Collins, Andreas Arzt, Sebastian Flossmann, and Gerhard Widmer. Siarct-cfp: Improving precision and the discovery of inexact musical patterns in point-set representations. In *ISMIR*, pages 549–554, 2013.
- [5] Afshin Dehghan, Enrique G Ortiz, Ruben Villegas, and Mubarak Shah. Who do i look like? determining parent-offspring resemblance via gated autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1757–1764, 2014.
- [6] Alain Droniou, Serena Ivaldi, and Olivier Sigaud. Learning a repertoire of actions with deep neural networks. In *IEEE International Joint Conferences on Development and Learning and Epigenetic Robotics (ICDL-Epirob)*, pages 229–234. IEEE, 2014.
- [7] Alain Droniou, Serena Ivaldi, and Olivier Sigaud. Deep unsupervised network for multimodal perception, representation and classification. *Robotics and Autonomous Systems*, 71:83–98, 2015.
- [8] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.
- [9] Stefan Lattner and Maarten Grachten. Learning transformations of musical material using gated autoencoders. In *Proceedings of the 2nd Conference on Computer Simulation of Musical Creativity, CSMC 2017, Milton Keynes, UK, September 11-13, 2017*, 2017.
- [10] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. A predictive model for music based on learned relative pitch representations. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*.
- [11] Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng. Sparse deep belief net model for visual area V2. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 873–880. Curran Associates, Inc., 2007.
- [12] Matija Marolt. A mid-level representation for melody-based retrieval in audio collections. *IEEE Transactions on Multimedia*, 10(8):1617–1625, 2008.
- [13] Josh McDermott and Andrew Oxenham. Music perception, pitch, and the auditory system. *Current Opinion in Neurobiology*, 18:1–12, 2008.
- [14] Roland Memisevic. Gradient-based learning of higher-order image features. In *IEEE International Conference on Computer Vision (ICCV), 2011*, pages 1591–1598. IEEE, 2011.
- [15] Roland Memisevic. On multi-view feature learning. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML ’12, pages 161–168, New York, NY, USA, July 2012. Omnipress.
- [16] Roland Memisevic and Georgios Exarchakis. Learning invariant features by harnessing the aperture problem. In *ICML (3)*, pages 100–108, 2013.
- [17] Roland Memisevic and Geoffrey Hinton. Unsupervised learning of image transformations. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR.*, pages 1–8. IEEE, 2007.
- [18] Roland Memisevic and Geoffrey E Hinton. Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Computation*, 22(6):1473–1492, 2010.
- [19] David Meredith. Cosiatec and siateccompress: Pattern discovery by geometric compression. In *International Society for Music Information Retrieval Conference*, 2013.
- [20] David Meredith, Kjell Lemström, and Geraint A Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4):321–345, 2002.
- [21] Vincent Michalski, Roland Memisevic, and Kishore Konda. "modeling deep temporal dependencies with recurrent grammar cells". In *Advances in neural information processing systems*, pages 1925–1933, 2014.

- [22] Decebal Constantin Mocanu, Haitham Bou Ammar, Dietwig Lowet, Kurt Driessens, Antonio Liotta, Gerhard Weiss, and Karl Tuyls. Factored four way conditional restricted Boltzmann machines for activity recognition. *Pattern Recognition Letters*, 66:100–108, 2015.
- [23] Meinard Müller and Michael Clausen. Transposition-invariant self-similarity matrices. In Simon Dixon, David Bainbridge, and Rainer Typke, editors, *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007, Vienna, Austria, September 23-27, 2007*, pages 47–50. Austrian Computer Society, 2007.
- [24] Eita Nakamura and Shinji Takaki. Characteristics of polyphonic music style and markov model of pitch-class intervals. In Tom Collins, David Meredith, and Anja Volk, editors, *Mathematics and Computation in Music - 5th International Conference, MCM 2015, London, UK, June 22-25, 2015, Proceedings*, volume 9110 of *Lecture Notes in Computer Science*, pages 109–114. Springer, 2015.
- [25] Oriol Nieto and Morwaread M Farbood. Identifying polyphonic patterns from audio recordings using music segmentation techniques. In *Proc. of the 15th International Society for Music Information Retrieval Conference*, pages 411–416, 2014.
- [26] Bruno A Olshausen, Charles Cadieu, Jack Culpepper, and David K Warland. Bilinear models of natural images. In *Electronic Imaging 2007*, pages 649206–649206. International Society for Optics and Photonics, 2007.
- [27] Laura Rimell, Amandla Mabona, Luana Bulat, and Douwe Kiela. Learning to negate adjectives with bilinear models. *EACL 2017*, page 71, 2017.
- [28] Jan Schlueter and Christian Osendorfer. Music similarity estimation with the mean-covariance restricted Boltzmann machine. In *10th International Conference on Machine Learning and Applications and Workshops (ICMLA), 2011*, volume 2, pages 118–123. IEEE, 2011.
- [29] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
- [30] Joshua B Tenenbaum and William T Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.
- [31] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.
- [32] C Von der Malsburg. The correlation theory of brain function reprinted in e. domani, jl van hemmen and k. schulten (eds.), *models of neural networks ii*, 1981.
- [33] Thomas C Walters, David A Ross, and Richard F Lyon. The intervalgram: an audio feature for large-scale melody recognition. In *Proc. of the 9th International Symposium on Computer Music Modeling and Retrieval (CMMR)*. Citeseer, 2012.
- [34] Cheng-i Wang, Jennifer Hsu, and Shlomo Dubnov. Music pattern discovery with variable markov oracle: A unified approach to symbolic and audio representations. In Meinard Müller and Frans Wiering, editors, *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, pages 176–182, 2015.
- [35] Gerhard Widmer. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence*, 146(2):129–148, 2003.

Session F

Machine and human
learning of music

INFLUENCES ON THE SOCIAL PRACTICES SURROUNDING COMMERCIAL MUSIC SERVICES: A MODEL FOR RICH INTERACTIONS

Louis Spinelli Josephine Lau Liz Pritchard Jin Ha Lee

Information School, University of Washington, Seattle

spinelli@uw.edu, jolau@uw.edu, epritch@uw.edu, jinhalee@uw.edu

ABSTRACT

Music can play an important role in social experiences and interactions. Technologies in-use affect these experiences and interactions and as they continue to evolve, social behaviors and norms surrounding them also evolve. In this paper, we explore the social aspects of commercial music services through focus group observation and interview data. We seek to better understand how existing services are used for social music practices and can be improved. We identified 9 social practices and 24 influences surrounding commercial music services. Based on the user data, we created a model of these practices and influences that provides a lens through which social experiences surrounding commercial music services can be understood. An understanding of these social practices within their contextual ecosystem help inform what influences should be considered when designing new technologies. Our findings include the identification of: the underlying relationships between practices and their influences; practices and influences that inform the weight of relationships in social networks; social norms to be considered when designing social features; influences that add additional insight to previously observed behaviors; and a detailed explanation of how music selection and listening practices can be supported by commercial music services.

1. INTRODUCTION

Music plays a role in social experience and social cohesion [3, 18]. It can function as an icebreaker, to facilitate informal interactions, to initiate friendships, and to strengthen relationships [18]. Different music media and technology such as tapes, CDs, and digital files offer different affordances that influence the activities surrounding music [4]. These music-related activities are also influenced by the physical and social context of the technology [5]. In this paper, we investigate music-related “social practices,” defined as activities a person carries out on a regular basis involving others or in the presence of other people. Social

practices surrounding music previously studied include listening, discovering [12], sharing [4, 12], exploring and peeping [20].

In 2006, O’Hara and Brown’s work [20] on social and collaborative aspects of music consumption technologies provided an up-to-date and comprehensive foundation for research into the social practices surrounding music at that time. Since then, Komulainen et al. [16] explored music sharing of youth in the context of new social media and mobile devices. Leong and Wright [18] explored social practices surrounding music in households. Cunningham et al. also explored social music practices in specific places and situations like parties [8] and cars [9]. Hagen and Luders [12] explored sharing and following behaviors with commercial music services. However, existing research on social practices surrounding music provides only a glimpse of the social behaviors surrounding music respective of that time and then-current technology.

Technological aspects related music practices have also been explored. For instance, Chamberlain and Crabtree [6] explored the workflows and technologies involved in the discovery, identification, acquisition, and organization of music in domestic settings. Goto [11] has also explored the influence of new “intelligent” interfaces on music practices. Several researchers including Barrington et al. [1], Zhu et al. [23], and Kamehkhosh et al. [15] also explored different factors, dimensionality of music, or automated recommendations that affect playlist generation/evaluation and music search. Although their research did not specifically focus on social practices surrounding commercial music services, our research adds new depth and context to their findings.

Streaming music technologies have grown in prevalence [19] and new features such as app integrations, and auto generated playlists as well as new interactive mechanisms like voice control have become more ubiquitous. The majority of music listeners in America are now streaming music [19]. Because of the role music plays in society and the influence of technology on social activities, it is important to gain an updated understanding of the influences on social practices surrounding commercial music services.

Building upon earlier research, our work looks specifically at commercial music services such as Spotify, Pandora, and Google Play, which have become pervasive in recent years. As social practices surrounding music have co-evolved with commercial music services, it is important



© Louis Spinelli, Josephine Lau, Liz Pritchard, Jin Ha Lee. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Louis Spinelli, Josephine Lau, Liz Pritchard, Jin Ha Lee. “Influences on the Social Practices Surrounding Commercial Music Services: A Model for Rich Interactions”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

to understand the implications these new technologies have on social practices, both positive and negative. However, updated literature on social practices surrounding commercial music services — as well as influencers on social practices — is lacking. To address this gap in knowledge we conducted six focus groups on two university campuses. We found that social practices surrounding commercial music services exist within a rich ecosystem of influences. We contribute a codebook defining 24 influences identified from user data as well as a model that provides a lens through which these experiences can be viewed and understood. This codebook provides insight into what influences need to be taken into consideration when designing technologies that are used socially. Notable findings based on our codebook and model include the identification of: the underlying relationships between practices and their influences; practices and influences that inform the weight of relationships in social networks; social norms to be considered when designing social features; influences that add additional insight to previously observed behaviors; and a detailed explanation of how music selection and listening practices can be supported by commercial music services.

2. RELATED WORK

Social practices were previously observed and described in context of the technologies of the day. When O'Hara and Brown published their book in 2006, MP3 sharing platforms like Gnutella, Kazaa, and Soulseek had just replaced Napster, and iTunes was a new legal addition to the market. In 2013, Leong and Wright observed changes in social behaviors around the exploration, discovery, and sharing of music in relation to changes in technologies including streaming internet and bluetooth on mobile phones. They noted “an emergence of new sociality and new forms of social practices around music” as well as new social tensions emerging around music selection and listening in shared settings [18].

Previous research in social practices surrounding music highlights the important role evolving technologies play in understanding the social practices surrounding music. A body of related work focuses on technology in shared environments. Brush and Inkpen [5] examine the use and sharing of technology in domestic environments. They found two common models for sharing devices: the appliance model and the profile model. Sharing of devices using the appliance model is mediated through social protocols. In contrast, sharing of devices using the profile model is mediated by allowing users to have individual profiles. Brush and Inkpen [5] also looked at the ownership models of devices - individual ownership versus shared ownership - within domestic environments. The physical locations of technology, privacy, and capability for personalization of technologies all influenced social behaviors [5]. They found that video game systems exemplified devices with shared ownership whereas mobile music players exemplified devices with individual ownership [5]. Jacobs, Cramer, and Barkhuus found four types of behaviors when studying the sharing practices of personal devices

between cohabiting couples: “intentional sharing, explicitly not sharing, unintentional access and unintentionally inhibiting access” [14]. They also observed that couples support sharing behaviors by “hacking” the intended use of the technologies [14].

These studies highlight that practices surrounding technology are influenced by the relationships of the individuals involved as well as the environment. Not only are social practices influenced by individual relationships, previous research also has shown social music practices also *influence* relationships and individual behavior. Boer and Abubakar [3] described the benefits of music listening in families and peer groups in which they found “benefits for young people’s social cohesion and emotional well-being”. Yang, Wang, and Mourali describe the influence of peers on unauthorized music downloading and sharing [22]. The physical context (environment) of previous research into social music practices also includes cars, public locations, workspaces, and dance clubs [20].

Research has also focused on individuals and their practices surrounding music and commercial music services. In 2013, Belcher and Haridakis [2] explored the motives people have for listening to music. Their results included the identification of social motivations influencing music listening and selection behaviors. Lee and Price [17] developed seven personas based on empirical music user data. These personas provide greater insight into design implications for users than user groups based on demographics. Our work further expands understanding of commercial music service users and their behaviors in social situations. Finally, while Hagen and Lüders [12] have looked at the sharing and following behaviors of commercial music service users, we take a broader look at the social practices surrounding commercial music services and their influences.

3. STUDY DESIGN AND METHODS

Six focus groups were held at Eckerd College and at the University of Washington, Seattle (UW). Participants were commercial music service users who were aged eighteen to thirty-four and lived with roommates. Results were analyzed by qualitative content analysis using a constant comparative method. Focus groups were selected to enable rich conversations where participants could prompt and remind one another of social situations they may have encountered. Participants were often prompted by situations similar to what they had encountered but described handling them in different ways. Participants often contrasted their social behaviors with individual behaviors.

3.1 Participants

Recruitment activities for the focus groups consisted of displaying flyers, posting to listservs, and posting on social media. Physical flyers were placed on boards around the Eckerd College Campus, the UW Campus, and in businesses surrounding each campus. Posts were also made to additional listservs and social media outlets known to

the researchers that included student affiliated groups and groups of people not affiliated with either university. Participants were compensated with an Amazon gift card for being part of two additional activities for subsequent research not discussed in this paper. All recruiting activities directed potential participants to a screener survey.

The screener survey was used to ensure all participants were between the ages of 18 and 34, currently living with a roommate or roommates, and using at least one commercial music service. Of the 80 potential participants who filled out the survey, 61 were eligible and available to attend the focus group on one of the preselected dates. Focus groups were filled on a first come first serve basis.

In total, 6 participants from the screener for Eckerd College and 20 participants from the screener for UW took part in six focus sessions - 2 held at Eckerd College and 4 held at UW. Of the 25 participants who reported a gender identity, 16 were female and 9 were male. 24 participants were between the ages 18 to 24, and 2 others were ages 25 to 34. Participants reported using a diverse array of commercial music services currently on the market including Spotify, Pandora, Google Play Music, YouTube, Soundcloud, Apple Music, Online radio services (e.g. NPR music, iHeartRadio, etc.), Indieshuffle, Tidal, and Amazon Music.

3.2 Procedures

Each of the six focus group sessions were approximately an hour long, with 3 to 6 participants per session. Each session had a facilitator and note-taker. Each session was also recorded and transcribed to ensure accurate analysis. Two pilot focus groups were held - one at each location - to test the focus group script, although refinement of the script continued throughout the study.

The script¹ was designed to prompt participants to have open-ended dialogues with each other about their social practices surrounding commercial music services. Generally, each focus group began with a warm-up activity where each participant introduced themselves and described the commercial music services they use, then the group brainstormed different locations where they listen to music identifying the social situations. Participants were then prompted to talk about their social practices in co-listening situations, when sharing, and with technology.

3.3 Analysis

After transcribing each focus group session, we anonymized participant information and analyzed the results in a two-part qualitative content analysis process.

For the first part, we separated each participant comment into individual post-it notes and conducted constant comparative analysis including affinity diagramming [21]. We opted for this method because it allows the creation of categories to be driven by the raw data and not established a priori [21]. When building an affinity diagram,

not only did social practices surrounding commercial music services emerge from the data, but influencers to these practices also emerged. Reflecting on previous work developing personas of commercial music service users [17], we recognized the importance of understanding these influences on individual behavior in social situations.

For the second part, a codebook with social practices and influences was developed following an iterative coding process using Dedoose, custom Python code, and Google Sheets. An initial version of the codebook was produced by a single team member who coded all transcripts asking two questions about each excerpt: 1) Does this describe a social practice surrounding a commercial music service? and 2) Does this describe something that influences a social practice surrounding a commercial music service?

The codebook was then revised and refined through an iterative team coding process, following a consensus model [13]. During each test, each excerpt was coded with applicable codes for social practices and influences. Each excerpt describing a social situation could be coded with multiple social practices and influences. Codes applied to excerpts by different independent coders were then compared. When applied codes differed, the team members who coded the excerpts discussed their reasoning leading to an agreement that one coder erred, that an update to the codebook was needed, or that a third team member was needed as a tie-breaker.

4. RESULTS

4.1 Social Music Practice Codebook and Model

During analysis, 9 distinct social practices and 24 influences emerged¹ (Table 1). These influences were grouped into three categories: group/social influences, external influences, and internal influences.

Many of the practices and influences that emerged could be applied to – and have been observed with – other technologies. For example, Music Technology Management could have been described as Technology Management – something that has been studied in households with all technologies [5].

Co-occurrence of social practices and influences when applying codes to the transcript led to the insight that each social practice surrounding commercial music services happens in an ecosystem. The ecosystem's complexity is captured in our model of the influences on the social practices surrounding commercial music services, as explained in more detail in the following sub-sections.

4.1.1 A Flexible Model of a Rich Ecosystem

Our model of the social practices surrounding commercial music services and the influences thereof represents a rich community of practices and influences. Influences affect other influences. Social practices affect other social practices as well as their own influences. This meant that a simple situation would likely capture multiple social practices and multiple influences, as illustrated by the sample quote below.

¹ Documented here: <https://perma.cc/Z58H-XQJU>

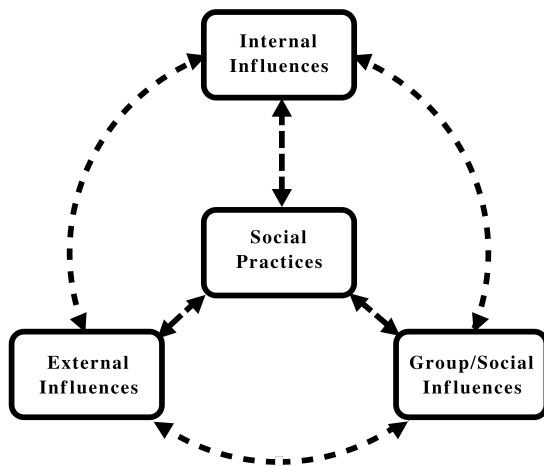


Figure 1. A model of the influences on the social practices surrounding commercial music service.

“...my roommate also likes Phish so I was at a show and they played her favorite song, so I like sent it to her... Like I sent a video recording of it and she was so excited. And we bonded over that.” (P22)

We coded this sample quote of a participant describing sharing a video of a song with her roommate with two social practices (*Sharing Music (Information)* and *Social Interaction/Navigation*) as well as three influences - two external influences (*Technology/Music Collection and Event/Activity*) and one group/social influence (*Level of Group Intimacy*). *Level of Group Intimacy* and *Social Interaction/Navigation* demonstrate the bidirectional nature of the influences and social practices. The practice here - sharing music - is influenced by an existing relationship which it also strengthened.

4.1.2 A Non-Linear Model

Our model does not define the sequence in which social practices occur. Similar to Fosters’ nonlinear model of information seeking behavior, our model of social practices is nonlinear [10]. Practices precipitate other practices, co-occur, and can be repeated. While a practice like *Music Identification* can occur after listening to music, it can also happen during listening or before listening.

“Yeah, for me, they need to be vetted as friend first, and then I’ll see if I’ll take their music recommendations.” (P2)

As described by participant 2, for a social practice like sharing (*Sharing Music (Information)*) to be successful it may need to be preceded by social interactions (*Social Interaction/Navigation*) that build trust and develop intimacy. These preceding practices could also include sharing (*Music (Information) Sharing*).

4.2 The Full Spectrum of Each Influence

Each code refers to the full spectrum of that influence, so users could describe an influence as being very important to them or not at all. For instance, the *Privacy and Security Considerations* code also captures the lack of privacy

concerns when a participant described sharing their phone with friends:

“You can never be too sure, but I just don’t care. If they see anything, they see it. I guess I trust them not to snoop around on the phone.” (P15)

5. DISCUSSION

To illustrate the utility of the codebook and model, we discuss findings that emerged from its application, adding to the overall understanding of music services in social contexts.

5.1 The Unequal Weight of Influences

Understanding the meaning and underlying weight of influences supports quantitative data and adds depth to design considerations. Quantitative network analysis is bolstered by an understanding of both the strength of connections and what is driving each relationship. Understanding which influences will hinder the adoption of a new feature or technology can save time and money, and identify negative externalities that may not have been considered.

5.1.1 Weight of Relationships in Social Networks

Social network analysis is used to study the “connection[s] and interaction[s] between social actors” [7]. Crossley et al.’s 2014 collection of essays explore different applications of social network analysis for understanding “music worlds” - a phrase they use to describe collective actions that are similar to social movements [7]. Rather than looking at large scale social movements, we focused on what practices surrounded commercial music services and why. Understanding why users share music and what sharing indicates about their relationships provides insight into the strengths of connections between social actors.

“Sometimes I will share [playlists] ... but usually just [with] close friends...” (P18)

“She knows music and knows when it sounds good and when it doesn’t. So, if she’s like ‘listen to this because it sounds good’, then I take every opinion that she has.” (P23)

Our findings indicate that a person is more likely to share a playlist with close friends. In addition, participants reported that after receiving music, they were more likely to listen to songs shared by vetted friends. With this finding in mind, quantitative data showing social actors that share playlists and listen to one another’s recommendations could indicate users that have a high *Level of Intimacy*.

5.1.2 Social Norms as Design Considerations

We found social norms exist around vehicles and residences, and have strong influences on social practices. Our participants indicate the location and the type of relationship influence music sharing in social situations. Participants were unlikely to share a request with a stranger, a host, or a driver in a social situation because they did not believe it was socially appropriate. On the other hand, the

A Social Practices Surrounding Music Any activity involving two or more participants engaging with music, or situations where an individual changes his/her music-related behavior based on social influences. Otherwise, does not apply to individual settings.

A.1 Music Discovery When one finds new music with someone, from someone else, or from a social interaction/environment.

A.2 Music Identification When one takes action to determine the title of a song, artist, or album in a social interaction/environment.

A.3 Music Listening When a group (two or more participants) listens to music together.

A.4 Music Management When one actively engages with music and music metadata (e.g., adding an album, adding a song to a playlist, browsing another's collection, using a storing practice to remember a song for later).

A.5 Music Selection When one selects music for co-listening, sharing, and for collaborative playlists.

A.6 Music Technology Management When one regulates how one's music technologies and accounts are protected, shared, accessed, or determined for their uses in a social interaction/environment.

A.7 Navigating Space/Setting When one adjusts their behavior depending on the setting (e.g., leaving a room when roommate is playing bad music or passing the phone around in a car).

A.8 Music (Information) Sharing When one shares or receives music/ music metadata.

A.9 Social Interaction/Navigation Bonding activities, group dynamic formation activities, group norming activities.

B Internal Influences Any activity where a social music practice is influenced by how an individual engages with the practice to a varying degree.

B.1 Assertiveness Level The degree an individual influences or confronts others, or vice versa (e.g., taking over music selection for the group or not feeling

comfortable suggesting a song be changed when one does not like it).

B.2 Considerateness The degree to which an individual cares about bothering others.

B.3 Current State The current emotional state, mood, or preference of an individual. NOT the current event or activity, for this USE: External Influence >Event/Activity.

B.4 Effort/Engagement Level of effort or engagement an individual is willing to put forth or take on responsibility.

B.5 Expertise/Knowledge of Music Having an expertise/knowledge about music, such as an awareness of new music, or lack thereof.

B.6 Impression Management Considerations When others' perceived reception of music choices, suggestions, or tastes affect an individual's actions in a social music practice.

B.7 Openness Willingness to explore new music or others' recommendations.

B.8 Privacy and Security Considerations Considerations relating to privacy and/or security that influence an individual's actions in a social music practice.

B.9 Social Driver Social need or purpose for interaction (e.g., a bonding experience, a desire to share the same space with another person).

B.10 Technology Knowledge/Considerations Knowledge and consideration, or lack thereof, of features of technology, preferences for technology, personal attachment to technology.

B.11 Tolerance The degree an individual endures music or music-related behavior/situation they do not like.

B.12 Trust/Reliability When group members have varying degree of confidence in another member's ability to undertake social music practices (e.g., music selection/sharing) or when a member has varying degree of confidence from other group members.

B.13 Willingness The degree of willingness an individual exhibits to take part in a practice.

C External Influence Any activity where a social music practice is influenced by something outside of individual, group, or social traits.

C.1 Event/Activity Attributes of an event or activity including the goals and the situational context.

C.2 Norm/Expectation Societal norms including for places, events, and gatherings.

C.3 Ownership and/or Control of Service or Technology Possession of technology (e.g., speakers, Chromecast) or access (e.g., subscription) to a commercial music service.

C.4 Popularity/Reception of Music The wider societal and cultural reputation of a song, artist, or genre as well as the prevalence of this knowledge.

C.5 Technology/Music Collection Technology, or the attributes/features thereof, being used (e.g., commercial music service or physical collections likes vinyl or CDs).

C.6 Temporal/Spatial Physical space, physical proximity, or temporality.

D Group/Social Influence Any activity where a social music practice is influenced by the social aspect of a situation or setting to a varying degree.

D.1 First Mover When someone else being in a setting first affects the social situation.

D.2 Group Dynamic When a group's shared preferences or norms affect how they generally engage with music (e.g., a group's preferences for songs, genres, technologies, or a group member to play music).

D.3 Group Size When the number of people in the social situation affects how the group engages with music.

D.4 Level of Group Intimacy When the level of familiarity between group members affects how they engage with music (e.g., perceived knowledge of another's taste or opinion in music).

Table 1. Codebook of Social Practices and Influences.

more intimate the relationship, the less these social norms stood in the way.

"If someone's playing music, it's usually the driver's call." (P1)

"I've never even thought about asking an Uber driver to play music, like I don't even know like, well I guess like because a driver's a stranger I'd feel kind of weird." (P12)

"It depends on the people you are with, if you're with friends, it's fine, if you're with siblings it's possible you can compete, but if you're with people who you don't know much, you would rather listen to what's going on rather than insist on playing something or maybe just plug in own earphones and not notice it." (P9)

Participants discussed situations where they would or would not ask music to be changed or request a song to be played. The ownership of the space seemed to matter significantly as they talked about respecting the host of the party or the driver being in control of the music. Participants indicated that they would be uncomfortable requesting a song or a song change if they were not the driver unless with a group of friends. Most participants reported that they would be unlikely to do either of these behaviors in a rideshare vehicle.

5.2 Insight into Invisible Influences

Leong and Wright observed recent technologies supporting social practices, but also contributing to social tensions [18]. They observed nuanced situations that involved

control (*Assertiveness*), rituals (*Group Dynamic*), cultural/linguistic elements (*Social Norms*, *Group Dynamic*), relationship (*Level of Group Intimacy*), *Considerateness*, and setting (*Temporal/Spatial*, *Event/Activity*). While independently developing our codebook we captured similarly nuanced situations. In addition, we identified an additional influence not explained in the previous research, "*Social Driver*", to describe situations where participants simply wanted to be collocated with others.

"Sometimes it's nice with my roommate, we'll go to our rooms... and just listen to our own individual music and do our own thing which can be nice, but you know when we want to be social... then it's kind of nice to listen to music together..." (P17)

"[Headphones allow] me to immerse myself, like, in myself while still being in public." (P23)

This need to be social has implications for social music practices and what technologies should consider. Social needs may drive people with disparate music tastes to use a commercial music service together to select music or for people to wear headphones in a shared space. *Event/Activity*, *Temporal/Spatial*, and *Level of Group Intimacy* played a role in what participants did in these situations. For instance, when studying, participants would often use headphones, but when taking a break from studying, participants would more likely select music with their roommates. This finding — that an interaction between *Social Driver* and *Event/Activity* affects *Music Selection* —

has implications for playlist development and how commercial music services can support individual and social listening (described in Section 5.3).

5.3 Supporting Selection and Listening Practices through Playlist Generation

Music Selection practices varied depending on a number of influences including *Group Size*, *Group Dynamic*, *Event/Activity*, and *Effort/Engagement*. The broadness of a music collection and playlist content (*Technology/Music Collection*) played a role in what technology was used and what music was selected during different experiences.

“Depending on the activity, most of the time when I’m alone, I’ll just put on a playlist that’s already curated. Like running, or at the gym, that way I don’t have to skip anything. It’s usually pretty tailored to that specific activity. Or like studying, I usually don’t really listen stuff with a lot of lyrics, because that can be distracting. But when I’m with friends, there’s more skipping, we’ll have a pretty broad variety of songs.” (P16)

For many social situations, participants described either developing or selecting playlists that included a larger variety of music enabling more skipping of songs. This behavior can be supported by anticipating this skipping behavior with playlists - both pre-made and auto-generated - by including a larger selection of songs knowing that some will be skipped.

“You might choose to play some common songs which generally people would like and not very exotic choices so that most of them enjoy - not all - but maybe most of them would like to have it there.” (P9)

“If I really want to listen to something, I’ll usually just listen to it by myself, because I can just focus on the music or focus on the task that I’m doing rather than having to socialize with other people, which is kind of, where you get lost in the conversation and miss out on the music.” (P15)

For social situations, participants described choosing playlists and songs “commonly” popular amongst their social groups. Many participants described selecting music to set the mood, but recognized the music would not be the sole focus of the social situation. They would choose to listen to music on their own if they really wanted to focus on it. The described playlists for social situations included go-to songs for the group, current hits, classics, and other music already likely to be known by group members.

“If we’re pre-gaming, we can use a playlist and just use that and that way no one has to touch the phone. Everyone can be talking while the music is just in the background.” (P16)

“It is annoying to like constantly be adding songs and also to listen to three people’s songs that you may not like their music as much.” (P7)

“I do the queue thing a lot, like taking requests, if not just queueing up stuff yourself.” (P6)

Effort and engagement also played a role in music selection in social situations. While most participants preferred playlists in social situations, some participants described behaviors that required more effort – selecting and queue-

ing songs was one of these behaviors.

6. CONCLUSION AND FUTURE WORK

In this work, we created a model of these practices and influences that provides a lens through which social experiences surrounding commercial music services can be understood as technology continues to evolve and affect them. Our model, building on previous work, provides insight into the social practices and influences that should be taken into consideration when designing commercial music services. Applying our codebook on qualitative data pertaining to specific technologies and user groups enables researchers to gather design considerations for social practices specific to their own technology and context.

Design implications for music services based on the user data include:

1. **Invitations to break social norms:** If a music service wanted to support music sharing in social situations, a push notification sent from a host or driver inviting the guest or passenger to make a request through a selected music service could help overcome inhibitive social norms.
2. **Customization of playlists and stations for social influences:** Playlists and stations are currently organized by genre, activity, and mood. Designing to support social listening would involve allowing participants to select, customize, or generate playlists and stations based on group size. Recognition of group consumption should increase the amount of songs and their level of popularity/broad appeal.

The addition of individual interviews in future studies would likely lead to further insight into privacy considerations and impression management behaviors, although participants seemed to speak freely about both. Additional studies with different methodologies and different segments of the population will allow the model and codebook to be revised, enriched, updated, and validated. Also, further research is currently underway exploring the Q-method as a way to understand the personal significance of different influences for individual participants.

While this model was based on the social practices surrounding commercial music services, it can apply to other technologies. A good example might be the streaming video services that were often described analogously by participants. It is likely that many of the influences will be similar, although the technologies differ.

7. ACKNOWLEDGEMENTS

The authors would like to thank Katie O’Leary, Briana Keller, Pandora, Spotify, and Google Play Music.

8. REFERENCES

- [1] L. Barrington, R. Oda, and G.R.G. Lanckriet. Smarter than Genius? Human Evaluation of Music Recommender Systems. In *Proc. ISMIR*, pages 357–362, 2009.

- [2] J.D. Belcher and P. Haridakis. The Role of Background Characteristics, Music-Listening Motives, and Music Selection on Music Discussion. *Communication Quarterly*, 61(4):375–396, 2013.
- [3] D. Boer and A. Abubakar. Music listening in families and peer groups: Benefits for young people’s social cohesion and emotional well-being across four cultures. *Frontiers in Psychology*, 5, 2014.
- [4] B. Brown and A. Sellen. Sharing and Listening to Music. In *Consuming Music Together*, pages 37–56. Springer, Dordrecht, 2006.
- [5] A.J.B. Brush and K.M. Inkpen. Yours, Mine and Ours? Sharing and Use of Technology in Domestic Environments. In *UbiComp 2007: Ubiquitous Computing*, pages 109–126, 2007.
- [6] A. Chamberlain and A. Crabtree. Searching for music: Understanding the discovery, acquisition, processing and organization of music in a domestic setting for design. *Personal and Ubiquitous Computing*, 20(4):559–571, 2016.
- [7] N. Crossley, S. McAndrew, and P. Widdop. *Social Networks and Music Worlds*. Routledge, New York, 2014.
- [8] S.J. Cunningham and D.M. Nichols. Exploring social music behaviour: An investigation of music selection at parties. In *Proc. ISMIR*, pages 747–752, 2009.
- [9] S.J. Cunningham, D.M. Nichols, D. Bainbridge, and H. Ali. Social music in cars. In *Proc. ISMIR*, pages 457–462, 2014.
- [10] A. Foster. A Nonlinear Model of Information-Seeking Behavior. *Journal of the American Society for Information Science and Technology*, 55(3):228–237, 2004.
- [11] M. Goto. Intelligent Music Interfaces. In *Proc. IUI*, pages 3–4, 2018.
- [12] A.N. Hagen and M. Lüders. Social streaming? Navigating music as personal and social. *Convergence: The International Journal of Research into New Media Technologies*, 23(6):643–659, 2017.
- [13] C.E. Hill, S. Knox, B.J. Thompson, E.N. Williams, S.A. Hess, and N. Ladany. Consensual qualitative research: An update. *Journal of Counseling Psychology*, 52(2):196–205, 2005.
- [14] M. Jacobs, H. Cramer, and L. Barkhuus. Caring About Sharing: Couples’ Practices in Single User Device Access. In *Proc. of the 19th International Conference on Supporting Group Work*, pages 235–243, 2016.
- [15] I. Kamehkhosh, D. Jannach, and G. Bonnin. How Automated Recommendations Affect the Playlist Creation Behavior of Users. In *MILC*, 2018.
- [16] S. Komulainen, M. Karukka, and J. Häkkinen. Social Music Services in Teenage Life: A Case Study. In *Proc. OZCHI*, pages 364–367, 2010.
- [17] J.H. Lee and R. Price. Understanding Users of Commercial Music Services Through Personas: Design Implications. In *Proc. ISMIR*, pages 476–482, 2015.
- [18] T.W. Leong and P.C. Wright. Revisiting Social Practices Surrounding Music. In *Proc. CHI*, pages 951–960, 2013.
- [19] Everyone Listens to Music, But How We Listen is Changing. *Nielson Newswire*, 2015, <https://perma.cc/C5DM-26P5>.
- [20] K. O’Hara and B. Brown. *Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies*. Springer, Dordrecht, 2006.
- [21] A.J. Pickard. *Research Methods in Information*. Neal-Schuman, Chicago, 2nd ed. edition, 2013.
- [22] Z. Yang, J. Wang, and M. Mourali. Effect of peer influence on unauthorized music downloading and sharing: The moderating role of self-construal. *Journal of Business Research*, 68(3):516–525, 2015.
- [23] S. Zhu, J. Cai, J. Zhang, Z. Li, J.C. Wang, and Y. Wang. Bridging the User Intention Gap: An Intelligent and Interactive Multidimensional Music Search Engine. In *Proc. WISMM*, pages 59–64, 2014.

INVESTIGATING CROSS-COUNTRY RELATIONSHIP BETWEEN USERS' SOCIAL TIES AND MUSIC MAINSTREAMINESS

Christine Bauer

Johannes Kepler University Linz
christine.bauer@jku.at

Markus Schedl

Johannes Kepler University Linz
markus.schedl@jku.at

ABSTRACT

We investigate the complex relationship between the factors (i) preference for music mainstream, (ii) social ties in an online music platform, and (iii) demographics. We define (i) on a global and a country level, (ii) by several network centrality measures such as Jaccard index among users' connections, closeness centrality, and betweenness centrality, and (iii) by country and age information. Using the LFM-1b dataset of listening events of Last.fm users, we are able to uncover country-dependent differences in consumption of mainstream music as well as in user behavior with respect to social ties and users' centrality. We could identify that users inclined to mainstream music tend to have stronger connections than the group of less mainstream users. Furthermore, our analysis revealed that users typically have less connections within a country than cross-country ones, with the first being stronger social ties, though. Results will help building better user models of listeners and in turn improve personalized music retrieval and recommendation algorithms.

1. INTRODUCTION

When meeting new people, they frequently tend to talk about their favorite music as conversation starter [30]. Indeed, several studies (e.g., [3, 23, 33, 43]) indicate that shared music preferences create and intensify social bonds. For instance, Boer et al. found in a study that participants liked others with the same music preferences more than those with different music preferences [3]. Based on this result, the authors conclude that shared music preferences can generate and increase social attraction.

In online social networks (OSN), such as Facebook, Instagram, or Twitter, the social bonding effects of shared music preferences are expected to follow similar patterns as the ones observed in offline settings, i.e., in the physical world. In the context of OSN, it is particularly interesting to consider that connections between users are not constrained to any single country, which is frequently the case in offline scenarios [5]; indeed, many social ties between

users are cross-country connections [1]. Yet, sometimes individuals center their interactions within locally bounded social circles also in their online interaction behavior [10]. Whether they do so or rather not, however, strongly depends on the users' cultural backgrounds. For instance, Choi et al. found that American users maintained larger but looser networks, whereas Korean users had smaller but denser networks [9]. Barnett and Benefield analyzed cross-country friendship connections on Facebook and found that international ties tended to share borders, language, civilization, and migration aspects [1].

Similarly, it has been found that music preferences are highly influenced by the cultural background of listeners [40]. In particular, they strongly depend on the country the user lives in, and each country has its own characteristics with respect to which music is considered popular or mainstream in that very country [38].

In contrast to the above general studies on cross-country user connections and music preferences, little is known about how shared music preferences and social ties are related in OSN and how the social bonding effect varies for cross-country ties. Against this background, the research questions (RQ) we address are:

- RQ1: In which ways do listeners in different countries differ in terms of their inclination to listen to mainstream music (considering both global and country-specific mainstream)?
- RQ2: In which ways do listeners in different countries differ in terms of their social ties and connectedness in a music-related online social network (Last.fm)?
- RQ3: In which ways do the previous two aspects interrelate, i.e., does maintaining strong social ties (within or between countries) interrelate with a preference for mainstream music?

The answers to these questions will help building better models of listeners—individually and on a country level—and in turn improve personalized music retrieval and recommendation algorithms, as it has already been shown for other user characteristics, such as demographics [47], activity [49], or mood [26]. For instance, the intensity of cross-country ties of a user u together with information about the music mainstream of u 's country and the countries u 's friends originate from may be used to tailor recommendations for u . To give an example, if a Spanish user u maintains very strong ties to users in Brazil, a music recommender system may include in its recommendation list



© Christine Bauer, Markus Schedl. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Christine Bauer, Markus Schedl. "Investigating Cross-Country Relationship between Users' Social Ties and Music Mainstreaminess", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

a few music items that are popular only in Brazil, to ideally provoke serendipitous music encounters for u .

The remainder of this article is organized as follows. Section 2 presents related work on music mainstream, social connectedness, and culture-aware listener analysis and modeling. Section 3 details the methodology we apply to answer the research questions. Section 4 presents and discusses the obtained results. Eventually, Section 5 rounds off this work with a conclusion and pointers to future research.

2. RELATED WORK

The work at hand connects to research on music preferences and mainstream, on user connectedness in social networks, and on culture-aware music and listener analysis. We briefly discuss the most important related literature in these areas and connect our work to it.

2.1 Music Mainstream

A user's music preferences are shaped by various factors. Extant studies have investigated the relationship between music preferences and, amongst others, demographics (e.g., [4]), personality traits (e.g., [7]), or social influences (e.g., [3, 46]). Music tastes and preferences are measured in various ways, for instance, in terms of genre (e.g., [3, 29, 32, 40]), artist (e.g., [36, 48]), or mood (e.g., [14, 18]) preference.

Another approach to distinguish music preferences is to consider the degree of people's tendency to favor music that is considered *mainstream*, i.e., music that is most popular within the entire population [41]. In short, measuring music preferences in terms of a user's degree of mainstreamness is a popularity-based approach that considers the degree to which a user prefers music items that are currently popular or rather ignores such trends [34]. Further studies revealed that people's preferences vary across countries, which holds true for both music genres [40] as well as mainstream music [38]. Early research with respect to music mainstreamness for the use in music recommendation systems shows that the population which a user is compared to tremendously impacts the outcome with respect to recommendation performance [2, 34]. More specifically, a user may be compared to the mainstream from a global perspective, but also from a country perspective. Yet, an in-depth analysis of country-specific differences concerning mainstreamness—from a global perspective and a country perspective—is a research gap.

2.2 Social Connectedness

Research on the strength of social connections dates back to Granovetter's paper entitled "The Strength of Weak Ties" [15], describing the social network theory, which he later revisited in [16]. In OSN research, social connectedness has been a target of research since the early days of OSN. For instance, although theoretically not constrained to any single region [5, 9], social connections on

OSN sometimes tend to center within locally bounded social circles [10, 51], because social ties in OSN may follow the spatial, structural, and cultural perimeters of the societal system that OSN users belong to in offline settings, i.e., in the physical world [5].

Initially, designing measures of tie strength had been difficult as Granovetter [15, 16] had not given a precise conceptual definition for it [24]. A scale of measures has developed since then. Among the most common measures for tie strength and derived measures for node importance are the overlap in users' neighborhoods via Jaccard index (J), the closeness centrality (C), and the betweenness centrality (B), which we therefore also use in our work, and detail in Section 3.2.

Studies have revealed that music preferences play an important role in creating and intensifying social bonds [3, 23, 33, 43], because shared music preferences can generate and increase social attraction [3]. In other words, people tend to like people with the same music preferences more than people with different music preferences [3].

This fact has been exploited, among others, in [25], where a social approach for music recommendation is presented. It is based on the assumption that friendship relations in OSN are similar to those offline and that Facebook relationships are indicative of similar music tastes. The proposed system recommends YouTube music tracks to a target user, which have been positively rated (with at least 3 on a 5-point Likert scale) by the target users Facebook friends, but have not been rated by the target user him or herself.

While previous research on music and social bonding most often measures music preferences in terms of genre (e.g., [3, 23, 43]), we argue that music mainstreamness may be an additional, insightful indicator for music preferences with regard to social bonding.

2.3 Culture-aware Music and Listener Analysis

Generally, human preferences have shown to be rooted and embodied in culture [20], and also listeners' music preferences are affected by cultural aspects (e.g., [11]). For instance, perception of music varies across cultures [22, 44, 45], which obviously influences music preferences. Furthermore, national market structures, including local airplay and subsidizing (e.g., local music quotas on radio) are different across countries [28, 31] and shape country-specific popularity of artists and songs. This results, among others, in the fact that pop music preferences disconverge rather than converge within European countries [8].

With the increasing popularity of personalized music recommender systems—i.e., systems that tailor recommendations for particular music items (e.g., artists, albums, or songs) to the preferences of individuals [42]—and the acknowledgement that tailoring recommendations to a listener's cultural specificities may substantially increase the performance of a music recommender system [2, 38, 47], research investigating and describing music and listener profiles from a culture perspective has received attention

lately. To provide some examples, [27] show that incorporating cultural characteristics allows for more precise characterization of listeners; [50] integrate cultural aspects for modeling music similarity; [21] use culture-aware approaches describing and modeling intonation of audio music recordings. Comparisons of listener profiles across countries have been presented from many different angles [11, 37, 39], most frequently in terms of genres, while our work concentrates on mainstreamness.

3. METHODOLOGY

For our study, we use and extend the LFM-1b dataset [35], which comprises 1,088,161,692 listening events of 120,322 unique Last.fm users. Since our investigation aims at uncovering country-specific factors, we consider only the subset of the LFM-1b dataset that includes listening events of users who provide country information. To reduce the likelihood of less significant results due to a sample bias of users within a given country, we furthermore filter countries with less than 100 users, which results in a dataset of 53,258 users from 47 countries. Some of the users do not maintain any social ties on Last.fm. Excluding those (because we cannot compute the respective measures), we finally end up with a stable dataset of 5,680 users from 18 countries, on which we conduct our analysis.

3.1 Music Mainstreamness

To quantify the proximity of a user to both the country-specific and the global mainstream, we employ the approach proposed in [2, 38]. Schedl and Bauer identified two rank-based measures as being best suited to estimate mainstreamness of a user among his or her fellow citizens within the same country (Equation 1) and compared to a global mainstream (Equation 2). In the equations, which have been simplified from [2], where a complex framework is proposed, $M(u, c)$ denotes the rank-based mainstreamness of user u in regard to country c (which is in our case always the country of the user); $M(u)$ denotes u 's global mainstreamness. Furthermore, τ denotes the rank-order correlation coefficient according to Kendall [19]; AF denotes a vector containing the global artist frequencies of all artists in the dataset, keeping a fixed order (i.e., the first element in vector AF is the total number of listening events to the artist who is most frequently listened to globally, and so on); $AF(c)$ is defined analogously, but only considers listening events in country c , maintaining the ordering of artists given by the global AF vector; $AF(u)$ analogously, but only considering listening events of user u (again maintaining the global ordering); $ranks(\cdot)$ represents the ranks of the real-valued artist frequencies given in vector (\cdot) .

Less formally, $M(u, c)$ measures how well user u 's ranking of artist preferences corresponds to that of all users in country c ; $M(u)$ measures how well u 's ranking of artist preferences matches with the global ranking. Higher values indicate closer to the mainstream.

$$M(u, c) = \tau(ranks(AF(c)), ranks(AF(u))) \quad (1)$$

$$M(u) = \tau(ranks(AF), ranks(AF(u))) \quad (2)$$

3.2 Social Ties and Centrality Measures

To uncover social ties between users in the LFM-1b dataset, we first enrich the dataset using the Last.fm API endpoint `user.getFriends`¹ to obtain the connections of all users in LFM-1b. Since we are only interested in the intra-connectedness between users in the dataset, we exclude all friendship connections to users that are not contained in the LFM-1b dataset. This results in a total of 79,254 connections by 11,801 users (5,680 users only considering the 18 countries with at least 100 users). On the resulting network, we then compute tie strength and centrality scores that estimate the importance of nodes (users) in a network. More precisely, we use Jaccard index (J), closeness centrality (C), and betweenness centrality (B) since they are among the most common measures. Jaccard index (J) is defined as the fraction of shared neighbors among all neighbors of the two users u and v under consideration [17]. To obtain a single measure per user u , we compute the arithmetic mean of the Jaccard indices between u and all users connected to u . Closeness centrality (C) of user u is defined as the reciprocal of the sum of the shortest path distances between u and all other users in the network [13]. Higher values of closeness therefore indicate higher centrality. Betweenness centrality (B) of user u is defined as the sum of the fraction of all shortest paths between pairs of nodes v, w ($\neq u$) that pass through u [12]. Betweenness can therefore be regarded as how much in the way between two arbitrary users u lies. Users with high betweenness are assumed to have more control in the network, because more information will pass through them.

4. RESULTS AND DISCUSSION

4.1 Country vs. Mainstreamness

To answer the first research question, i.e., how listeners in different countries vary in terms of their inclination to listen to mainstream music, Table 2 shows basic statistics (mean and standard deviation) of country-specific and global mainstreamness, for the top countries in the dataset (those with at least 100 users). The grand means and SD are 0.091 ± 0.060 for M_{country} and 0.103 ± 0.062 for M_{global} . Additionally, mean, standard deviation, and median age of users are depicted. The countries with highest local mainstreamness are the Netherlands, the United Kingdom, and Canada ($M_{\text{country}} = M(u, c) > 0.1$); those with highest global mainstreamness are Finland, the Netherlands, and Mexico ($M_{\text{global}} = M(u) > 0.11$). This is in line with previous work [36], which used a different definition of mainstreamness, nevertheless identified the Netherlands, the United Kingdom, Belgium, and Canada as most mainstream countries.² The high rank of Finland in our results may be surprising since many citizens of this country are known to have a preference for metal music, cf. [38], which is rather not considered mainstream. At the same time, however, also the standard deviation of

¹ <https://www.last.fm/api/show/user.getFriends>

² Note that Belgium is not included in our analysis because only 63 Belgian users remained after filtering.

Table 1. Top 20 global artists and their deviations of Finnish preference from the global preference in terms of artist frequency.

Artist	Global rank	Deviation
The Beatles	1	-47.44 %
Radiohead	2	-43.95 %
Pink Floyd	3	-25.80 %
Metallica	4	+126.72 %
Muse	5	+131.66 %
Arctic Monkeys	6	-55.71 %
Daft Punk	7	+96.84 %
Coldplay	8	-16.63 %
Linkin Park	9	-11.17 %
Red Hot Chili Peppers	10	-0.10 %
System of a Down	11	+152.54 %
Nirvana	12	-30.23 %
Iron Maiden	13	+170.77 %
Rammstein	14	+171.76 %
Depeche Mode	15	-22.87 %
Lana Del Rey	16	-28.33 %
Lady Gaga	17	+132.72 %
Led Zeppelin	18	-34.54 %
Florence + the Machine	19	-29.49 %
David Bowie	20	-19.43 %

mainstreamness is very high for Finland, which indicates a strong dispersion over mainstream and non-mainstream music preferences among Fins. In fact, a deeper analysis reveals a large variety of music tastes in Finland, cf. Table 1. On the one hand, metal bands such as Metallica, System of a Down, and Iron Maiden are indeed more popular among Fins than globally. On the other hand, also artists such as Muse (top tags on Last.fm: alternative, rock), Daft Punk (electronic, house), and Lady Gaga (pop, dance) are highly popular in Finland.

According to our dataset, the least mainstreamy countries are Germany, Australia, and the Czech Republic, regardless of whether mainstreamness is computed on the country level or globally.

Another observation is that the Scandinavian countries Norway and Sweden both show low standard deviations in their citizens' mainstreamness level, indicating a stable inclination for a certain level of mainstream among the listeners in these countries. Interestingly, for Norway this goes together with a rather low mainstreamness level (low tertile), while Sweden's level ranges in the high tertile.

We further investigate the correlation between all aspects in Table 2. Computing Pearson correlation coefficients between all pairs of aspects and a 2-tailed t-test to investigate significance, we identify the following significant correlations at $p \leq 0.05$: $\rho(\text{M_country: mean, M_global: mean}) = 0.819$ ($p \approx 0.0$), $\rho(\text{M_global: mean, Age: mean}) = 0.280$ ($p = 0.05$).

4.2 Country vs. Social Ties and Centrality

Towards answering the second research question, i.e., how listeners in different countries vary in terms of their social ties and their connectedness within the Last.fm social network, Table 3 shows means and standard deviations of social tie strength (Jaccard index), closeness, and betweenness (cf. Section 3.2), again for the top 18 countries in the dataset. The grand means and SD for tie strength (J), closeness, and betweenness are 0.285 ± 0.101 , 0.150 ± 0.067 , and 0.027 ± 0.067 , respectively. The countries with highest average tie strength are Sweden ($J = 0.319$) and Finland ($J = 0.301$), closely followed by Poland ($J = 0.299$) and the Netherlands ($J = 0.297$). These J values indicate that, on average, users in these countries share nearly one third of their neighbors with all users they are connected to. The lowest tie strength values are present for Ukraine and the Czech Republic ($J \approx 0.26$), closely followed by Italy, Spain, Russia, and Australia ($J \approx 0.27$).

With respect to closeness centrality, the countries with highest C value are Ukraine, Italy, Spain, Russia, and Mexico ($C > 0.16$), those with lowest closeness are Sweden ($C = 0.117$), Poland, Finland, and the Netherlands ($C \approx 0.13$). Interestingly, in the case of Sweden, the lowest mean closeness centrality is paired with the highest standard deviation ($C = 0.117 \pm 0.084$). Investigating the reason for this, we find that there are many Swedish outliers with very low closeness centralities. Quantitatively, the 25-, 50-, and 75-percentiles for closeness in Sweden are 0.0002, 0.1500, and 0.1790, respectively, while being 0.1248, 0.1672, and 0.1910, on average, among all other countries.

As for betweenness, the countries with highest values ($B > 0.0004$) are Mexico and Italy, while lowest scores ($B < 0.0002$) are realized by users in the Netherlands, Sweden, and France. Mexico and Italy, however, also show the largest standard deviations. In fact, the median of their B values approaches zero. About half of Italian and Mexican users therefore have no or very few connections. Still, these countries' 75-percentile as well as maximum B is at the same time the highest among all countries, $B \approx 0.0003$ and $B \approx 0.01$, respectively. A few users in Italy and Mexico are hence extremely well connected and can be assumed to have a high level of influence in the entire analyzed network, i.e., sub-network of Last.fm [6].

Investigating which of the aspects in Table 3 correlate, Pearson correlation coefficients are significant at $p \leq 0.05$ for the following pairs of aspects: $\rho(\text{B: mean, J: mean}) = -0.363$ ($p = 0.01$) and $\rho(\text{C: mean, J: mean}) = -0.637$ ($p \approx 0.0$). The negative correlations between tie strength and centrality measures indicate that while direct neighbors between connected users show significant overlaps, this does not generalize to the whole network. Our assumption, which we test in the next section, is that these local neighbors who are well connected are rather users in the same country.

Table 2. Statistics of country-specific and global mainstreamness as well as age for countries with at least 100 users. Country names are abbreviated according to ISO 3166-1 alpha-2.

Country	Users	M_country		M_global		Age		
		mean	std	mean	std	mean	std	median
US	927	0.091	0.062	0.096	0.067	20.8	13.6	22.0
RU	789	0.093	0.057	0.102	0.061	18.9	12.0	21.0
PL	775	0.095	0.066	0.104	0.070	19.2	10.3	20.0
BR	531	0.091	0.065	0.107	0.069	19.7	10.0	21.0
UK	470	0.102	0.057	0.107	0.057	21.2	13.8	23.0
DE	463	0.081	0.062	0.088	0.066	20.7	13.3	22.0
FI	217	0.092	0.094	0.112	0.065	20.2	10.3	22.0
UA	207	0.097	0.052	0.108	0.052	19.3	11.5	22.0
IT	175	0.090	0.058	0.106	0.067	23.1	14.0	23.0
ES	157	0.088	0.053	0.104	0.059	21.9	12.1	24.0
NL	155	0.106	0.058	0.112	0.059	25.6	16.0	23.0
SE	132	0.094	0.049	0.105	0.053	21.6	13.9	22.0
CA	127	0.101	0.059	0.108	0.061	19.3	11.3	22.0
CZ	124	0.075	0.057	0.093	0.063	19.2	10.4	22.0
MX	109	0.087	0.060	0.110	0.062	21.7	11.4	23.0
FR	108	0.088	0.055	0.101	0.058	22.3	11.8	25.0
AU	107	0.085	0.061	0.092	0.070	20.0	11.4	21.0
NO	107	0.090	0.048	0.100	0.058	20.6	13.9	22.0

Table 3. Statistics of social tie strength and centrality measures for countries with at least 100 users. Country names are abbreviated according to ISO 3166-1 alpha-2.

Country	Users	Social Ties (J)		Closeness		Betweenness (x100)	
		mean	std	mean	std	mean	std
US	927	0.287	0.102	0.152	0.066	0.023	0.061
RU	789	0.270	0.103	0.162	0.064	0.031	0.073
PL	775	0.299	0.106	0.132	0.072	0.023	0.061
BR	531	0.287	0.102	0.159	0.060	0.028	0.075
UK	470	0.290	0.098	0.149	0.067	0.028	0.069
DE	463	0.286	0.106	0.145	0.072	0.024	0.056
FI	217	0.301	0.112	0.133	0.080	0.022	0.057
UA	207	0.261	0.098	0.165	0.054	0.027	0.055
IT	175	0.268	0.086	0.163	0.059	0.040	0.125
ES	157	0.269	0.092	0.163	0.055	0.032	0.067
NL	155	0.297	0.113	0.135	0.080	0.017	0.053
SE	132	0.319	0.104	0.117	0.084	0.019	0.044
CA	127	0.294	0.105	0.156	0.067	0.023	0.058
CZ	124	0.265	0.098	0.152	0.063	0.027	0.065
MX	109	0.295	0.100	0.161	0.064	0.042	0.122
FR	108	0.282	0.100	0.154	0.062	0.019	0.039
AU	107	0.270	0.096	0.157	0.060	0.025	0.060
NO	107	0.291	0.103	0.142	0.068	0.026	0.067

4.3 Mainstreamness vs. Social Ties and Centrality

Regarding RQ3, i.e., in which ways do mainstream and social connectedness interrelate, we analyzed various aspects with respect to the 33,974 connections between the users in our sample. Most connections in our sample are cross-country (26,914 connections, i.e. 79%), while only 21% (or 7,060) are between users of the same country.

In a detailed analysis for differences between different degrees of mainstreamness vs. social ties and centrality, we found two significant differences: As conjectured, the social tie strength of users within the same country (measured by the Jaccard index between the connections of the two users to compare, cf. Section 3.2) differs from the social tie strength of cross-country connections. In a 2-tailed t-test, the difference between connections within a country

($mean = 0.241$, $std = 0.109$) and cross-country connections ($mean = 0.219$, $std = 0.095$) is highly significant ($t = 17.154$; $df = 33972$, $p = 0.000$).

Comparing each user's social tie strength (averaged over all his or her connections with his or her respective mainstreamness level), in a t-test, we found that the difference between the group of users with a low preference for mainstream ($mean = 0.281$, $std = 0.102$) and the group of high mainstream users ($mean = 0.289$, $std = 0.104$) is highly significant ($t = -2.819$, $df = 3777.883$, $p = 0.005$), when using the M_global measure. When using the M_country measurement, this effect disappears. We conjecture that from a country perspective of mainstreamness, the different forms of mainstream per country and the more focused music preference within a country levels the effect that can be seen from a global perspective.

Investigating individual countries, Table 4 shows that for all countries, the social tie strength between users within the country is higher than for connections spanning two countries. The difference is highly significant ($p \leq 0.001$) for BR, CA, DE, FI, NO, PL, SE, UA, UK, and US; the difference is significant ($p \leq 0.05$) for ES, NL, and RU. So, although the number of cross-country connections is higher than the number of connections within a country, the social tie strength for inner-country connections is higher for all countries under investigation.

5. CONCLUSION

Using the LFM-1b dataset of country-specific listener and listening information, we set out to answer three research questions: In which ways do listeners in different countries differ in terms of their inclination to listen to mainstream, on a global and a country level (RQ1)? In which ways do listeners in different countries differ in terms of their social ties and connectedness in Last.fm (RQ2)? In which ways do mainstream and social connectedness interrelate (RQ3)?

We found large differences between countries in terms of the level of global and regional mainstream consumption of listeners as well as their fluctuations, i.e., standard deviations (RQ1). A particularly interesting example is Finland with a mid (regional) to high (global) mainstreamness level. While seeming surprising at first glance, a high standard deviation in mainstreamness reveals that there is a group of Finnish listeners that largely follows the trend, whereas another large group established their own preferences, far away from the mainstream. Further analysis showed that this group's influence foremost stems from metal music. In contrast, Finland's neighbors Sweden and Norway show a very stable level of preference for mainstream.

In terms of social ties and centrality measures (RQ2), we found that, on average, Last.fm users share between one fourth (Italy, Spain, Russia, and Australia) and one third (Sweden and Finland) of their neighbors. Moreover, social tie strength is negatively correlated with betweenness and closeness centrality, which indicates that direct neighbors between connected users show significant overlaps,

Table 4. Differences in social tie strength between connections within a country and cross-country connections. Country names are abbreviated according to ISO 3166-1 alpha-2. Significance levels are: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.

country		connections	mean social ties (J)	std	t	df	p	
AU	within country	34	0.25620	0.11058	1.784	35.268	0.083	
	cross-country	760	0.22181	0.09615				
BR	within country	1075	0.25639	0.11569	7.736	3622.000	0.000	***
	cross-country	2549	0.22605	0.10438				
CA	within country	28	0.29538	0.12547	3.259	874.000	0.001	***
	cross-country	848	0.23501	0.09537				
CZ	within country	110	0.22807	0.09750	0.051	184.758	0.959	
	cross-country	315	0.22753	0.09416				
DE	within country	369	0.25107	0.11538	7.542	2704.000	0.000	***
	cross-country	2337	0.21144	0.08993				
ES	within country	180	0.23885	0.09972	2.730	248.262	0.007	*
	cross-country	880	0.21680	0.09397				
FI	within country	171	0.26051	0.12002	4.761	1252.000	0.000	***
	cross-country	1083	0.22110	0.09719				
FR	within country	42	0.25933	0.12916	1.114	44.620	0.271	
	cross-country	673	0.23666	0.10755				
IT	within country	246	0.24656	0.08706	1.856	1261.000	0.064	
	cross-country	1017	0.23359	0.10085				
MX	within country	108	0.22272	0.11188	0.002	128.309	0.998	
	cross-country	908	0.22270	0.10033				
NL	within country	67	0.26510	0.12334	2.113	75.556	0.038	*
	cross-country	717	0.23217	0.10690				
NO	within country	84	0.26555	0.10218	4.769	105.179	0.000	***
	cross-country	578	0.20911	0.09553				
PL	within country	958	0.25610	0.11539	12.336	3270.000	0.000	***
	cross-country	2314	0.20937	0.09075				
RU	within country	1596	0.21208	0.09940	2.201	5299.000	0.028	*
	cross-country	3705	0.20598	0.08945				
SE	within country	50	0.32686	0.11978	5.880	57.000	0.000	***
	cross-country	474	0.22339	0.10359				
UA	within country	160	0.22599	0.11370	3.547	1228.000	0.000	***
	cross-country	1070	0.19947	0.08376				
UK	within country	513	0.25545	0.11070	7.558	2869.000	0.000	***
	cross-country	2358	0.22043	0.09135				
US	within country	1269	0.23737	0.10070	4.474	5595.000	0.000	***
	cross-country	4328	0.22367	0.09456				

but this does not generalize to the whole network.

Our hypothesis that users whose neighborhoods are well connected are likely from the same country could be verified (RQ3). For most analyzed countries, our analysis revealed significantly higher social tie strength for connections within the same country compared to cross-country connections. In other words, although users have less connections within the same country than cross-country ones, the social ties are stronger for inner-country connections. Furthermore, our analysis identified that the group of mainstream users have stronger social ties compared to the group of users less inclined to mainstream music concerning tie strength.

The logical next step in this line of research is to integrate the findings into a music recommendation system. The mainstreamness and country information is highly useful to alleviate cold-start; the information about cross-

country social ties can be exploited to personalize recommendations depending on the tie strength between the target user and connections to users in other countries. For instance, collaborative filtering techniques could be extended by a mainstreamness or social tie filtering component, in a fashion similar to [38].

Finally, it would be worth investigating whether results generalize to platforms other than Last.fm. However, this research question may be hard to investigate externally and independently in the absence of publicly available datasets from the big players.

6. ACKNOWLEDGMENTS

This workshop is supported by the Austrian Science Fund (FWF): V579.

7. REFERENCES

- [1] George A Barnett and Grace A Benefield. Predicting international facebook ties through cultural homophily and other factors. *New Media & Society*, 19(2):217–239, 2017.
- [2] Christine Bauer and Markus Schedl. On the importance of considering country-specific aspects on the online-market: An example of music recommendation considering country-specific mainstream. In *51st Hawaii International Conference on System Sciences (HICSS 2018)*, pages 3647–3656.
- [3] Diana Boer, Ronald Fischer, Micha Strack, Michael H Bond, Eva Lo, and Jason Lam. How shared preferences in music create bonds between people: Values as the missing link. *Personality and Social Psychology Bulletin*, 37(9):1159–1171, 2011.
- [4] Arielle Bonneville-Roussy, P. Jason Rentfrow, Man K. Xu, and Jeff Potter. Music through the ages: Trends in musical engagement and preferences from adolescence through middle adulthood. *Journal of Personality and Social Psychology*, 105(4):703–717, 2013.
- [5] Danah Boyd. Why youth (heart) social network sites: The role of networked publics in teenage social life. *MacArthur foundation series on digital learning—Youth, identity, and digital media volume*, pages 119–142, 2007.
- [6] Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001.
- [7] Richard A. Brown. Music preferences and personality among japanese university students. *International Journal of Psychology*, 47(4):259–268, 2012.
- [8] Oliver Budzinski and Julia Pannicke. Do preferences for pop music converge across countries?—empirical evidence from the eurovision song contest. *Creative Industries Journal*, 10(2):168–187, 2017.
- [9] Sejung Marina Choi, Yoojung Kim, Yongjun Sung, and Dongyoung Sohn. Bridging or bonding? a cross-cultural study of social relationships in social networking sites. *Information, Communication & Society*, 14(1):107–129, 2011.
- [10] Nicole B Ellison, Charles Steinfield, and Cliff Lampe. The benefits of facebook “friends:” social capital and college students’ use of online social network sites. *Journal of computer-mediated communication*, 12(4):1143–1168, 2007.
- [11] Bruce Ferwerda, Andreu Vall, Marko Tkalčič, and Markus Schedl. Exploring music diversity needs across countries. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, pages 287–288. ACM, 2016.
- [12] Linton C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41, March 1977.
- [13] Linton C. Freeman. Centrality in Social Networks Conceptual Clarification. *Social Networks*, 1(3):215–239, 1978-1979.
- [14] Ronald S. Friedman, Elana Gordis, and Jens Förster. Re-exploring the influence of sad mood on music preference. *Media Psychology*, 15(3):249–266, 2012.
- [15] Mark S Granovetter. The strength of weak ties. In *Social networks*, pages 347–367. Elsevier, 1977.
- [16] Mark S Granovetter. The strength of weak ties: A network theory revisited. *Sociological theory*, pages 201–233, 1983.
- [17] Mangesh Gupte and Tina Eliassi-Rad. Measuring Tie Strength in Implicit Social Networks. In *Proceedings of the 4th Annual ACM Web Science Conference*, WebSci ’12, pages 109–118, New York, NY, USA, 2012. ACM.
- [18] Xiao Hu and Jin Ha Lee. A Cross-cultural Study of Music Mood Perception Between American and Chinese Listeners. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, October 2012.
- [19] Maurice G. Kendall. A New Measure of Rank Correlation. *Biometrika*, 30(1-2):81–93, 1938.
- [20] Shinobu Kitayama and Hyekyung Park. Cultural shaping of self, emotion, and well-being: How does it work? *Social and Personality Psychology Compass*, 1(1):202–222, 2007.
- [21] Gopala Krishna Koduri. Culture-aware approaches to modeling and description of intonation using multi-modal data. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 209–217. Springer, 2014.
- [22] Jin Ha Lee and Xiao Hu. Cross-cultural similarities and differences in music mood perception. *iConference 2014 Proceedings*, 2014.
- [23] Adam J Lonsdale and Adrian C North. Musical taste and ingroup favouritism. *Group Processes & Inter-group Relations*, 12(3):319–327, 2009.
- [24] Peter V Marsden and Karen E Campbell. Measuring tie strength. *Social forces*, 63(2):482–501, 1984.
- [25] Cedric S. Mesnage, Asma Rafiq, Simon Dixon, and Romain Brixet. Music Discovery with Social Networks. In *Proceedings of the 2nd Workshop on Music Recommendation and Discovery (WOMRAD)*, Chicago, IL, USA, October 2011.

- [26] Adrian C. North and David J. Hargreaves. Situational influences on reported musical preference. *Psychomusicology: Music, Mind and Brain*, 15(1-2):30–45, 1996.
- [27] Martin Pichl, Eva Zangerle, Günther Specht, and Markus Schedl. Mining culture-specific music listening behavior from social media data. In *2017 IEEE International Symposium on Multimedia (ISM)*, pages 208–215. IEEE, 2017.
- [28] Dominic Power And and Daniel Hallencreutz. Competitiveness, local production systems and global commodity chains in the music industry: entering the us market. *Regional Studies*, 41(3):377–389, 2007.
- [29] Peter J Rentfrow and Samuel D Gosling. The do re mi’s of everyday life: The structure and personality correlates of music preferences. *Journal of personality and social psychology*, 84(6):1236, 2003.
- [30] Peter J Rentfrow and Samuel D Gosling. Message in a ballad: The role of music preferences in interpersonal perception. *Psychological science*, 17(3):236–242, 2006.
- [31] Paul Rutten. Local popular music on the national and international markets. *Cultural Studies*, 5(3):294–305, 1991.
- [32] Thomas Schäfer. The goals and effects of music listening and their relationship to the strength of music preference. *PloS one*, 11(3):e0151634, 2016.
- [33] Thomas Schäfer, Peter Sedlmeier, Christine Städtler, and David Huron. The psychological functions of music listening. *Frontiers in psychology*, 4:511, 2013.
- [34] Markus Schedl. Ameliorating music recommendation: Integrating music content, music context, and user context for improved music retrieval and recommendation. In *Proceedings of International Conference on Advances in Mobile Computing & Multimedia, MoMM ’13*, pages 3:3–3:9, New York, NY, USA, 2013. ACM.
- [35] Markus Schedl. The LFM-1b Dataset for Music Retrieval and Recommendation. In *ACM International Conference on Multimedia Retrieval (ICMR)*, pages 103–110. ACM, 2016.
- [36] Markus Schedl. Investigating country-specific music preferences and music recommendation algorithms with the LFM-1b dataset. *International Journal of Multimedia Information Retrieval*, 6(1):71–84, 2017.
- [37] Markus Schedl. Investigating country-specific music preferences and music recommendation algorithms with the lfm-1b dataset. *International journal of multimedia information retrieval*, 6(1):71–84, 2017.
- [38] Markus Schedl and Christine Bauer. Introducing Global and Regional Mainstreaminess for Improving Personalized Music Recommendation. In *Proceedings of the 15th International Conference on Advances in Mobile Computing & Multimedia (MoMM 2017)*, Salzburg, Austria, December 2017.
- [39] Markus Schedl and Christine Bauer. Online music listening culture of kids and adolescents: Listening analysis and music recommendation tailored to the young. In *11th ACM Conference on Recommender Systems (RecSys 2017): International Workshop on Children and Recommender Systems (KidRec 2017)*, New York, NY, 2017. ACM.
- [40] Markus Schedl and Bruce Ferwerda. Large-Scale Analysis of Group-Specific Music Genre Taste from Collaborative Tags. In *Proceedings of the 19th IEEE International Symposium on Multimedia (ISM 2017)*, Taichung, Taiwan, December 2017.
- [41] Markus Schedl and David Hauger. Tailoring Music Recommendations to Users by Considering Diversity, Mainstreaminess, and Novelty. In *Proc. of SIGIR*, pages 947–950, Santiago, Chile, 2015.
- [42] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskis. Music recommender systems. In *Recommender Systems Handbook*, pages 453–492. Springer, 2015.
- [43] Maarten HW Selfhout, Susan JT Branje, Tom FM ter Bogt, and Wim HJ Meeus. The role of music preferences in early adolescents friendship formation and stability. *Journal of Adolescence*, 32(1):95–107, 2009.
- [44] Abhishek Singhi and Daniel G Brown. On cultural, textual and experiential aspects of music mood. In *ISMIR*, pages 3–8, 2014.
- [45] Catherine J Stevens. Music perception and cognition: A review of recent cross-cultural research. *Topics in cognitive science*, 4(4):653–667, 2012.
- [46] Tom F.M. ter Bogt, Marc J.M.H. Delsing, Maarten van Zalk, Peter G. Christenson, and Wim H.J. Meeus. Intergenerational continuity of taste: parental and adolescent music preferences. *Social Forces*, 90(1):297–319, 2011.
- [47] Gabriel Vigliensoni and Ichiro Fujinaga. Automatic Music Recommendation Systems: Do Demographic, Profiling, and Contextual Features Improve Their Performance? In *17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 94–100, 2016.
- [48] Jef Vlegels and John Lievens. Music classification, genres, and taste patterns: A ground-up network analysis on the clustering of artist preferences. *Poetics*, 60:76–89, 2017.
- [49] Xinxi Wang, David Rosenblum, and Ye Wang. Context-aware Mobile Music Recommendation for Daily Activities. In *Proceedings of the 20th ACM International Conference on Multimedia*, pages 99–108, Nara, Japan, 2012. ACM.

- [50] Daniel Wolff and Tillman Weyde. Learning music similarity from relative user ratings. *Information retrieval*, 17(2):109–136, 2014.
- [51] Shanyang Zhao and David Elesh. Copresence as 'being with': Social contact in online public domains. *Information, Communication & Society*, 11(4):565–583, 2008.

LISTENER ANONYMIZER: CAMOUFLAGING PLAY LOGS TO PRESERVE USER'S DEMOGRAPHIC ANONYMITY

Kosetsu Tsukuda

Satoru Fukayama

Masataka Goto

National Institute of Advanced Industrial Science and Technology (AIST), Japan

{k.tsukuda, s.fukayama, m.goto}@aist.go.jp

ABSTRACT

When a user signs up with an online music service, she is often requested to register her demographic attributes such as age, gender, and nationality. Even if she does not input such information, it has been reported that user attributes can be predicted with high accuracy by using her play log. How can users enjoy music when using an online music service while preserving their demographic anonymity? To solve this problem, we propose a system called *Listener Anonymizer*. *Listener Anonymizer* monitors the user's play log. When it detects that her confidential attributes can be predicted, it selects songs that can decrease the prediction accuracy and recommends them to her. The user can camouflage her play logs by playing these songs to preserve her demographic anonymity. Since such songs do not always match her music taste, selecting as few songs as possible that can effectively anonymize her attributes is required. *Listener Anonymizer* realizes this by selecting songs based on feature ablation analysis. Our experimental results using Last.fm play logs showed that *Listener Anonymizer* was able to preserve anonymity with fewer songs than a method that randomly selected songs.

1. INTRODUCTION

When a user signs up with an online music service (e.g., Last.fm¹ and Spotify²), it is common for the user to be asked to input her demographic attributes such as age and gender. Registering such demographic attributes is beneficial for her because various songs are recommended to her by the service according to her attributes. In addition, she can follow another user who has similar demographic attributes, and they can communicate with each other. Despite such benefits, many users conceal their demographic attributes because they would be concerned about privacy. As shown in Section 5.1, as many as 49.3% of Last.fm users do not register any of the age, gender, and nationality attributes. If a user does not register her demographic attributes, is her privacy fully protected?

Several studies have aimed to predict users' demographic attributes from their music play logs [10, 12, 25].

¹ <http://www.last.fm>

² <http://www.spotify.com>



© Kosetsu Tsukuda, Satoru Fukayama, Masataka Goto. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Kosetsu Tsukuda, Satoru Fukayama, Masataka Goto. "Listener Anonymizer: Camouflaging Play Logs to Preserve User's Demographic Anonymity", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

They have tackled the problem because it had been reported that the attributes contribute to improving music recommendation accuracy [21, 23, 27]. However, users who do not register their attributes might not want researchers or companies to predict their demographic attributes. It is not only a psychological problem; if a user's demographic attributes are predicted, she may suffer damage. For example, suppose one day the email address and music play logs of a female user who has not input her gender on an online music service are leaked from the website, and a malicious company obtains the data. If the malicious company can predict her gender with high accuracy from the logs, it can send her spam e-mails that target females.

What should we do to enable users to enjoy music when using an online music service while preserving their demographic anonymity? In this paper, we propose a system called *Listener Anonymizer* to solve this problem. *Listener Anonymizer* camouflages the user's play log and preserves the anonymity of her confidential attributes. To be more specific, when a user plays a song using an online music service, *Listener Anonymizer* monitors the songs that are played. If *Listener Anonymizer* detects that the user's confidential attributes can be predicted with an accuracy above a certain level, the system selects songs that can decrease the prediction accuracy and recommends them to her. The user can camouflage her play log by playing them and preserve her anonymity. However, since such selected songs do not always match her music taste, selecting as few songs as possible that can effectively anonymize her attributes is required. To achieve this, we propose a method for selecting songs according to the user's confidential attributes.

Our contributions in this paper are as follows.

- To the best of our knowledge, this is the first study that introduces the concept of preserving the anonymity of the users' demographic attributes while they play songs using an online music service.
- We propose an approach that camouflages the user's play log to preserve her anonymity. We marshaled factors to consider for selecting songs from five viewpoints: the definition of anonymity, method for predicting demographics, timing of camouflaging play logs, user's true demographics, and anonymization of multiple demographics.
- To examine the effectiveness of the proposed method, we carried out experiments using Last.fm play logs. Our experimental results showed that our proposed method was able to preserve anonymity with fewer songs than a method that randomly selected songs. Based on the experiments, we dis-

cuss four important considerations: impact on recommendation accuracy, user's taste in music, simulation of multiple prediction methods, and real-time monitoring of songs that are played.

2. RELATED WORK

Since predicting the user's demographic attributes can be used in many applications such as content recommendation and user behavior analysis, studies of demographic prediction have been conducted in various domains. One of the most popular domains is social media such as Twitter³ and Facebook⁴. It is known that language use on social media varies according to demographic attributes such as age [9] and gender [7]. Hence, most studies have used text data posted to social media and utilized machine learning techniques to predict users' demographic attributes [15, 18, 24]. Although it was thought that predicting demographics was a difficult task [15, 16], recent studies reported a high prediction accuracy. For example, age and gender can be predicted with mean absolute error (MAE) of 3.40 and a binary classification accuracy of 91.9%, respectively [18]. In addition to social media, demographic prediction has been conducted in the fields of blogs [2, 5] and web search queries [8].

In the field of music information retrieval (MIR), too, users' demographic attributes on an online music service play an important role mainly for music recommendation. As reported by Uitdenbogerd and Schnydel [22], music preference is affected by individual factors including age and ethnicity. In fact, it was revealed that music recommendation accuracy was improved by considering demographic attributes [21, 23, 27]. Motivated by these results, several studies in MIR have aimed to predict demographic attributes. The main way to perform this task is to use play logs obtained from an online music service and supervised machine learning techniques. Liu and Yang [12] predicted age and gender by using timestamps, song/artist metadata, and acoustic features of music signals. Wu *et al.* [25] also proposed methods to predict age and gender based on music metadata. They created two kinds of features: a TF-IDF-based one and a GSV(Gaussian super vector [3]))-based one. They applied support vector machine (SVM) to them. Krismayer *et al.* [10] predicted nationality in addition to age and gender based on music metadata (artist names and artist's tags). The details of their method are described in Section 4.2. By using their method, it was reported that demographic attributes can be predicted with high accuracy. The age was predicted with MAE of 4.13, and the gender and nationality were predicted with a classification accuracy of 81.36% and 69.37%, respectively.

Unlike these studies, our goal is to preserve users' demographic anonymity since some users do not want researchers or companies to predict their demographic attributes. Although several studies have discussed privacy problems (*e.g.*, the release of a user query log can lead to loss of privacy [8], confidential information such as medical conditions can be inferred from tweets [14], and

how should researchers deal with personal information in MIR [19]), our study is different from these studies in that we propose a concrete anonymization system and carried out experiments to evaluate how well it works.

3. FACTORS FOR REALIZING LISTENER ANONYMIZER

As we described in Section 1, we propose an approach that selects songs and camouflages the user's play log by playing these songs so that the user can preserve demographic anonymity. To enable an intuitive understanding of our idea, we give the following example story.

Emma is a 22-year-old French female. She is a Last.fm user and concealed her nationality when she signed up. She also uses Listener Anonymizer, which monitors the music she plays using Last.fm. One day, when Emma is listening to music using Last.fm with her smartphone, Listener Anonymizer detects that her nationality can be predicted as French with high accuracy from her play logs. Thus, Listener Anonymizer shows an alert message stating "your nationality can be predicted as French with a probability of 67%" on her smartphone screen and recommends three songs to her. Emma plays the songs to preserve the anonymity of her nationality.

Although this is just an example story, we need to consider various factors to realize Listener Anonymizer. Below, we marshal the factors from five viewpoints.

3.1 Definition of Anonymity

First, we define the anonymity of demographic attributes. In this paper, we propose two kinds of concepts for anonymity: *not-first-anonymity* and *k-flat-anonymity*. Suppose a demographic attribute d has n attribute values represented by $A_d = \{a_1, a_2, \dots, a_n\}$. For example, when d is nationality, $a_i \in A_d$ can be French, Japanese, etc. When user u has an attribute value $a_u \in A_d$ and conceals the attribute, given her music play log, we can compute the probability $p(a_i)$ for each attribute value in A_d by using an attribute prediction method ($0 \leq p(a_i) \leq 1$ and $\sum_{i=1}^n p(a_i) = 1$). In this case, *not-first-anonymity* is satisfied if the following condition is met: the rank of $p(a_u)$ among all attribute values is not the highest. In the case of Emma, *not-first-anonymity* is satisfied when the probability of French is not the highest.

In the case of *k-flat-anonymity*, the anonymity is satisfied if the following condition is met. Given the top k attribute values in terms of the probability, a_u is included in the top k attribute values and the probability gap between any two attribute values is lower than θ . In *k-flat-anonymity*, user's demographic attributes may be regarded as unpredictable because the top k attribute values have almost the same probabilities. In the example of Emma, suppose k and θ are set to 3 and 0.05, respectively, and the probabilities of French, Spanish, and German are 0.32, 0.28, and 0.29, respectively. In this case, because the probabilities of other nationalities are lower than those of the three nationalities and the probability gap between any two nationalities out of the three nationalities is lower than 0.05, *k-flat-anonymity* is satisfied even though French has the highest probability.

³ <https://twitter.com>

⁴ <https://facebook.com>

3.2 Method for Predicting Demographics

To realize Listener Anonymizer, simulating the method used in a demographics prediction system is required so that we can show an alert at the right time. However, since it is generally impossible to know the prediction method, we have to assume some prediction methods and propose a song selection method according to them. It is common to use music metadata extracted from the user's play log for predicting demographic attributes [10, 12, 25]. If we propose a song selection method that works well for state-of-the-art methods that are based on music metadata, we can say that our proposed method is robust to a certain extent. In light of the above, in this paper, we propose a method for selecting songs in Section 4.3 and show the effectiveness of this method through experiments in Section 5.

3.3 Timing of Camouflaging Play Logs

Listener Anonymizer selects songs and camouflages play logs in two main situations. One is when not-first-anonymity (or k -flat-anonymity) is no longer satisfied as we described in the example story at the beginning of this section. The other is when a user does not use a smartphone such as when she is sleeping or taking a bath. In the former case, since the user listens to her favorite songs before the songs are recommended by Listener Anonymizer, it should select as few songs as possible so that the user can soon resume listening to her favorite songs. In the latter case, since the user has enough time to play recommended songs and does not need to listen to them, a method that randomly selects many songs might be enough for recovering not-first-anonymity. Some users still hope to play as few recommended songs as possible to save on the packet communication fee.

3.4 User's True Demographics

In the preceding sections, we assumed that our anonymization system knows the user's true attribute values (e.g., Emma's nationality is French). That is, the user has to input the true demographic attributes before starting to use Listener Anonymizer. However, some users would not want to tell even the system their true demographics. When the system does not know the user's true demographic attribute, not-first-anonymity can be defined as follows: when the difference between the highest probability and the second highest probability is lower than θ . In this case, not-first-anonymity will not often be satisfied, and songs will be more frequently recommended to the user than the case where the system knows the true demographic attribute. In the example of Emma, suppose she does not tell her nationality to Listener Anonymizer. If she wants to preserve complete anonymity, she must play recommended songs at every alert, but this is a heavy burden for her. She could ignore an alert if the predicted nationality is wrong. However, if she plays only the recommended songs when the predicted nationality is French, Listener Anonymizer can estimate that Emma's nationality is French.

When the anonymization system knows a user's true attribute, the alert is displayed only when the true demographic can be predicted, which reduces the user's burden. In addition, if we can implement Listener Anonymizer as a

stand-alone smartphone application, the user's true demographics are stored only in the smartphone and are not sent to a server. In this case, users do not need to worry about leakage of demographic information from the server.

3.5 Anonymization of Multiple Demographics

We need to consider a situation where a user wants to preserve anonymity of more than one demographic attribute. For example, in the example of Emma, she anonymized only her nationality; now suppose she did not register her nationality, age, and gender on Last.fm. She may think that it does not matter if her age is predicted but may think it is a big problem if her nationality and gender are predicted. In such a case, she tells Listener Anonymizer the two demographic attributes that she wants to preserve the anonymity of. The system shows an alert and recommends songs when at least one demographic does not satisfy not-first-anonymity. If more than one demographic attribute does not satisfy not-first-anonymity at the same time, the system needs to select songs that can recover not-first-anonymity for all of the demographic attributes by playing recommended the songs. When a user tells the system many demographics that she wants to preserve the anonymity of, alerts may frequently be displayed, and this makes it difficult for the user to enjoy listening to her favorite songs. Therefore, the user has to select demographic attributes that she really wants to preserve the anonymity of.

4. CAMOUFLAGING PLAY LOGS

In Section 3, we described various factors to be considered to realize Listener Anonymizer. In this section, based on these factors, we discuss the situation dealt with in this paper, give the problem definition, and propose a method for selecting songs for camouflaging play logs.

4.1 Problem Definition

In terms of the type of anonymity, we use not-first-anonymity because of its simplicity. If we can show the effectiveness of our proposed method in not-first-anonymity, we will deal with k -flat-anonymity in future work. As for the timing of selecting songs and camouflaging play logs, we camouflage the user's play log with as few songs as possible. That is, given user u 's play log L_u that consists of m songs ($L_u = \{s_1, s_2, \dots, s_m\}$ where s_i represents a song), we aim to anonymize u 's confidential demographic attribute by selecting as few songs as possible. We assume our system knows the user's true demographic attributes. This assumption is reasonable because users will not hesitate to tell their demographics to the system if it is implemented as a stand-alone application as we described in Section 3.4. Finally, for preserving the anonymity of multiple demographics, since this paper deals with a new research problem, we consider single demographic anonymity as a first step. We are fully aware of the issue of multiple demographic anonymity; we leave this for future work.

Based on the above assumptions, our problem is defined as follows: "User u conceals an attribute value a_u in demographic d and wants to preserve not-first-anonymity regarding a_u . Given u 's play log consisting of m songs, we

verify if a_u satisfies not-first-anonymity. If it does not, we select as few songs as possible so that a_u can satisfy not-first-anonymity by playing them.”

4.2 Demographic Prediction Method

To the best of our knowledge, the state-of-the-art method for prediction of users’ demographic attributes of an on-line music service is the method proposed by Krismayer *et al.* [10]. They proposed a feature modeling approach. More specifically, given the users’ play logs in the training data, they extract an artist name and the artist’s tags as features for each song in each user’s play log. Here, only the top 10,000 artists and top 10,000 tags in terms of the popularity in the training data are used to create feature vectors. They compute the weight for each feature in the form of TF-IDF values and create a feature vector for each user. The feature vectors, each of which has 20,000 dimensions, are reduced to 500 dimensions by principal component analysis (PCA) [6]. Finally, the classifier is built by using SVM. Since their evaluation results showed that the polynomial kernel achieved high prediction accuracy on average, we assume that using the SVM with the polynomial kernel is the state-of-the-art method. More details can be found in Krismayer *et al.* [10]. Once a classifier is built, given a user’s play log, the classifier computes the probability distribution over demographic attribute values and outputs the attribute value that has the highest probability as the user’s predicted attribute value. We implemented this prediction method by ourselves with reference to Krismayer *et al.* [10].

4.3 Song Selection Method

When Emma’s nationality is predicted as French, Listener Anonymizer needs to select as few songs as possible that can anonymize her nationality. To achieve this, we aim to find songs that can largely increase the probability of the second-highest nationality (in this example, suppose Italian has the second highest probability). Since the feature vector corresponding to a song is compressed and the compressed vector is projected onto a new coordinate space by a polynomial kernel of SVM, it is difficult to find such songs based on the characteristics of an original 20,000-dimension vector. Instead, we assume such songs are played by users who are in the training dataset and are classified as Italian with high probabilities. From these users’ play logs, we extract effective songs by using feature ablation analysis [1]. More formally, given L_u , we first compute the probability distribution over n attribute values by using the method in Section 4.2. If $p(a_u)$ is not the highest among them, we do not need to do anything. If $p(a_u)$ is the highest, we select songs as follows.

Suppose $a_j (\neq a_u)$ has the second-highest probability after a_u . Let $U = \{u_1^t, u_2^t, \dots, u_q^t\}$ be a set of users in the training data. By developing the SVM classifier, user $u_i^t \in U$ has the probability $p(a_j, u_i^t)$ that represents the probability of u_i^t on a_j . From all users in U , we collect the top r users in terms of $p(a_j, u_i^t)$. Each user has her play log that consists of m songs. Suppose we remove the l th song from u_i^t ’s play log and compute the new probability of $p(a_j, u_i^t)$ by applying the SVM to the remaining $m - 1$

Table 1. Percentage of users who anonymize their demographic attributes. “✓” represents anonymization.

Age	Gender	Nationality	No. of users	%
✓	✓	✓	59,350	49.3
✓	✓		2,345	1.95
✓		✓	2,713	2.25
	✓	✓	454	0.377
✓			9,794	8.14
	✓		2,402	2.00
		✓	2,615	2.17
			4,0649	33.8

songs (let the new probability be $p'(a_j, u_i^t)$). If the score of $p(a_j, u_i^t) - p'(a_j, u_i^t)$ is large, we assume that the l th song is essential to increase the probability of a_j . Based on this idea, we compute the score for each of the $r \times m$ songs and collect the top c corresponding artists based on the score. After collecting the top c artists, we randomly select one artist; then we randomly select one song of the artist’s songs. By adding the song to L_u , we generate a camouflaged play log consisting of $m + 1$ songs. We repeatedly select a song and add it to L_u until the camouflaged play log satisfies not-first-anonymity.

5. EXPERIMENTS

In this section, we carry out experiments to evaluate the effectiveness of our proposed method.

5.1 Dataset

We used the Last.fm dataset provided by Schedl [20]. As for the user’s demographic attributes, this dataset includes age, gender, and nationality. Table 1 shows the numbers of users and the percentages for each of the combinations of confidential attributes, where “✓” indicates a confidential attribute. It can be observed that as many as 49.3% of users do not register any of their attributes, and 66.2% of them conceal at least one attribute. These statistics suggest the importance of preserving the user’s demographic anonymity, though there might be other reasons. The dataset also includes users’ play logs, each of which consists of the user ID, artist ID, track ID, and timestamp.

Following Krismayer *et al.* [10], we selected users who registered all of the three attributes, had equal to or more than 500 play logs, and had a nationality that was one of the 25 most common nationalities in terms of the number of users in the dataset. This gave us 32,991 users. We used 70% of them as training data and developed an SVM classifier. The remaining 30% of them were used as test data. Artists’ tags were collected by using the Last.fm API⁵. The number of classes of each attribute is as follows. The nationality consists of 25 classes that correspond to the top 25 most common nationalities, the gender has two classes that are male and female, and following Schedl *et al.* [21], the age was divided into seven age groups ([6 - 17], [18 - 21], [22 - 25], [26 - 30], [31 - 40], [41 - 50], and [51 - 60]).

5.2 Methods Comparison

5.2.1 Settings

Our first research question is “Is our proposed method able to camouflage play logs with fewer songs than a base-

⁵ <https://www.last.fm/api>

line method that randomly selects songs?” To answer this question, we count the number of songs selected by each method to preserve anonymity as follows. In this evaluation, for each user in the test dataset, we use the first 30 songs from the oldest songs in the play logs (*i.e.*, $m = 30$). For example, given the demographic attribute “nationality,” we first compute each user’s probability distribution over 25 nationalities when the 30 songs are played. We then sample 50 users whose nationality does not satisfy not-first-anonymity (*i.e.*, the user’s nationality has the highest probability among 25 nationalities). Note that in this case, each user’s play log in the training data also consists of 30 songs. Given a sampled user’s play log consisting of 30 songs, we add a song selected by our proposed method and compute the new probability distribution for the 31 songs. If the probability of the user’s nationality is not the highest among 25 nationalities, it means her nationality is anonymized and the song selection process ends; otherwise, we add a new song selected by our method and compute the probability distribution for the 32 songs. If the user’s nationality is not anonymized even after selecting the additional 30 songs, we stop the song selection process. In this way, we count the number of selected songs for all of the 50 users. In this evaluation, the values of r and c were set to 3 and 1, respectively. Note that r is the number of users used for selecting candidate songs and c is the number of artists used for recommending songs as we described in Section 4.3. The random baseline method (hereafter, the random method) randomly selects a song from all the songs in the dataset and counts the number of songs in the same manner as described above.

In addition to the proposed method and the random method, we use a popularity-based baseline method (hereafter, the popularity method). Intuitively, in this method, if we want to decrease the probability of France, for example, we select a song that is not popular in France but is popular in the other 24 nationalities. To achieve this, we rank all artists in each nationality where an artist’s score is the number of users who have listened to one of the artist’s songs at least once. The artists are ranked in descending order of their score. When a user u ’s nationality a_u is not anonymized, the popularity method first selects an artist b^* where

$$b^* = \arg \max_{b \in B} \frac{\sum_{a_i \in A_d \setminus \{a_u\}} (\text{rank}(a_i, b) - \text{rank}(a_u, b))}{|A_d \setminus \{a_u\}|}.$$

In the equation, B is the set of all artists and $\text{rank}(a_i, b)$ represents the rank of artist b in nationality a_i . Finally, a song of b^* is selected and added to u ’s play log.

Note that in this evaluation, we used the same 50 users for all of the three methods for a fair comparison.

5.2.2 Results

The results are shown in Figure 1 where each bar represents the average number of selected songs over 50 users. It can be observed that our proposed method outperformed the other two methods in all attributes. In the “gender” attribute, even the proposed method selected as many as 19.68 songs on average. Since the “gender” attribute has only two classes (male and female), the probability tended

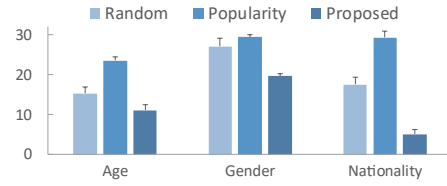


Figure 1. Comparison results between three methods. The y-axis is the average number of selected songs for camouflaging play logs. Error bars indicates the standard error.

to be strongly biased to one class. Thus, we presume that many songs were needed to fill the large gap. In the “nationality” attribute, the proposed method was especially effective: it selected less than one third of the songs selected by the random method.

The results of the popularity method were worse than those of the random method. This is because of the complexity of the demographic prediction method as described in Section 4.3. These results indicate that songs selected by the popularity method are rarely plotted to ideal points in the coordinate space created by an SVM polynomial kernel. Moreover, in the random method, a song that largely decreases the probability of the user’s confidential attribute can be selected by chance. Because of these reasons, the random method outperformed the popularity method.

5.3 Parameter Effect

5.3.1 Settings

Remind that our method has a parameter c that determines how many artists we use from the result of the feature ablation, although we set c to 1 in Section 5.2. Our second research question is “What is the relation between the value of c in the proposed method and the number of selected songs?” To answer the question, we change the value of c from 1 to 10 and count the number of selected songs for each c . In each of the three demographic attributes, the same 50 sampled users were used for all of the c values.

5.3.2 Results

Figure 2 shows the results. In the “age” and “nationality” attributes, the number of selected songs decreases when c changes from 1 to 2 and the number is at a minimum when c is 2 or 3; then the number increases with the increase of c . In particular, in the “age” attribute, when c is 2, only 3.22 songs are required on average to camouflage play logs consisting of 30 songs. In the “gender” attribute, although the number of selected songs decreases when c changes from 1 to 2, the minimum score was 9.28 when c is 10. From these results, we can say that selecting songs only from the best artist in terms of feature ablation analysis does not lead to the best result.

In addition to the decrease of the number of selected songs for large c , the increase of c has another advantage. When c is 1, songs are always selected from one artist to anonymize an attribute value. This may enable a company that wants to predict users’ demographic attributes to easily detect the camouflaged logs and predict the true attributes by removing the camouflaged logs. In contrast, when c is large, it becomes difficult to detect the

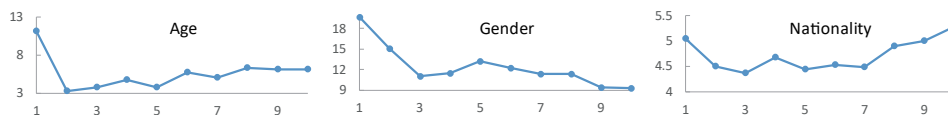


Figure 2. The relation between the value of c in the proposed method (x-axis) and the average number of selected songs for camouflaging play logs (y-axis).

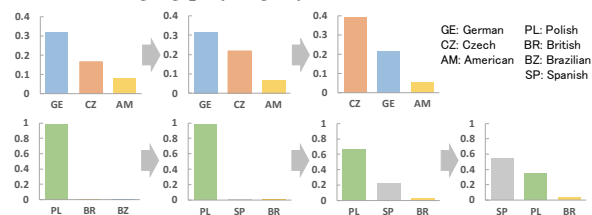


Figure 3. Examples of transition of probability distribution when two songs (top) and three songs (bottom) are selected for camouflaging play logs (y-axis: probability).

camouflaged parts in play logs. Moreover, by selecting songs from various artists, Listener Anonymizer may be able to expand the user’s music taste while preserving her anonymity. Figure 2 shows that even when c is 10, our method can anonymize an attribute with much fewer songs than the random method in all attributes. It would be useful to enable a user to select the value of c while thinking about a trade-off between the number of songs and the diversity of selected songs.

Figure 3 shows examples of the transition of probability distribution when two or three songs are selected by our proposed method in the “nationality” attribute. The value of c was set to 3. For visibility, we show only the top three nationalities in terms of the probability. In the top example, the user’s attribute is German. Listener Anonymizer can anonymize the user’s attribute by selecting two songs. In the bottom example, although the initial probability distribution is strongly biased to the user’s nationality (PL), this user can camouflage the play log by playing only three songs recommended by Listener Anonymizer.

6. DISCUSSION

In Section 5, we showed the effectiveness of the proposed method. However, since preserving demographic anonymity by camouflaging play logs is a quite new research theme, we discuss four important considerations.

6.1 Impact on Recommendation Accuracy

Since Listener Anonymizer camouflages play logs, it might degrade recommendation accuracy of music services. Although this paper dared to propose this controversial topic of research to give users an option of increasing the privacy and raise privacy issues in the MIR community, we are fully aware of the importance and usefulness of music recommendation to improve the user’s music experience. We hope that our paper could contribute to discuss a diversity of options for music experiences while balancing privacy versus accuracy in music recommendation.

6.2 User’s Taste in Music

In our method, the selected songs do not always match her taste in music. Even if those songs can camouflage the play logs, she might be reluctant to keep listening to the songs. Hence, it is beneficial to select songs by considering the

user’s taste in music. Many studies about song recommendation [11, 26, 28] and playlist generation [4, 13, 17] that can reflect the user’s taste in music have been conducted. By introducing the methods proposed in these studies, we plan to propose a song selection method that can balance camouflaging the play logs and taste. That would also be beneficial to satisfy both anonymization and good recommendation.

Considering the user’s taste has another advantage. If our method to camouflage play logs gains in popularity, companies that want to know the user’s demographic attributes will try to predict them by removing songs that camouflage her play log. By selecting songs that match the user’s taste, it becomes more difficult to detect songs that are played for camouflage.

6.3 Simulation of Multiple Prediction Methods

In our experiments, we assumed that the system knew that the method by Krismayer *et al.* [10] is used to predict the user’s demographic attributes. However, we cannot always know the prediction method in advance. When we do not know it, one strategy is to prepare multiple possible prediction methods and simulate them one at a time. An alert is issued when more than v methods detect that the user’s demographic attribute can be predicted with high accuracy. For small v , the degree of anonymity preservation is high but alerts are often issued and vice versa for large v . It would be useful for a user to be able to set the value of v according to the degree of anonymity she requires.

6.4 Real-time Monitoring of a Play Log

In our experiments, the number of songs in a given play log was set to 30. Hence, all logs in training data also consisted of 30 songs. However, this assumption is not sufficient to monitor the user’s played songs and recommend songs at the right time as we described in Section 3.3. This is because, when a user plays her first song, there is no play log in the training data consisting of only one song and we cannot correctly compute the probability distribution for the song. To solve this problem, we need to develop classifiers for various values of l , where l is the number of songs included in a play log.

7. CONCLUSION

In this paper, we proposed Listener Anonymizer that can preserve the user’s demographic anonymity by camouflaging her play log. Our experimental results show the effectiveness of our proposed method to select as few songs as possible. For example, in the “age” attribute, 15.3 songs were selected by the random method, while only 3.22 songs were selected by our method. Since this paper proposed a new concept, there are many remaining issues to be addressed as we discussed in Section 3 and 6. We plan to tackle them one by one and make Listener Anonymizer more flexible and useful.

8. ACKNOWLEDGMENTS

This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan.

9. REFERENCES

- [1] Eugene Agichtein, Ryen W. White, Susan T. Dumais, and Paul N. Bennett. Search, interrupted: Understanding and predicting search task continuation. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'12, pages 315–324, 2012.
- [2] John D. Burger and John C. Henderson. An exploration of observable features related to blogger age. In *In Computational Approaches to Analyzing Weblogs: Papers from the 2006 AAAI Spring Symposium*, pages 15–20, 2006.
- [3] William M. Campbell, Douglas E. Sturim, and Douglas A. Reynolds. Support vector machines using gmm supervectors for speaker verification. *IEEE Signal Processing Letters*, 13(5):308–311, 2006.
- [4] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. Playlist generation using start and end songs. In *Proceedings of the 9th International Conference on Music Information Retrieval*, ISMIR'08, pages 173–178, 2008.
- [5] Sumit Goswami, Sudeshna Sarkar, and Mayur Rustagi. Stylometric analysis of bloggers' age and gender. In *Proceedings of the 3rd International AAAI Conference on Weblogs and Social Media*, ICWSM'09.
- [6] Harold Hotelling. Analysis of a complex of statistical variables with principal components. *Journal of Educational Psychology*, 24:417–441 and 498–520, 1933.
- [7] David Huffaker and Sandra Calvert. Gender, identity, and language use in teenage blogs. *Journal of Computer-Mediated Communication*, 10(2), 2005.
- [8] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. "I know what you did last summer": Query logs and user privacy. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM'07, pages 909–914, 2007.
- [9] Margaret L. Kern, Johannes C. Eichstaedt, H. Andrew Schwartz, Gregory Park, Lyle H. Ungar and David J. Stillwell, Michal Kosinski, Lukasz Dziurzynski, and Martin E. P. Seligman. From "sooo excited!!!" to "so-proud": Using language to study development. *Developmental psychology*, 50(1):178–188, 2014.
- [10] Thomas Krismayer, Markus Schedl, Peter Knees, and Rick Rabiser. Prediction of user demographics from music listening habits. In *Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing*, CBMI'17, pages 8:1–8:7, 2017.
- [11] Dawen Liang, Minshu Zhan, and Daniel P. W. Ellis. Content-aware collaborative music recommendation using pre-trained neural networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, ISMIR'15, pages 295–301, 2015.
- [12] Jen-Yu Liu and Yi-Hsuan Yang. Inferring personal traits from music listening history. In *Proceedings of the 2nd International ACM Workshop on Music Information Retrieval with User-centered and Multimodal Strategies*, MIRUM'12, pages 31–36, 2012.
- [13] Beth Logan. Content-based playlist generation: Exploratory experiments. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, ISMIR'02, pages 295–296, 2002.
- [14] Huina Mao, Xin Shuai, and Apu Kapadia. Loose tweets: An analysis of privacy leaks on twitter. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, WPES'11, pages 1–12, 2011.
- [15] Dong-Phuong Nguyen, Rilana Gravel, Rudolf Berend Trieschnigg, and Theo Meder. "How old do you think I am?": A study of language and age in twitter. In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media*, ICWSM'13, pages 439–448, 2013.
- [16] Dong-Phuong Nguyen, Rudolf Berend Trieschnigg, A. Seza Dogruoz, Rilana Gravel, Mariet Theune, Theo Meder, and Franciska M.G. de Jong. Why gender and age prediction from tweets is hard: Lessons from a crowdsourcing experiment. In *Proceedings of the 25th International Conference on Computational Linguistics*, COLING'14, pages 1950–1961, 2014.
- [17] Elias Pampalk, Tim Pohle, and Gerhard Widmer. Dynamic playlist generation based on skipping behavior. In *Proceedings of the 6th International Conference on Music Information Retrieval*, ISMIR'05, pages 634–637, 2005.
- [18] Maarten Sap, Gregory Park, Johannes C. Eichstaedt, Margaret L. Kern, David Stillwell, Michal Kosinski, Lyle H. Ungar, and H. Andrew Schwartz. Developing age and gender predictive lexica over social media. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP'14, pages 1146–1151, 2014.
- [19] Pierre Saurel, Francis Rousseaux, and Marc Danger. On the changing regulations of privacy and personal information in mir. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, ISMIR'14, pages 597–602, 2014.
- [20] Markus Schedl. The lfm-1b dataset for music retrieval and recommendation. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, ICMR'16, pages 103–110, 2016.

- [21] Markus Schedl, David Hauger, Katayoun Farrahi, and Marko Tkalčič. On the influence of user characteristics on music recommendation algorithms. In *Proceedings of the 37th European Conference on Information Retrieval, ECIR'15*, pages 339–345, 2015.
- [22] Alexandra L. Uitdenbogerd and Ron G. van Schyn-del. A review of factors affecting music recommender success. In *Proceedings of the 3rd International Society for Music Information Retrieval Conference, ISMIR'02*, pages 204–208, 2002.
- [23] Gabriel Vigliensoni and Ichiro Fujinaga. Automatic music recommendation systems: Do demographic, profiling, and contextual features improve their performance? In *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR'16*, pages 94–100, 2016.
- [24] Yuan Wang, Yang Xiao, Chao Ma, and Zhen Xiao. Improving users' demographic prediction via the videos they talk about. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP'16*, pages 1359–1368, 2016.
- [25] Ming-Ju Wu, Jyh-Shing Roger Jang, and Chun-Hung Lu. Gender identification and age estimation of users based on music metadata. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR'14*, pages 555–560, 2014.
- [26] Zhe Xing, Xinxi Wang, and Ye Wang. Enhancing collaborative filtering music recommendation by balancing exploration and exploitation. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR'14*, pages 445–450, 2014.
- [27] Billy Yapriady and Alexandra L. Uitdenbogerd. Combining demographic data with collaborative filtering for automatic music recommendation. In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 3684 of *LNCS*, pages 201–207. 2005.
- [28] Kazuyoshi Yoshii, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *Proceedings of the 7th International Conference on Music Information Retrieval, ISMIR'06*, pages 296–301, 2006.

ON THE IMPACT OF MUSIC ON DECISION MAKING IN COOPERATIVE TASKS

Elad Liebman

The University of Texas at Austin
Computer Science Department
eladlieb@cs.utexas.edu

Corey N. White

Missouri Western State University
Department of Psychology
cwhite34@missouriwestern.edu

Peter Stone

The University of Texas at Austin
Computer Science Department
pstone@cs.utexas.edu

ABSTRACT

Numerous studies have demonstrated that mood affects emotional and cognitive processing. Previous work has established that music-induced mood can measurably alter people's behavior in different contexts. However, the nature of how decision-making is affected by music in social settings hasn't been sufficiently explored. The goal of this study is to examine which aspects of people's decision making in inter-social tasks are affected when exposed to music. For this purpose, we devised an experiment in which people drove a simulated car through an intersection while listening to music. The intersection was not empty, as another simulated vehicle, controlled autonomously, was also crossing the intersection in a different direction. Our results indicate that music indeed alters people's behavior with respect to this social task. To further understand the correspondence between auditory features and decision making, we have also studied how individual aspects of music affected response patterns.

1. INTRODUCTION

There is plentiful evidence that one's mood can affect how one processes information in a wide array of contexts and tasks. Previous work has established that positive mood induces a relative preference for positive emotional content and vice versa [6, 14]. Recent work has confirmed this effect is indeed induced by music that is culturally categorized as "happy" vs. "sad", and illustrated how the emotional content of music informs the apriori expectation for the emotional content of verbal stimuli [11]. As for non-emotional and quantitative decision-making, previous work has shown robust effects of loss aversion, whereby participants put more weight on potential losses than potential gains. In a recent study, Liebman et al. presented evidence for the complex impact of music-induced mood on risky decision-making in the context of gambling. They observed an overall improved stimulus processing in participants listening to "happy" music compared to "sad"

music, i.e., music-induced positive mood has led to better and faster decision-making overall [12].

Given the complexity and variability of the observed effects of music on decision-making in the context of different tasks, an inevitable question arises - how does music affect more complex tasks? More specifically, how does music affect complex decision-making that involves taking into consideration the agency of other entities? In this paper, we study the impact of music on decision behavior in the context of cooperative tasks, in which a person has to take into account the intentions of another agent when attempting to achieve their own goal. To this end, we design an experiment in which a person must cross a simulated intersection that is simultaneously being crossed by another autonomous agent, controlled by artificial intelligence. Our results indicate different types of music indeed have a differential effect on people's behavior in this setting.

The structure of the paper is as follows. In Section 3 we discuss our experimental design and how data was collected from participants. In Section 4 we present and analyze the results of our behavioral study. In Section 5 we examine more closely how music altered the participants' behavior in more specific contexts. In Section 6, we analyze how individual auditory components correlate with the behavioral patterns observed in our human study. In Section 2 we provide additional context about previous work leading up to this paper. Lastly, in Section 7 we recap our results and discuss them in a broader context.

2. RELATED WORK

Studies that induce mood through listening to happy/sad music have shown mood-congruent bias across a range of tasks. Behen et al. [9] showed participants happy and sad faces while they listened to positively or negatively valenced music and underwent fMRI. Participants rated the happy faces as more happy while listening to positive music, and the fMRI results showed that activation of the superior temporal gyrus was greater when the face and music were congruent with each other. In a study of mood and recall, De l'Etoile [4] found that participants could recall significantly more words when mood was induced (through music) at both encoding and retrieval.

Previous work at the intersection of musicology and cognitive science has also studied the connection between



© Elad Liebman, Corey N. White, Peter Stone. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Elad Liebman, Corey N. White, Peter Stone. "On the Impact of Music on Decision Making in Cooperative Tasks", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

music and emotion. As Krumhansel points out [10], emotion is a fundamental part of music understanding and experience, underlying the process of building tension and expectations. There is neurophysical evidence of music being strongly linked to brain regions linked with emotion and reward [2], and different musical patterns have been shown to have meaningful associations to emotional affectations [15]. Similarly, studies have indicated that mood also affects the perception of music [18]. Not only is emotion a core part of music cognitive processing, it can also have a resounding impact on people's mental state, and aid in recovery, as shown for instance by Zumbansen et al. [19] in the case of people suffering from Brocas aphasia. People regularly use music to alter their moods, and evidence has been presented that music can alter the strength of emotional negativity bias [3]. All this evidence indicates a deep and profound two-way connection between music and emotional perception.

Considering the impact of music on risk-related decision making, previous work has studied the general connection between gambling behavior and ambiance factors including music [5, 8, 17] in an unconstrained casino environment. Additionally, Noseworthy and Finlay have studied the effects of music-induced dissociation and time perception in gambling establishments [13].

Lastly, in the context of music and its impact on cooperation, not much research has been done to quantitatively explore how music impacts the cooperative and adversarial behaviors of participants in social settings. Greitemeyer presented evidence that Exposure to music with prosocial lyrics reduces aggression [7]. From a different perspective entirely, Baron was able to show how environmentally-induced mood helped improve negotiation and decrease adversarial behavior [1]. To the best of our knowledge, this is the first work to study how different types of music differentially affect people's decision-making in the context of tasks involving other agents.

3. EXPERIMENTAL SETUP

In this section we describe the details of the experiment conducted in this study. First, we describe the overall procedure. We proceed to describe the participants, the autonomous car behavior, the music selected for the experiment, and the data collected for analysis.

3.1 Procedure

In this study, participants were given control of a simulated vehicle crossing an intersection. They had three control options - speed forward, go in reverse, and brake. In addition to the human-controlled vehicle, another vehicle, controlled autonomously by an artificial agent, was also crossing the intersection from a different direction. If the two cars collided, they would crash. Participants were instructed to safely cross the intersection without crashing. Participants were also instructed that the autonomous car would generally respect the laws of traffic but cannot be blindly relied upon to drive safely. Each time both vehicles

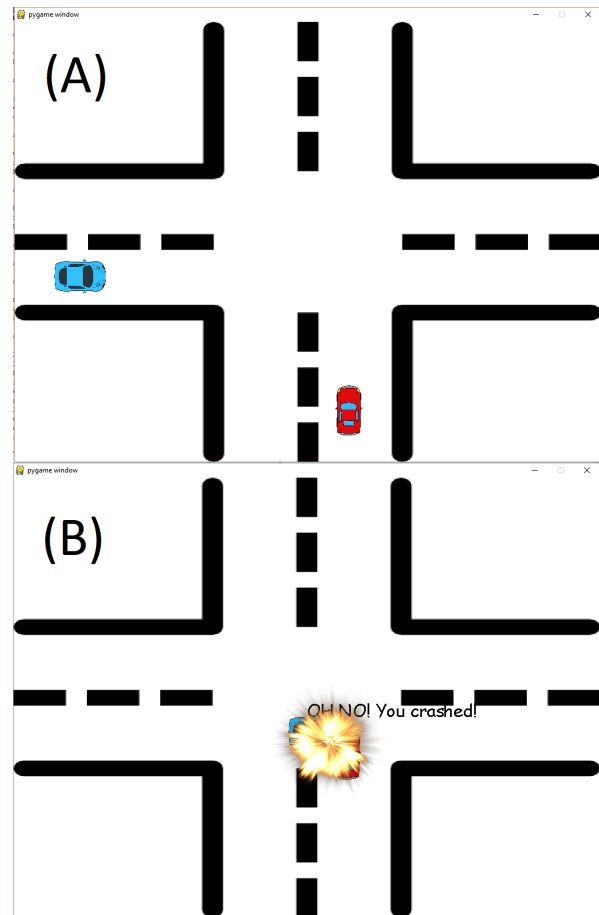


Figure 1. (A) A screen capture of the experiment. The red car was controlled by the participant. The blue car was controlled autonomously. (B) A collision would result in a crash, as demonstrated in this screen capture. After the crash, the trial terminates and the next trial begins.

cleared the intersection and reached the end of the screen safely, the trial would end and a new trial would commence (a 2 second pause was introduced between trials). The experiment was divided into 8 blocks of 12 trials (for a total of 96 trials per participant). In each trial the behavior of the blue vehicle was randomized, determining its speed and the amount of time it would wait by default in the intersection if it had arrived to the intersection first. In each block, a different song was played, alternating between positive and negative music across blocks. The order of the songs was counterbalanced across subjects. A 3 second pause before the beginning of each block to make sure the new song had started before a new trial commenced. Each experiment lasted approximately 20 minutes. A snapshot of the experiment is presented in Figure 1.

3.2 Participants

For this paper we have originally collected data from 20 participants. All participants were graduate students who volunteered to participate in the study. Two participants were filtered out for behaving uniformly without paying

attention to the experimental conditions (always going forward at the beginning of each trial without slowing, stopping or paying attention to the autonomous vehicle), leaving a total of 18 participants. Note that the comparisons of interest were within participants (happy vs. sad music). Thus, the sample size was sufficient to detect statistically significant differences in behavior between these conditions.

3.3 Autonomous Car Behavior

The key variability in stimuli in this experiment was presented through randomization of the autonomous car behavior. The three main aspects of the autonomous car behavior that were variable were its speed approaching the intersection, how long it would wait in the intersection before going forward if it arrived to the intersection first, and how fast it would move into the intersection and onward after entering the intersection. Participants were instructed not to blindly rely on the autonomous car's behavior, but in the scope of this experiment we opted to have the autonomous car always give right of way if the human-controlled car made it to the intersection first. The consequence of this was that the decision whether to give right of way or move forward was almost always in the hands of the human participant. Indeed, one of the explicit goals of this study were to examine how different music-induced mood would affect people's aggressiveness vs. their inclination to give right of way.

3.4 Music

The music used for this experiment is the same as that used in [11]. It is a collection of 8 publicly available songs which was surveyed to isolate two clear types - music that is characterized by slow tempo, minor keys and somber tones, typical to traditionally "sad" music, and music that has upbeat tempo, major scales and colorful tones, which are traditionally considered to be typical to "happy" music. The principal concern in selecting these musical stimuli, rather than their semantic categorization as either happy or sad, was to curate two separate "pools" of music sequences that were broadly characterized by a similar temperament (described above), and show they produced consistent response patterns. In [11], it has been shown experimentally that the selected music was effective for inducing the appropriate mood. This was done by selecting a separate pool of 40 participants and having them rate each song on a 7-point Likert scale, with 1 indicating negative mood and 7 indicating positive mood. It was then shown that the songs designated as positive received meaningfully and statistically significantly higher scores than those denoted as sad.

4. OVERVIEW OF RESULTS

In this section we survey the key findings of the study, examining the participants' behavior globally (that is, across all types of circumstances and autonomous vehicle behavior).

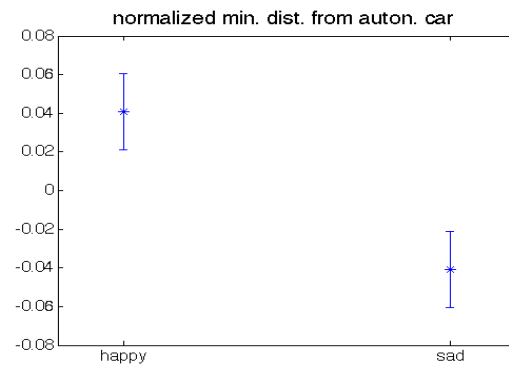


Figure 2. Normalized minimal distance kept from the autonomous car by the participants in the sad and happy music conditions (here and elsewhere, bars represent std. error). Participants tended to keep a lower minimal distance when listening to sad music.

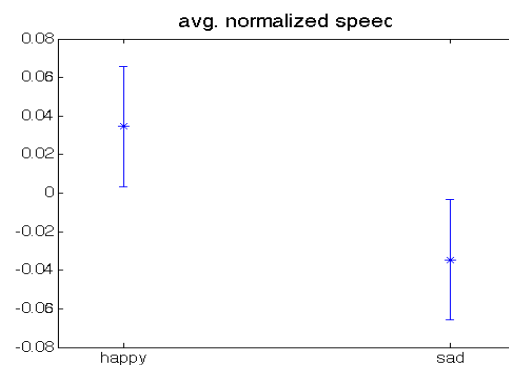


Figure 3. The average normalized speed of the participants in the happy and sad music conditions. Participants were more likely to go faster when listening to happy music.

4.1 Minimal Distance from Autonomous Car

The most statistically significant difference ($p < 0.05$ using a paired t-test) across all trials was that participants listening to sad music kept a lower minimal distance overall from the autonomous car compared to when they were listening to happy music. In other words, their behavior when listening to sad music was riskier and less considerate ("cutting it closer" with respect to how much margin for error they kept when entering the intersection). This result is illustrated in Figure 2.

4.2 Driving Speed

Participants also differed in their driving speed in the sad and happy music conditions (significant at $p < 0.05$ using a paired t-test). Overall, participants were more likely to go fast in the happy music condition compared to the sad music condition, as reflected in Figure 3.

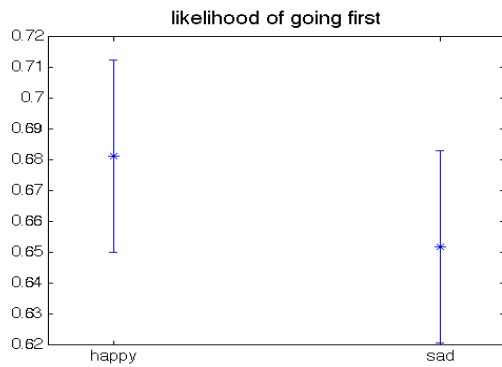


Figure 4. The likelihood of the participants to go first into the intersection in the sad and happy music conditions. Participants were more likely to go first when listening to happy music.

4.3 Right of Way

Another difference, which is strongly related to the previous observation, and is borderline significant¹ (at $p < 0.1$ using a paired t-test) was that participants listening to happy music were more likely to go into the intersection first compared to when they were listening to sad music, as illustrated in Figure 4.

5. BREAKDOWN OF USER BEHAVIOR UNDER DIFFERENT TRIAL CONDITIONS

In this section we consider how different music induced different participant behavior when breaking down the trials by the different types of autonomous car behavior. It is worth noting that the observation made in the previous section held under most partitions of the trial data.

5.1 Behavior under Different Autonomous Car Intersection Wait Times

If we compare how participants behaved when the autonomous vehicle waited < 4 seconds at the intersection, the difference in the participants' driving speed because dramatically more accentuated in the happy vs. sad music conditions. Additionally, the participants' difference in wait times at the intersection in the happy and sad music conditions also becomes more differentiated when we only consider trials in which the autonomous car waited less than < 4 seconds. These observations are presented in figures 5(a) and 5(b), and are both statistically significant with $p < 0.05$ using an unpaired t-test.

¹ A 0.1 threshold for testing the significance of p-values is accepted in the context of relatively small samples sizes. Nonetheless, we strive to use these measures responsibly in our choice of language, thus using the equally common term "borderline significance" to describe results with $p\text{-value} < 0.1$ but > 0.05

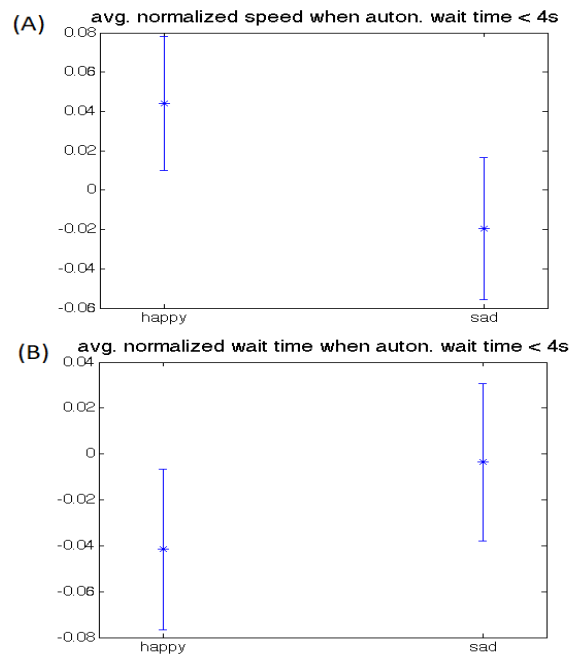


Figure 5. (a) Normalized average per-trial speed of participants in the happy and sad music conditions, specifically in the case that the autonomous vehicle waited less than 4 seconds. (b) Normalized per-trial time waiting at the intersection of participants in the happy and sad music conditions, specifically in the case that the autonomous vehicle waited less than 4 seconds.

5.2 Behavior under Different Autonomous Car Average Speed

A similar related trend to that observed in the previous section were observed when considering the average speed of the autonomous car. In trials in which the average speed of the autonomous vehicle was above the median, people were slower to drive and took longer to wait at the intersection while listening to sad music, compared to when listening to happy music (again with $p < 0.05$ using an unpaired t-test).

6. IMPACT OF MUSICAL PARAMETERS ON USER BEHAVIOR

The partition between "positive" and "negative" mood-inducing songs is easy to understand intuitively, and in itself is enough to induce the different behavioral patterns discussed in the previous section. However, similarly to the analysis performed in [11] and [12], we are interested in finding a deeper connection between the behavior observed in the experiment and the different characteristics of music. More exactly, we are interested in finding the correspondence between various musical features, which also happen to determine how likely a song is to be perceived as happy or sad, and the driving decision-making manifested by participants. To this end, we considered the 8 songs used in this experiment, extracted key character-

izing features which we assume are relevant to their mood classification, and examined how they correlate with the subject behavior we observed.

6.1 Extracting Raw Auditory Features

We focused on four major auditory features: a) overall tempo; b) overall “major” vs. “minor” harmonic character (we will refer to this feature as “major chord ratio” for simplicity); c) average amplitude, representing overall loudness; and d) maximum amplitude, representing peak loudness. Features (a), (c) and (d) were computed using the Librosa library [16]. To compute feature (b), we implemented the following procedure, similar to that described in [11]. For each snippet of 20 beats an overall spectrum was computed and individual pitches were extracted. Then, for that snippet, according to the amplitude intensity of each extracted pitch, we identified whether the dominant harmonic was major or minor. The major/minor score was defined to be the proportion of major snippets out of the overall song sequence. Analysis done in [11] confirms these features are indeed associated with our identification as “positive” vs. “negative”. Having labeled “positive” and “negative” as 1 and 0 respectively, a Pearson correlation of 0.7–0.8 with p -values ≤ 0.05 was observed between these features and the label. Significance was further confirmed by applying an unpaired t -test for each feature for positive vs. negative songs (p -values $< .05$).

6.2 Results

Overall, the most prominently influential aspect of the music as observed by statistical analysis is the loudness of the music. Additional effects were observed relating to tempo and major chord ratio, but they did not meet the same criteria for significance.

6.3 Loudness and Overall Time Out of Intersection

The normalized overall time out of intersection is the total time it took the participant to drive up to the intersection, wait, and cross the intersection, normalized per subject. The normalized time out of the intersection was statistically significantly ($p < 0.05$ ²) inversely correlated with both the average loudness ($r = -0.72$) and the maximum loudness ($r = -0.77$) of the music. The correspondence between the average loudness and the overall time out of intersection is presented in Figures 6 (the findings for the maximum loudness are similar). In other words, the louder the music was, the faster people were to complete the task.

6.4 Loudness, Speed, Time Stopped, and Minimal Distance

Loudness also impacted various aspects of participant behavior that are related to the participants’ driving speed and overall aggressiveness. These results are borderline significant at $p < 0.1$ for all correlations reported in this subsection.

² P -values for correlation are results obtained by analysis of the distribution of correlation values given the null hypothesis.

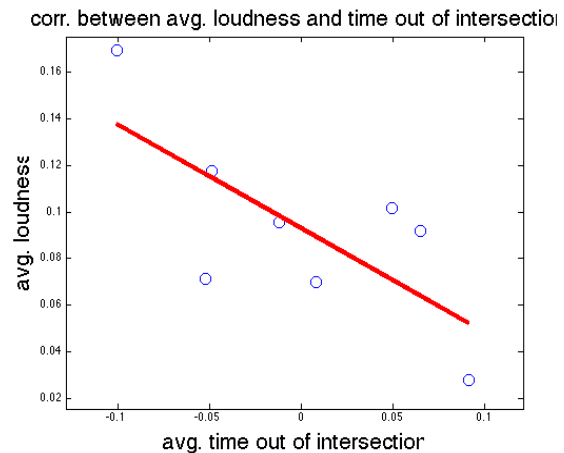


Figure 6. Correlation between the average loudness of the music and the normalized total time out of the intersection for the participants.

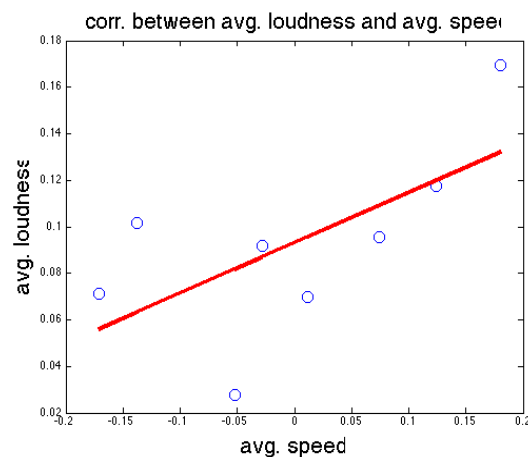


Figure 7. Correlation between the average loudness and the average speed of the participants.

- Most straightforwardly, the average loudness was positively correlated ($r = 0.65$) with the normalized average speed of the participants, meaning that participants drove faster when listening to louder music. This result is illustrated in Figure 7.
- Similarly, other metrics reflect overall speed, including the minimum speed, the median speed and the initial speed (speed after 1 second from the beginning of the trial) were positively correlated with $r > 0.6$.
- The overall normalized time the participants stopped at the intersection was inversely correlated at $r = -0.67$ with the average loudness, meaning people were faster to continue into the intersection when listening to louder music. This finding is presented in Figure 8
- Lastly, the minimal distance the participants kept from the autonomous car was positively correlated

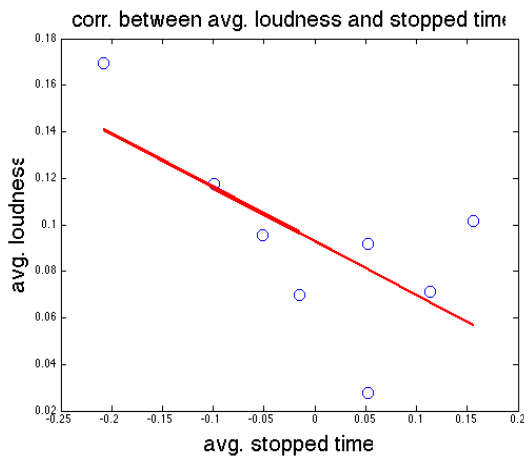


Figure 8. Correlation between the average loudness and the average time the participants stopped at the intersection.

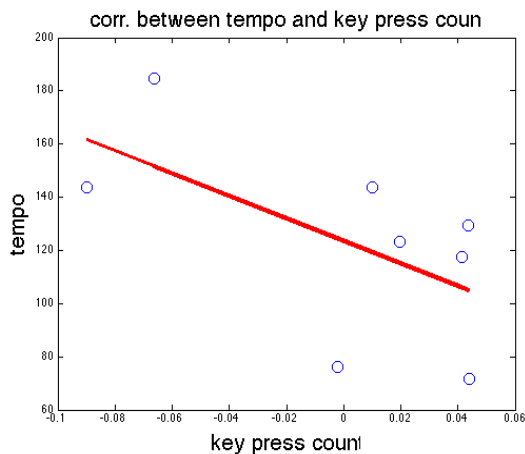


Figure 9. Correlation between the normalized key press count of the participants and the tempo.

with the average loudness, meaning the louder the music was, the higher the minimal distance was. Considering the other findings in this section and the fact that the minimal distance and the average speed are positively correlated at $r = 0.75$ (and $p < 0.05$), it is reasonable to assume this relationship is a result of the impact of loudness on the participants' speed rather than an indication of how loud music increases people's risk aversion, for instance.

6.5 Tempo and Hesitancy

The total number of key presses per trial, normalized per participant, is a good proxy for hesitancy in decision making (speeding and slowing down, going forward and braking, etc). Interestingly, the key press count was inversely correlated to the tempo ($r = -0.59$ and $p < 0.1$), suggesting faster music reduced people's hesitancy. This results is presented in Figure 9.

6.6 Additional Observations

Beyond the results reported thus far in this section, several relationships between musical features and participant behavior were observed that did not meet the $p < 0.1$ criterion for significance, but came sufficiently close to merit mention:

- The normalized key press count was also inversely correlated with the major chord ratio (at $r = -0.52$), implying it's possible that music that leans heavier towards major harmonies also reduces hesitancy in the participants.
- The the major chord ratio was also positively correlated with the maximum speed of the participants, and the minimal distance the participants kept from the autonomous car, at $r = 0.54$ and $r = 0.52$, respectively.
- The tempo was positively correlated with both the average and the max speed at $r = 0.53$ for both.

7. SUMMARY AND DISCUSSION

In this study we analyzed how people's decision-making behavior is affected by music in the context of a social task which requires a certain level of cooperation to avoid adverse consequences. Participants were required to drive a simulated car through an intersection while another car, controlled by an autonomous agent, was also crossing from a different direction. Examining the results reveals a compound picture befitting the subtleties of the performed task. While happy music induced some aspects of behavior that could be described as more social, namely that participants kept a safer distance from the other car when crossing, they also manifested less social behavior by driving faster and being less likely to let the autonomous vehicle go first. All in all, our initial expectation that happier music would make people more cooperative was not supported by the findings. Conversely, it can be argued that sad music made people slower and more cautious, and therefore safer to their environment and to the other agent specifically. This study is the first step towards a better understanding of how music informs people's decision-making in multi-agent environments that require some level of cooperation. Follow-up work would help refine our observations, as well as possibly leverage them in the context of human-agent interaction and negotiation.

8. ACKNOWLEDGMENTS

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-1637736, IIS-1651089, IIS-1724157), Intel, Raytheon, and Lockheed Martin. Peter Stone serves on the Board of Directors of Cogitai, Inc. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

9. REFERENCES

- [1] Robert A Baron. Environmentally induced positive affect: Its impact on self-efficacy, task performance, negotiation, and conflict. *Journal of Applied Social Psychology*, 20(5):368–384, 1990.
- [2] Anne J Blood and Robert J Zatorre. Intensely pleasurable responses to music correlate with activity in brain regions implicated in reward and emotion. *Proceedings of the National Academy of Sciences*, 98(20):11818–11823, 2001.
- [3] Jie Chen, Jiajin Yuan, He Huang, Changming Chen, and Hong Li. Music-induced mood modulates the strength of emotional negativity bias: An erp study. *Neuroscience Letters*, 445(2):135–139, 2008.
- [4] Shannon K de l'Etoile. The effectiveness of music therapy in group psychotherapy for adults with mental illness. *The Arts in Psychotherapy*, 29(2):69–78, 2002.
- [5] Laura Dixon, Richard Trigg, and Mark Griffiths. An empirical investigation of music and gambling behaviour. *International Gambling Studies*, 7(3):315–326, 2007.
- [6] Rebecca Elliott, Judy S Rubinsztein, Barbara J Sahakian, and Raymond J Dolan. The neural basis of mood-congruent processing biases in depression. *Archives of general psychiatry*, 59(7):597–604, 2002.
- [7] Tobias Greitemeyer. Exposure to music with prosocial lyrics reduces aggression: First evidence and test of the underlying mechanism. *Journal of Experimental Social Psychology*, 47(1):28–36, 2011.
- [8] Mark Griffiths and Jonathan Parke. The psychology of music in gambling environments: An observational research note. *Journal of Gambling Issues*, 2005.
- [9] Jeong-Won Jeong, Vaibhav A Diwadkar, Carla D Chugani, Piti Sinsoongsud, Otto Muzik, Michael E Behen, Harry T Chugani, and Diane C Chugani. Congruence of happy and sad emotion in music and faces modifies cortical audiovisual activation. *NeuroImage*, 54(4):2973–2982, 2011.
- [10] Carol L Krumhansl. Music: A link between cognition and emotion. *Current Directions in Psychological Science*, 11(2):45–50, 2002.
- [11] Elad Liebman, Peter Stone, and Corey N. White. How music alters decision making - impact of music stimuli on emotional classification. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, pages 793–799, 2015.
- [12] Elad Liebman, Peter Stone, and Corey N White. Impact of music on decision making in quantitative tasks. In *ISMIR*, pages 661–667, 2016.
- [13] Theodore J Noseworthy and Karen Finlay. A comparison of ambient casino sound and music: Effects on dissociation and on perceptions of elapsed time while playing slot machines. *Journal of Gambling Studies*, 25(3):331–342, 2009.
- [14] Kristi M Olafson and F Richard Ferraro. Effects of emotional state on lexical decision performance. *Brain and Cognition*, 45(1):15–20, 2001.
- [15] Sébastien Paquette, Isabelle Peretz, and Pascal Belin. The musical emotional bursts: a validated set of musical affect bursts to investigate auditory affective processing. *Frontiers in psychology*, 4, 2013.
- [16] Brian McFee ; Matt McVicar ; Colin Raffel ; Dawen Liang ; Douglas Repetto. Librosa. <https://github.com/bmcfee/librosa>, 2014.
- [17] Jenny Spenwyn, Doug JK Barrett, and Mark D Griffiths. The role of light and music in gambling behaviour: An empirical pilot study. *International Journal of Mental Health and Addiction*, 8(1):107–118, 2010.
- [18] Jonna K Vuoskoski and Tuomas Eerola. The role of mood and personality in the perception of emotions represented by music. *Cortex*, 47(9):1099–1106, 2011.
- [19] Anna Zumbansen, Isabelle Peretz, and Sylvie Hébert. The combination of rhythm and pitch can account for the beneficial effect of melodic intonation therapy on connected speech improvements in brocas aphasia. *Frontiers in human neuroscience*, 8, 2014.

VENUERANK: IDENTIFYING VENUES THAT CONTRIBUTE TO ARTIST POPULARITY

Emmanouil Krasanakis¹ Emmanouil Schinas^{1,2}
Symeon Papadopoulos¹ Yiannis Kompatsiaris¹ Pericles Mitkas²

¹CERTH-ITI, Thessaloniki, Greece - {maniospas,manosetro,papadop,ikom}@iti.gr

²AUTH, Thessaloniki, Greece - manosetro@issel.ee.auth.gr, mitkas@auth.gr

ABSTRACT

An important problem in the live music industry is finding venues that help expose artists to wider audiences. However, it is often difficult to obtain live music audience data to tackle this task. In this work, we investigate whether important venues can instead be inferred through social media data. Our approach consists of employing bipartite graph ranking algorithms to help discover important venues in artist-venue graphs mined from Facebook. We use both well-established algorithms, such as BiRank, and a modification of their common iterative scheme that avoids the impact of possibly erroneous heuristics to the ranking, which we call VenueRank. Resulting venue ranks are compared to those obtained from feature extraction for predicting the most listened artists and large listener increments in Spotify. This comparison yields high correlation between venue importance for listener prediction and bipartite graph ranking algorithms, with VenueRank found more robust against overfitting.

1. INTRODUCTION

In the music industry, artists aim to present themselves to wide audiences. Therefore, it is important for them to gain as much exposure as possible from their live performances. In turn, this exposure can influence their popularity, as expressed by the size of their audience.

In this paper we try to identify which performances offer higher exposure. Factors such as timing and other recent events can influence this. Listener geolocation has also been found to contribute to artist popularity prediction [3, 21]. Consequently, it is reasonable to hypothesize that performing in certain venues could contribute more to artist popularity. Having access to a ranking of venues based on expected exposure could be valuable for artists and their agents; when confronted with different options regarding their future performances, they could consider these rankings as an important decision criterion.

To rank venue exposure, one could try to predict it using machine learning algorithms. Unfortunately, there are difficulties in quantifying the notion of exposure, not least of which is that real-life data may misrepresent audience size and reactions. For example, participating in a large event held in a well-known venue with many other artists may contribute less to gaining popularity compared to an artist-focused event. A viable alternative to measuring venue exposure, which we also adopt in this work, is to instead estimate whether venues contribute to artist popularity (e.g. the number of listeners in music services) from a machine learning perspective. To do so, we can employ feature extraction methods to identify the most important venues that help determine and increase artist popularity.

However, even this formulation depends on obtaining live music audience data required for supervised training. Such data are not necessarily easy to obtain, as they are typically considered confidential. Therefore, in this work we attempt inferring important venues through unsupervised training, which does not require such data.

In particular, given a graph representation, where artists are linked with venues they have performed in, we use graph ranking algorithms to rank venues. To validate whether this approach ranks venues based on offered exposure, we compare the produced ranks with venue importances obtained through feature extraction for predicting popular artists and artist popularity increments. We find that ranking methods can be more informative than raw social media measures in predicting important venues.

2. BIPARTITE GRAPH RANKING

2.1 Motivation for Graph Ranking

We can organize data pertaining to artists \mathcal{A} and venues \mathcal{V} where they have performed as bipartite graphs, i.e. graphs in which vertices form two disjoint sets linking only to each other. To analyze the importance of venues based on the structure of such artist-venue graphs, we employ ranking algorithms, which are used to determine the relative importance of nodes given a graph's structure [17]. These algorithms often operate under the premise that nodes linked to a higher number of important nodes are also more important.

Formulations such as HITS [14] further refine this concept by recognizing that there can be two types of important nodes; authorities that provide important information



© Emmanouil Krasanakis, Emmanouil Schinas, Symeon Papadopoulos, Yiannis Kompatsiaris, Pericles Mitkas. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: Emmanouil Krasanakis, Emmanouil Schinas, Symeon Papadopoulos, Yiannis Kompatsiaris, Pericles Mitkas. "VenueRank: Identifying Venues that Contribute to Artist Popularity", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

and hubs that point to a lot of information sources. A node's authority is then derived from its predecessors' hub scores and its hub score is derived from its successors' authority scores, forming an iterative process.

The distinction between authorities and hubs is of great interest when applied on bipartite graphs, especially if the links between disjoint set elements represent the same type of relation. In this case, we can formulate that one of those sets contains only authorities and the other only hubs.

For example, in our artist-venue graph setting, where venues always represent locations where artists have performed, artists can be considered as authorities and venues as hubs from which popularity-related authority stems. Intuitively, this means that artists are considered more important if they have performed in more important venues, whereas venues are considered more important if more important artists have performed there.

2.2 Ranking based on Prior Ranks

Formally, bipartite graph ranking algorithms attempt to rank nodes in a graph defined by a (weighted) adjacency matrix $W : \mathcal{V} \times \mathcal{A}$ between the disjoint groups \mathcal{A}, \mathcal{V} based on heuristically estimated prior ranks.

Prior ranks¹ are supported by most graph ranking algorithms and are used to introduce ranking bias that is driven by information unrelated to graph structure. For example, in web searches [17] prior ranks place more weight on the pages more similar to the search query. In bipartite graphs, prior ranks often represent an informed belief about ranks and help attract the solution towards convergence. However their usage can reduce the robustness of extracted structural characteristics (see Subsection 2.3).

As demonstrated by He et al. [12], previous bipartite graph ranking algorithms follow similar formulations. In particular, if S and S' are normalizations of W and its transposition W^T respectively, a_0, v_0 are prior ranks that initially estimate node ranks for the two bipartite graph groups and r_a, r_v are prior rank elimination parameters, approaches follow a common recursive rule for calculating bipartite graph group ranks as recursions $n \rightarrow \infty$:

$$a_{n+1} = (1 - r_a)a_0 + r_a S' v_n \quad (1a)$$

$$v_{n+1} = (1 - r_v)v_0 + r_v S a_n \quad (1b)$$

In practice, this iterative process stops when rank differences converge to a stable set of values. In this work, we empirically adopt a simple stopping criterion across algorithms that stops after rank changes become small enough:

$$\|a_{n+1} - a_n\|_2^2 + \|v_{n+1} - v_n\|_2^2 < 0.1 \quad (2)$$

Differences between bipartite graph ranking algorithms lie in the way normalization is performed on the adjacency matrix and its transposition. If D_A, D_V are two diagonal matrices containing the node degrees of the disjoint sets \mathcal{A}, \mathcal{V} , those algorithms perform normalization as:

$$S = D_v^{-p_v} W D_a^{-p_a} \quad (3a)$$

$$S' = D_a^{-p_a} W^T D_v^{-p_v} \quad (3b)$$

¹ Prior ranks have also been referred to as 'query vectors' [12].

where p_a, p_v are non-negative constants specific to each algorithm (see Table 1). These constants determine whether degree normalization should be performed row-wise or column-wise or whether the normalization should produce a stochastic matrix.

Algorithm	p_a	p_v
HITS [14]	0	0
Co-HITS [6]	1	0
BGRM [19]	1	1
BGER [1]	0	1
BiRank [12]	$\frac{1}{2}$	$\frac{1}{2}$

Table 1: Different parameters between bipartite graph ranking algorithms.

The advantage of the iterative scheme demonstrated in Eqn (1) over more general ranking schemes, which do not take the bipartite nature of the graph into account, is that the former converges fast to unique stationary solutions, as demonstrated below.

a) If $r_a, r_v < 1$ then substituting Eqn (1) into itself as $n \rightarrow \infty$ yields:

$$a_\infty = (I - r_a r_v S' S)^{-1} [r_a (1 - r_v) S' v_0 + (1 - r_a) a_0]$$

$$v_\infty = (I - r_a r_v S S')^{-1} [r_v (1 - r_a) S a_0 + (1 - r_v) v_0]$$

Although this solution can also help analytically derive node ranks a_∞, v_∞ , doing so can be computationally intensive, since it requires matrix inversion. For that reason, all previous approaches adopt the iterative scheme, which is computationally efficient, especially when W is sparse.

b) If $r_a = r_v = 1$ then:

$$\begin{aligned} a_\infty = S' v_\infty &\Rightarrow (S' S - I) a_\infty = 0 \\ v_\infty = S a_\infty &\Rightarrow (S S' - I) v_\infty = 0 \end{aligned}$$

Therefore, if $S' S, S S'$ are stochastic matrices, i.e. if $p_a + p_v = 1$ in Eqn (3), their largest eigenvalue is 1 and thus a_∞, v_∞ are their principal eigenvectors respectively. In this case, the iterative scheme resembles the power method for calculating the principal eigenvectors. However, due to absence of vector normalization after each step, large enough ranks may grow uncontrollably and fail to converge [16].

2.3 VenueRank

The above formulation of bipartite graph ranking algorithms relies heavily on the correctness of the prior ranks a_0, v_0 to produce accurate ranks. If the prior ranks are only partially correct (e.g. are only sparsely filled) ranking algorithms may converge to much different values. Furthermore, structure-related information could be more useful for important venue detection than prior rank heuristics. Hence, there exist cases where eliminating the effect of prior ranks is desirable [15].

In such cases, the previous bipartite graph ranking algorithms eliminate the prior ranks by selecting $r_a = r_v = 1$.

However, as discussed above, numeric convergence is not theoretically guaranteed for these parameters. For example, BiRank fails to converge for these parameters when run on data gathered in Subsection 3.1.

Therefore, we propose modifying the previous iterative process to gradually remove the dependency on initial prior ranks across iterations:

$$a_{n+1} = (1 - r_a)a_n + r_a S' v_n \quad (4a)$$

$$v_{n+1} = (1 - r_v)v_n + r_v S a_n \quad (4b)$$

Similarly to before, as $n \rightarrow \infty$ we obtain $a_\infty = S' v_\infty$ and $v_\infty = S a_\infty$ and thus a_∞, v_∞ become the principal eigenvectors of $S'S$ and SS' respectively, as long as the latter are stochastic matrices.

This iterative process differs from previous ones in that it stabilizes on these eigenvectors for any non-zero parameters r_a, r_v . As a result, we can retrieve theoretical guarantees [16] that there exist small enough r_a, r_v that make it converge. Moreover, we can see that:

$$S'S = D_a^{-p_v - p_a} W^T D_v^{-p_a - p_v} W$$

$$S'S = D_v^{-p_a - p_v} W D_a^{-p_v - p_a} W^T$$

Therefore, for constant $p_a + p_v = 1$ the eigenvectors of these two matrices remain the same and Eqn (4) converges to the same ranks regardless of the type of normalization defined by these two parameters.

In short, we have shown that, for the iterative scheme demonstrated in Eqn (4), which we will call VenueRank, it suffices to select any $p_a + p_v = 1$ and small enough r_a, r_v to converge to bipartite graph ranks where the effect of prior ranks is eliminated.

3. EXPERIMENTS

3.1 Data Collection

We collected two types of data for our experiments; data from *Facebook* about artist and venue pages and the respective number of listeners for those artists from *Spotify*. We use Facebook data to run bipartite graph ranking algorithms and Spotify data to extract the ground truth with which to evaluate these algorithms.

We started with a collection of 542 artists, for which we were granted access to their number of streams and listeners in Spotify Analytics² from 1 January 2015 to 3 May 2019. We also used the Facebook Graph API³ to automatically find Facebook pages for those artists and manually removed artists with erroneously matched pages. After this step, the collection comprises 323 artists, for whom we can retrieve both the monthly number of listeners in Spotify and their Facebook page.

Next, we retrieved the events published in the discovered Facebook pages dating later than 1 January 2014, as well as the venues that hosted them. This process results in a tripartite graph with nodes representing artists, events and venues (see Figure 1).

² <https://analytics.spotify.com/>

³ <https://developers.facebook.com/docs/graph-api/>



Figure 1: Artist (red) and venue (blue) pages in Facebook alongside their associated events (yellow) of the largest connected subgraph of the dataset that contains a Finnish rock band called ‘The Rasmus’ (green).

This graph contains a total of 105,251 events that took place in 4,051 venues across 72 timezones (see Figure 2). Using the events in that graph as indicators of the appearance of an artist in a venue, we infer a bipartite artist-venue graph, which we use for our analysis. Artists associated with a non-zero number of venues number 224 and they are associated with a total of 2,392 venues.

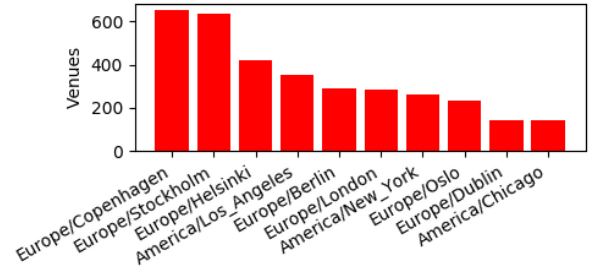


Figure 2: Timezones with more than 20 venues each.

Our dataset contains the number of Spotify listeners per month for each artist. From these listeners we procure artist popularity based on the total number of listeners for each artist, as well as the increase of the number of artist listeners for each month, which yields 8,619 datapoints across all artists. The number of total listeners spans a wide range of magnitudes. Hence, denoting the number of listeners for month m of an artist as L_m , we define:

$$popularity = \log(1 + \sum_m L_m)$$

which yields the normal-like artist distribution shown in Figure 3. We also quantify the relative increments of monthly listeners as:

$$inc_m = \min\{L_m/L_{m-1} - 1, 1\} \text{ when } L_{m-1} \neq 0$$

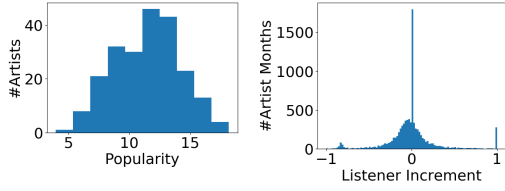


Figure 3: Artist popularities (left) and the distribution of relative listener increments (right).

3.2 Ground Truth Construction

It is difficult to directly extrapolate the exposure granted by venues using only artist *popularity* and *inc_m* data. However, we can employ a feature extraction scheme to obtain venue importances reflecting their contribution to predicting these quantities. Based on the systemic property of graph ranking algorithms in Section 2.2, to rank based on the nodes' positions on the graph, we can then evaluate the quality of those algorithms by measuring whether higher ranked venues are actually more important for prediction. Effectively, this would assert that higher ranks represent higher exposure. This way, venue importances extracted from machine learning on Spotify data can be used as ground truth against which to validate ranking algorithms, which do not utilize such data.

The machine learning setting for feature extraction can be either a regression or a classification task (popular-vs-unpopular). Here, we focus on the latter, since the classification task leads to clearer separation between venues that lead to each of the two target classes. Indeed, regression tasks using only venues to predict *popularity* and *inc_m* values yield high error rates, whereas the binary classifiers demonstrated below boast high predictive capabilities.

Labeling and Feature Selection

To set up the venue ranking task, we distinguish high *popularity* and *inc_m* values by performing outlier detection [13] and considering outliers residing in the 20% right tail of their distributions⁴ as popular and high increment ones respectively. We then use methods that perform robust feature extraction [10] based on these labels.

To do so, we consider venues as binary artist features to predict *popularity*, whereas we use exponential decay to model the decreasing influence [9] on *inc_m* of performing in a venue held at month *m_v* as $\exp(-\frac{m-m_v}{2})$ if $m_v \leq m$ and 0 otherwise. Using these feature values, the feature extraction mechanism then identifies which features (i.e. venues) contribute the most to label prediction (i.e. high artist popularity and high listener increments).

Unfortunately, outliers represent a small fraction of all datapoints and hence cause imbalance between label priors. Imbalance often affects classification validity and can distort or bias estimated feature importances. To alleviate such concerns, we employ SMOTE oversampling [2] to generate synthetic popular artist profiles, so that the number of popular artists becomes equal to the number of un-

⁴ To obtain the outliers residing in the right 20% distribution tail, we use the z-score detection threshold 0.84 for those greater than the median.

popular ones. We prefer an oversampling scheme, because the small number of collected artists prohibits an under-sampling one. This process is summarized in Figure 4.

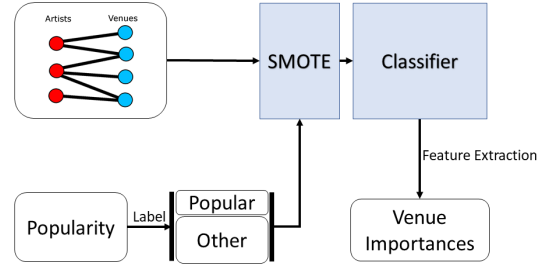


Figure 4: Extracting venue importances.

Classifier

We use the random forest classifier of the *sklearn* Python package [18] with an entropy feature selection criterion. Compared to other classification algorithms, random forests calculate feature importances during the training process and do not require tuning. On the other hand, they can produce lower importances for cross-correlated features. To improve the robustness of such features, we instead deploy an ensemble of random forests [20], which averages importance scores obtained from 10 random forests. To avoid erroneously overstating the importance of unique venue appearances, we train these ensembles and produce importances only for venues in our data where at least 2 artists have performed, which number 602.

Validation

To assert the validity of feature importances assigned by random forest ensembles, we performed leave-one-out cross-validation on trained random forests across 13 training repetitions. For predicting high *popularity* labels we obtained 9% false positive error rate (i.e. rate of assigning unpopular artists as popular) and 7% false negative error rate (i.e. rate of assigning popular artists as unpopular), whereas for predicting high *inc_m* labels we obtained 29% false positive error rate and 33% false negative error rate. Since error rates reflect informed classification, venue importances obtained through this process can indeed be considered as the ground truth for subsequent experiments. These error rates indicate that *popularity* importances are more accurate, although from a methodological standpoint causation is better explored by *inc_m* importances.

3.3 Compared Ranking Algorithms

In this section, we explore the performance of unsupervised ranking algorithms (such as those presented in Section 2) that aim to rank venues using only Facebook data. These algorithms require prior rank estimations, which we heuristically infer through metadata obtained from the Facebook Graph API. In particular, we estimate artist and venue prior ranks respectively as:

$$a_0 = \log(1 + fans + mentions/2)$$

$$b_0 = |\text{events in venues}|$$

We alternatively tried calculating venue prior ranks by summing of the size of all events hosted in a venue, heuristically estimated by $size = \max\{0, \log(1 + attending + interested/2 + maybe/2 - noReply/2 - declined)\}$. However, this reduced all BiRank evaluations more than 30% compared to their currently reported values. We compare the following algorithms:

Raw: Estimates venue ranks as their prior ranks.

RFE: Feature extraction using random forest ensembles, similarly to ground truth construction, but aiming to predict high artist prior ranks.

BiRank: BiRank on the artist-venue bipartite graph extracted from Facebook data. Unless stated otherwise, this method uses parameters $r_a = r_v = 0.85$, which are a common empirical selection for ranking algorithms [8, 12].

VenueRank: VenueRank on the artist-venue bipartite graph extracted from Facebook data. As argued above, VenueRank eventually removes the effect of prior ranks. Although inconsequential from a theoretical standpoint, we follow previous conventions and reasoning well-established for BiRank [12], to select the parameters $r_a = r_v = 0.85$ and $p_a = p_v = 0.5$, unless stated otherwise.

Evaluation Measures

To evaluate bipartite venue ranking algorithms, we compare the ranks they produce when applied on Facebook data with the ground truth importances extracted from Spotify data in Subsection 3.2. Our aim is to find whether venues are correctly ranked by unsupervised ranking algorithms. To this end, we measure rank similarities using the robust Spearman correlation coefficient [5], which is computed as a Pearson correlation between the cardinal ranks of compared quantities. It must be noted that, due to the possibility of negative exposures being found more important, the supremum of Spearman correlation can be less than 1. This, however, does not affect the fact that Spearman correlations closer to 1 indicate that higher ranked venues are more important and thus boast higher exposure.

Additionally, if $rank_{GT}$ lists venues in a descending order of their ground truth importances and $rank_C$ in descending order of their calculated ranks, we can define the overlap between the top N venues:

$$overlap(N) = \frac{|rank_{GT}[0 : N] \cap rank_C[0 : N]|}{N}$$

To evaluate the overall *overlap* curves across all venues, we also measure their Area Under Curve (AUC) [11], which is a fair method of curve comparison. To calculate this area, we perform numerical trapezoid integration of overlaps and normalize the result by dividing it with the width of the horizontal axis. Higher AUC values represent better ability to recognize both high-exposure and low-exposure venues.

3.4 Results

Experiments are performed under two variants of unsupervised training on Facebook data: venue ranking on the

same 224 artists (**224A**) (including venues with only one performance) as those used for ground truth construction and venue ranking using all 542 artists (**542A**) and their respective venues. Since the latter dataset contains more artists and venues, it presents a more challenging setting. Using both variants for evaluation helps identify which algorithms generalize better and are more robust.

In Table 2, we can see that BiRank and VenueRank achieve high correlation values with the ground truth in the 224A dataset. However, BiRank heavily relies on accurate prior ranks to do so and does not perform well in the larger 542A dataset, where it produces worse estimations compared to even its prior ranks. This implies that BiRank exhibits overfitting characteristics. On the other hand, both RFE and VenueRank boast great robustness in that they are less affected by the transition to the larger 542A dataset. Consequently, VenueRank exhibits high performance and is more suited to real-world applications, since it is more robust to artist-venue graph changes.

Since there exist -to the best of our knowledge- no previous studies that can serve as comparison for venue correlations, common guidelines [7] suggest that we can resort to the Cohen convention [4] to classify extracted venue ranks as strongly correlated between VenueRank and ground truth importances across all experiments.

Algorithm	<i>popularity</i>		<i>inc_m</i>	
	224A	542A	224A	542A
Raw	36%	38%	44%	42%
BiRank	70%	33%	76%	28%
RFE	57%	51%	64%	49%
VenueRank	71%	63%	69%	60%

Table 2: Spearman correlation coefficient between ground truth venue rankings and rankings produced by algorithms.

Feature importances forming the *popularity* and *inc_m* ground truths are themselves significantly correlated, with 58% Spearman correlation coefficient. This indicates that venues characterizing popular artists also tend to characterize higher listener increments and conversely.

Figure 5 shows the overlap between various algorithms and the ground truth for different numbers of top venues. We can see that, for a small number of top venues (i.e. less than 200), ranking methods do not produce high overlap with ground truth venues. However, for a greater number of venues, they rank highly a large portion of important venues. A curve over a larger area is more important than only identifying top venues, because ranking methods may be used to compare middle-ranked or low-ranked venues to the majority of artists instead of only the most popular ones. AUC results corroborate the previous ones. In particular, BiRank again performs better than other methods under perfect information, whereas VenueRank performs better than other methods and is thus more robust in the case of the more challenging 542A dataset variant.

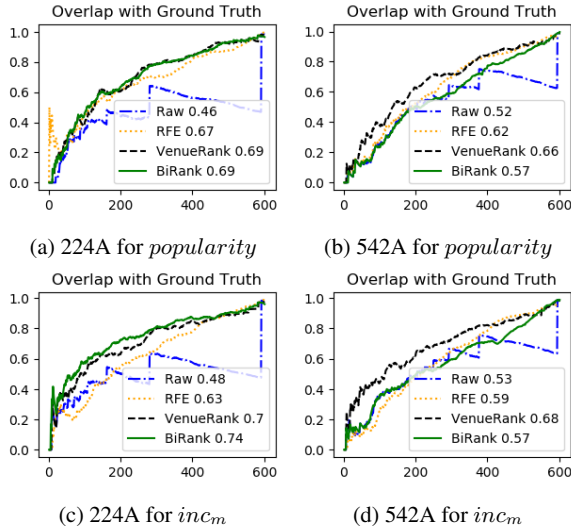


Figure 5: Curves and AUCs of high-ranked venue overlap between ranking algorithms and the ground truth.

3.5 Convergence when Ignoring Prior Ranks

In Figure 6 we present the convergence time of BiRank with respect to its iterative scheme parameters r_a, r_b . We can see that execution time increases asymptotically to infinity as prior ranks are ignored, i.e. $r_a = r_b \rightarrow 1$. Instead, VenueRank exhibits similar behavior for these parameters convergence-wise, and it always converges to the same stationary solution, as long as these parameters are not close enough to 1 to cause numeric errors. Furthermore, that solution is the same as BiRank when the effect of the prior ranks is completely eliminated. Hence, when the effect of prior ranks is undesirable, it is preferable to employ VenueRank instead of selecting BiRank parameters close to 1, if we want to achieve faster convergence.

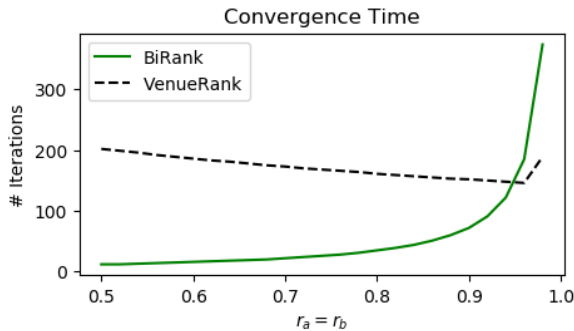


Figure 6: Convergence time of BiRank (green solid line) and VenueRank (dashed black line). VenueRank always has the stationary solution of BiRank with $r_a = r_b \rightarrow 1$.

3.6 Case Study

Finally, we conduct a case study, where we try to find important venues in the city of Stockholm, Sweden through venue ranking algorithms. To this end, we used two online

articles^{5, 6} to gather a total of 10 highly recommended venues and find their rankings obtained from running the previous algorithms on all 542 artists and 5,041 venues. In Table 3 we show their rank within the ordered list of all 635 Stockholm venues in our dataset (rank of the highest-ranked venue is 1).

VenueRank places 8 of the 10 venues in the top 50 ranks, whereas other methods place at most 5 of the 10 venues in the top 50 ranks. VenueRank’s performance is commendable, given that our dataset also includes popular parks and hotels often used for live music acts, which would not be recommended in the above articles, and that worse ranks often stem from incomplete data (e.g. the dataset contains only two events hosted in ‘Nalen’).

	Raw	RFE	BiRank	VenueRank
Annexet	95	105	147	40
Berwaldhallen	48	94	96	46
Cirkus	67	61	193	23
Debaser Medis	9	29	9	11
Debaser Rest.	3	8	4	1
Fasching	55	50	125	33
Nalen	195	97	172	113
Pet Sounds Bar	81	19	343	49
Sodra Teatern	39	1	24	2
Stallet	11	63	13	68

Table 3: Rank cardinality for recommended venues compared to other Stockholm venues.

4. CONCLUSIONS AND FUTURE WORK

In this work, we introduce VenueRank as a modification of the common iterative scheme of bipartite graph ranking algorithms that removes dependence on prior ranks while ensuring convergence. We then explore ranking algorithms that help identify which venues help predict artist popularity. Experiments on real-life data show that VenueRank applied on a Facebook artist-venue graph can robustly identify which venues are correlated with more popular artists and actively contribute to increasing their Spotify listeners. In particular, in a setting with partially inaccurate information, VenueRank yields substantial improvement compared to other unsupervised ranking algorithms.

In part, this shows that graph structure can be more important than rough social network metrics when predicting high-exposure venues. Furthermore, it demonstrates that there exists a clear link between graph structure and venue exposure that increases artist popularity.

In the future, we plan to carry out more detailed experiments on larger datasets. Furthermore, from a theoretical perspective, the VenueRank iterative scheme can also be combined with BiRank to produce more robust solutions across the whole parameter space. Finally, we propose improving venues ranks by taking into account how they contribute to the exposure of lower popularity artists.

⁵ <https://theculturetrip.com/europe/sweden/articles/the-6-best-live-music-venues-in-stockholm>

⁶ <https://scandinaviantraveler.com/en/places/7-best-music-venues-in-stockholm>

5. ACKNOWLEDGEMENTS

The authors would like to thank Playground Music Scandinavia for providing artist data. This work is partially funded by the European Commission under the contract number H2020-761634 FuturePulse.

6. REFERENCES

- [1] Lei Cao, Jiafeng Guo, and Xueqi Cheng. Bipartite graph based entity ranking for related entity finding. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 130–137. IEEE Computer Society, 2011.
- [2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.
- [3] Zhiyong Cheng and Jialie Shen. Just-for-me: An adaptive personalization system for location-aware social music recommendation. In *Proceedings of international conference on multimedia retrieval*, page 185. ACM, 2014.
- [4] Jacob Cohen. Statistical power analysis for the behavioral sciences 2nd edn, 1988.
- [5] Christophe Croux and Catherine Dehon. Influence functions of the spearman and kendall correlation measures. *Statistical methods & applications*, 19(4):497–515, 2010.
- [6] Hongbo Deng, Michael R Lyu, and Irwin King. A generalized co-hits algorithm and its application to bipartite graphs. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–248. ACM, 2009.
- [7] Joseph A Durlak. How to select, calculate, and interpret effect sizes. *Journal of pediatric psychology*, 34(9):917–928, 2009.
- [8] Nadav Eiron, Kevin S McCurley, and John A Tomlin. Ranking the web frontier. In *Proceedings of the 13th international conference on World Wide Web*, pages 309–318. ACM, 2004.
- [9] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 241–250. ACM, 2010.
- [10] Isabelle Guyon and André Elisseeff. An introduction to feature extraction. In *Feature extraction*, pages 1–25. Springer, 2006.
- [11] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.
- [12] Xiangnan He, Ming Gao, Min-Yen Kan, and Dingxian Wang. Birank: Towards ranking on bipartite graphs. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):57–71, 2017.
- [13] Boris Iglewicz and David Hoaglin. *Volume 16: how to detect and handle outliers, The ASQC basic references in quality control: statistical techniques*, Edward F. Mykytka. PhD thesis, Ph. D., Editor, 1993.
- [14] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.
- [15] Joel C Miller, Gregory Rae, Fred Schaefer, Lesley A Ward, Thomas LoFaro, and Ayman Farahat. Modifications of kleinberg’s hits algorithm using matrix exponentiation and web log records. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 444–445. ACM, 2001.
- [16] Erkki Oja and Juha Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of mathematical analysis and applications*, 106(1):69–84, 1985.
- [17] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.
- [18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [19] Xiaoguang Rui, Mingjing Li, Zhiwei Li, Wei-Ying Ma, and Nenghai Yu. Bipartite graph reinforcement model for web image annotation. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 585–594. ACM, 2007.
- [20] Yvan Saeys, Thomas Abeel, and Yves Van de Peer. Robust feature selection using ensemble feature selection techniques. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 313–325. Springer, 2008.
- [21] Markus Schedl and Dominik Schnitzer. Location-aware music artist recommendation. In *International Conference on Multimedia Modeling*, pages 205–213. Springer, 2014.

CONTENT-BASED USER MODELS: MODELING THE MANY FACES OF MUSICAL PREFERENCE

Eva Zangerle

Universität Innsbruck
Department of Computer Science
eva.zangerle@uibk.ac.at

Martin Pichl

Universität Innsbruck
Department of Computer Science
martin.pichl@uibk.ac.at

ABSTRACT

User models that capture the musical preferences of users are central for many tasks in music information retrieval and music recommendation, yet, it has not been fully explored and exploited. To this end, the musical preferences of users in the context of music recommender systems have mostly been captured in collaborative filtering-based approaches. Alternatively, users can be characterized by their average listening behavior and hence, by the mean values of a set of content descriptors of tracks the users listened to. However, a user may listen to highly different tracks and genres. Thus, computing the average of all tracks does not capture the user's listening behavior well. We argue that each user may have many different preferences that depend on contextual aspects (e.g., listening to classical music when working and hard rock when doing sports) and that user models should account for these different sets of preferences. In this paper, we provide a detailed analysis and evaluation of different user models that describe a user's musical preferences based on acoustic features of tracks the user has listened to.

1. INTRODUCTION

In the last decade, the amount of tracks available on streaming platforms has literally exploded. Users are supported in exploring and wading through these music collections by means of personalization—mostly by recommender systems that provide users with a list of tracks they might like to listen to. Such personalization is central for the success of streaming platforms as it eases the task of discovering new and enjoyable music for users.

For music information retrieval (MIR) and particularly, for personalization tasks in this context, modeling the musical preferences of users is naturally a central aspect. Yet, user modeling for MIR and music recommender systems (MRS) has hardly been investigated [4,32,33]. To this end, music recommender systems have mostly been realized by means of collaborative filtering (CF) methods [16] or more

advanced factorization approaches [17], where recommendations are based on interactions between users and items. Such systems are agnostic to content features as recommendations are computed based on the similarity of users (or items) based on their co-occurrence in the listening histories of all users. On the other hand, (the less adopted) content-based recommender systems [22] compute recommendations based on the similarity of content descriptors of tracks. Also, hybrid recommender systems combining CF- and content-based approaches have been proposed [7].

In the field of MIR, tracks are traditionally characterized by content descriptors—these range from detailed features such as MFCCs [21] to high-level content descriptors such as acousticness, tempo or danceability (e.g., provided by the Spotify platform¹). While these features are widely used to characterize single tracks, for a user model that captures the user's preferences well, these features have to be aggregated across all tracks the user has listened to. To this end, Pichl et al. [30] utilized content descriptors of tracks for representing a user's musical preference by computing the average acoustic features across all tracks the user has listened to. They also find that users create different playlists that feature different acoustic characteristics—implying that these playlists correspond to different sets of preferences of a user (which may naturally be context-related) and stress the need for more comprehensive user models to describe users' musical preferences [30]. Similarly, Wang et al. [36] state that people prefer different music for different daily activities. Along these lines, we argue that users may exhibit different preferences depending on the context and e.g., listen to more energetic music when doing sports or calming music when being at home [36]. These different preferences cannot be sufficiently reflected in a model that averages the characteristics of all the tracks a user listened to. In a probabilistic user model, Bogdanov et al. [4] characterize a user in a semantic feature space derived from low-level content features by utilizing Gaussian Mixture Models.

In this paper, we build upon and extend these previous works by proposing different user models to describe the musical preferences of users based on content descriptors of tracks. We perform a large-scale evaluation of these models in a track recommendation task based on 8 million listening events of 13,000 users. Our experiments show



© Eva Zangerle, Martin Pichl. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Eva Zangerle, Martin Pichl. "Content-based User Models: Modeling the many faces of musical preference", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ <https://developer.spotify.com/web-api/get-several-audio-features/>

that utilizing a user model based on a user's specific preferences regarding different types of music (modeled probabilistically by GMMs) complemented with a user's general musical preference achieves the best results. Our results show that in terms of recommendation quality, the proposed models contribute to substantially improved recommendation performance. We believe that our findings can contribute to improved user models for music recommender systems and generally, MIR tasks.

The remainder of this paper is organized as follows. Section 2 discusses related work and Section 3 presents the features utilized and the dataset underlying our experiments. Section 4 presents the user models proposed. Section 5 details the experimental setup and Section 6 presents the results of our study, which are discussed in Section 7. Section 8 concludes the paper and discusses future work.

2. RELATED WORK

Generally, Schedl et al. [32, 33] note that the user and his/her preferences are often not considered when it comes to MIR and MRS tasks. Particularly, the authors lay out that user modeling for such tasks has hardly been explored and evaluated yet.

To this end, content descriptors have widely been used in MIR and MRS. For similarity search, often a content-based similarity measure is used for matching queries and a music database [9, 20, 35, 39]. In the context of music recommender systems, Yoshii et al. [38] propose a hybrid recommender system that combines collaborative filtering via user ratings and content-based features modeled via Gaussian Mixture Models over MFCCs by utilizing a Bayesian network. Also, Liu [20] investigates different distance metrics for content-based recommender systems. Recently, also deep learning-based hybrid MRS have also been proposed [37]. In regards to user modeling for MRS, Bogdanov et al. compute a user's musical preferences by a set of exemplary tracks that the user enjoyed. They model the user's preference in a latent semantic space based on a set of diverse content features and propose a set of similarity-based recommender systems. One system models a user by a Gaussian Mixture Model based on the proposed semantic audio feature space. The authors evaluated these recommender systems in a user experiment with twelve users. As for musical preferences of users, Pichl et al. found in a large-scale study of Spotify users that music streaming users listen to different types of music. Those types can be observed via k-means clustering of content descriptors of tracks. They also found that users organize their music in playlists based on these types and stress the importance of more comprehensive user models to describe users' musical preferences [30]. Along these lines, we specifically investigate user models that are solely based on content descriptors. We propose six user models and compare these in a large-scale offline study based on a recommendation task comprising 13,000 users and 8 mio. listening events.

3. DATASET AND FEATURES

The main data source used in our experiments is the publicly available LFM-1b dataset [31], which provides the full listening histories of 120,322 Last.fm users. For each listening event (i.e., a certain user listening to a certain track), information about the track, artist, album and user is available. Besides the information contained within the LFM-1b dataset, we also require content features to describe tracks. Following the lines of, e.g., [1, 25, 30], we propose to rely on the Spotify API² to gather the following content descriptors for each track:

1. *Danceability* describes how suitable a track is for dancing and is based "on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity."
2. *Energy* measures the perceived intensity and activity of a track. This feature is based on the dynamic range, perceived loudness, timbre, onset rate and general entropy of a track.
3. *Speechiness* detects presence of spoken words. High speechiness values indicate a high degree of spoken words (talk shows, audio book, etc.), whereas medium to high values indicate e.g., rap music.
4. *Acousticness* measures the probability that the given track is acoustic.
5. *Instrumentalness* measures the probability that a track is not vocal (i.e., instrumental).
6. *Tempo* quantifies the pace of a track in beats per minute.
7. *Valence* measures the "musical positiveness" conveyed by a track (i.e., cheerful and euphoric tracks reach high valence values).
8. *Liveness* captures the probability that the track was performed live (i.e., whether an audience is present in the recording).

These features are high-level descriptors of the acoustic content of tracks. We argue that they are nevertheless representative and hence, the obtained results should give a good impression on the differences of the user models. We expect our findings to also hold for more complex and lower-level content descriptors such as e.g., Mel-Frequency Cepstral Coefficients (MFCC) [21].

To obtain these features for all tracks of the dataset, we apply the following steps: we perform a conjunctive query for the <track, artist, album>-triples extracted from the LFM-1b dataset using the Spotify search API³ to gather the Spotify URI of each track. This URI is subsequently used to query the acoustic features API⁴. Finally, we add tracks for which can obtain all required features to the dataset⁵.

Since the set of tracks a user listened to may also contain outlier tracks that may distort the user profile, we propose to remove outlier tracks from this set by apply-

² A detailed description of these features and the API can be found at <https://developer.spotify.com/web-api/get-several-audio-features/>.

³ <https://developer.spotify.com/web-api/search-item/>

⁴ <https://developer.spotify.com/web-api/get-several-audio-features/>

⁵ Except for tempo, all of these features are given in the range of [0, 1] and for tempo, we apply a linear min-max scaling.

Item	Value
Listening Events (LE)	8,457,205
Users	12,995
Tracks distinct	965,293
Min. LE per User	1
Q_1 LE per User	252
Median LE per User	478
Q_3 LE per User	826
Max. LE per User	21,660
Avg. LE per User	650.80 (\pm 713.99)

Table 1. Dataset statistics.

ing the median absolute deviation (MAD) outlier detection method [19]. We consider a feature value an outlier if it is not within $M \pm a \cdot MAD$, where M is the median of this particular feature across all tracks of a user and MAD is the median absolute deviation of these values. We consider a value an outlier if it is not within within three MAD s around the median, setting a rather conservative threshold $a = 3$ as proposed by [19]. Lastly, a track is considered as an outlier in the list of tracks of a particular user if one of its features is considered an outlier and consequently removed from the user listening history.

Applying this procedure results in a dataset of 55,149 users, 394,944,868 listening events and 3,478,399 distinct tracks. We randomly sample users from this dataset for our experiments, where we require each user to have more than 100 listening events to ensure that our user models are representative. We present basic statistics about the resulting dataset in Table 1. As can be seen, on average, each user has listened to 651 tracks.

4. USER MODELS

In the following, we present the proposed user models to capture user’s listening preferences. We specifically focus on modeling users solely by acoustic features of tracks they listened to and deliberately neglect other information that could contribute to a user model (e.g., demographic user aspects, cultural information or further contextual features that might improve MRS and MIR performance).

4.1 Feature Space

Based on the users, tracks and their acoustic features within the dataset, we perform the following steps prior to the computation of the user models. Most of the proposed models require clustering tracks based on their acoustic features to find groups of tracks that exhibit similar features. Given that we aim to perform a large-scale analysis of the proposed user models (we perform the analysis on 8 million tracks and 13,000 users), these clustering computations are computationally intensive. Hence, we firstly perform a proximity-preserving dimension reduction on the input data by applying UMAP (Uniform Manifold Approximation and Projection) [23]. Also, the use of latent representations of elements in the musical ecosystem (users, tracks, etc.) has been to be effective in MIR and

MRS tasks [18, 26, 27]. In our experiments, we compute a 2-d latent representation of tracks for the computation of user models. This allows us to inspect the resulting clusters visually during the development of the user models and, more importantly, reduces cluster computation time substantially, which naturally permits better scalability for larger datasets.

4.2 User Models

For modeling user preferences for musical tracks and their characteristics, we naturally require models for both tracks and users as we utilize a user’s model and compare it with track models to find suitable similar tracks that may be recommended to the user.

As for modeling tracks and their characteristics, we rely on their acoustic features (AF; e.g., danceability or tempo). However, for users we require more sophisticated user models, as these have to represent a possibly extensive and diverse set of tracks and their characteristics to eventually represent a user’s musical preferences. We propose user models that are based on clusters of similar tracks and utilize a user’s membership in these clusters (i.e., the fact that user has listened to tracks that belong to a given cluster) to get a fine-grained representation of the many faces of the listening preferences of a given user. For determining such clusters and computing the membership of tracks in these clusters, we experiment with two approaches: (i) we utilize k-means clustering to find tracks that exhibit similar acoustic features and use the characteristics of these clusters to characterize users; and (ii) we apply Gaussian Mixture Models (GMM) [24] as these allow to model a track by the computed probability density function regarding the GMM’s components. Based on a track’s density functions, we derive a set of GMM-based user models. Generally, the idea is that based on these clusters or components, we aim to model a user based on the characteristics of one or multiple of these track clusters.

In the following, we describe the proposed user models to capture the musical preferences of users. An overview of the user models and the features used to characterize users and tracks is shown in Table 2.

Content avg: In a baseline model, we utilize the eight acoustic features of all tracks a user has listened to and compute the average across all tracks of a user for each of the features presented in Section 3. This allows us to describe a user with his/her average listening behavior, breaking a user’s preferences down into eight acoustic features. Please note that in the remainder of this paper, we refer to models as *Content*-models if the representation of the user or a track relies on acoustic features.

Content avg, sd: This model is built upon the Content avg model, which we extend by adding the standard deviation of each of the acoustic features across all tracks of a user. We expect the added SD to mitigate the effects of averaging a large number of features that potentially differ substantially as users may listen to music with highly diverse acoustic characteristics. We again consider this model a baseline that additionally quantifies to which

Model	User Features	Track Feat.
Content avg	user AF avg	AF
Content avg, sd	user AF avg and SD	AF
Content binary k-means	avg. AF of single cluster	AF
Content weighted k-means	weighted avg. AF of clusters	AF
GMM	avg. densities of user's tracks	GMM densities
Content binary GMM	avg. AF of single GMM comp.	AF
Content weighted GMM	weighted avg. AF of GMM comp.	AF
GMM + Content avg, sd	GMM and user AF avg and SD	GMM, AF

Table 2. Overview of evaluated models (AF stands for acoustic features, GMM for Gaussian Mixture Model and SD for the standard deviation).

extent the user's musical preferences vary regarding the acoustic features of his/her listening history.

Content binary k-means: In this model, we rely on the clusters computed by a k-means clustering of all tracks within the dataset in the computed 2-d latent space. In a next step, we attribute each of the tracks a user has listened to a cluster and do a majority vote on the clusters to obtain the cluster that holds most of the user's tracks. We subsequently model a user using the characteristics of the cluster that contains the majority of the user's track. To represent this cluster, we compute the average of the eight acoustic features of all tracks contained in the cluster and add the according standard deviations. Single tracks are represented by its acoustic features. We consider this a rather simple model as we assign the user to a single cluster and hence, limit the model to a single preference scope.

Content weighted k-means: The previous model is limited as it is restricted to a single preference scope. To tackle this problem, we propose the Content weighted k-means model in which we now aim to address multiple sets of preferences of a user. Therefore, we again rely on the k-means clusters, however, we compute a weight for each cluster based on the number of tracks a user has listened to in each cluster. Based on the user's weights for each cluster, we compute a weighted average for each acoustic feature to represent the user, where each cluster is again characterized by its average acoustic features and its standard deviation. Again, in this model each track is represented by its acoustic features.

GMM: In this model, we utilize a Gaussian Mixture Model [24] for representing both the track and the user. Therefore, we compute Gaussian components and represent a track by its probability densities regarding the GMM components. For users, we compute the average probabilities for each component across all of the user's tracks to model a user's musical preferences by using the GMM components. We consider this model a proxy, as it does not directly utilize acoustic features to represent a track, but the probabilistic assignments of a track to a set of groups of tracks (components).

Content binary GMM: In contrast to the pure GMM model, this model relies on content features instead of probability densities to represent a user. Analogously to the Content binary k-means model, we rely on GMM to assign the user's tracks to components. In particular, we assign the tracks found in the user's listening history to

GMM components. In a next step, we select the component with the highest number of user tracks assigned to, where we assign a track to the component with the highest probability density for the track. The user is then modeled by the characteristics of the selected component (again using the average and standard deviation across all acoustic features of the tracks assigned to the component), whereas each track is again represented by its acoustic features.

Content weighted GMM: This model is again analogous to the content weighted k-means model. However, we rely on a GMM to assign a user's tracks to certain a component as described in the previous model. Based on these assignments, we analogously compute the weighted mean and standard deviation for each acoustic feature for each GMM cluster to represent a user and the characteristics of tracks are captured by their acoustic features.

GMM + content avg, sd: In this model, we combine the GMM components baseline model with the content avg, sd baseline model and hence, represent a user by his/her component weights regarding the Gaussian Mixture Model and further add the average and standard deviation across all acoustic features of the user's tracks. Similarly, a track is represented by its GMM densities and its acoustic features.

We also performed experiments on representing users and tracks with cluster or component assignments only and did an analysis of further combinations of the proposed models. However, the results were below the evaluated baselines and hence, we do not list these models here.

5. EXPERIMENTAL SETUP

We model the evaluation of the proposed user models as a recommendation task, where we aim to obtain a ranked list of tracks that are of interest to the user. For this task, we rely on Gradient Boosting Decision Trees. Particularly, we utilize the popular XGBoost system [8], a scalable end-to-end tree boosting approach that has been shown to be highly suited for recommendation tasks [2, 28]. For the training phase of the tree, we set the training objective to be the binary classification error rate (i.e., the number of wrongly classified tracks in relation to all tracks classified, where tracks with a predicted probability of relevance larger than 0.5 are classified as relevant for the given user, and all other tracks are considered irrelevant for the user). Please note that we deliberately chose a classification-based recommendation approach and refrained from utilizing more elaborate recommender approaches such as context-aware matrix factorization [3] or tensor-based factorization approaches [15] as we aim to focus on user modeling aspects in this paper.

For the recommendation task carried out, we require a rating for each track in the dataset to define whether a given track was listened to and thus, considered relevant for a given user. Hence, we add a binary factor *rating* to the processed dataset: for each unique $\langle user, track \rangle$ -combination, the *rating* $r_{i,j}$ is 1 if the user u_i has listened to track t_j . Due to a lack of publicly available data, our dataset does not contain any implicit feedback of users

(i.e., skipping behavior, session durations or dwell times during browsing the catalog). This is why we cannot estimate any preference towards a track a user not listened to as proposed by [14]. Thus, we assume tracks the user has not listened to as negative examples [14] and hence, assign a rating of 0 to these tracks. Even though there is a certain bias towards negative values as some missing values might be positive, Pan et al. [29] found that this method for rating estimation works well. To perform the proposed recommendation task via classification, we require the dataset to also include negative examples. Therefore, for each user, we add random tracks the user did not interact with (i.e., tracks t_j with $r_{i,j} = 0$ for the given user u_i) to the dataset until both the training and test sets are filled with 50% relevant and 50% non-relevant items. We chose to oversample the positive class to avoid class imbalance and hence, a bias towards the negative class.

Using the resulting data set, we train a XGBoost model that performs a binary classification on the relevance of tracks for a given users. We extract the probabilities underlying the classification decision to rank tracks by their probability of relevance in the recommendation task.

To evaluate the performance of the proposed user models in regards to recommendation quality, we perform a per-user evaluation. Therefore, we use each user's listening history and perform a *leave-k-out* evaluation (also known as hold-out evaluation) [6, 10] per user. Based on the dataset that now contains both positive and negative samples for each user, we compute a hold-out set of size k : along the lines of previous research [12, 13], we randomly select 10 positive samples (tracks that the user has listened to) and 100 negative samples (tracks the user has not listened to). These 110 tracks form the test set for each user, whereas the recommender system is trained on the remainder of the dataset. We compute the predicted ratings for the tracks in the test set and rank the track recommendation candidates w.r.t. the probability that the current track belongs to the positive class in descending order. For our experiments, we consider all predicted probabilities > 0.5 as a predicted interaction and thus, we consider these items as relevant, all others as irrelevant and hence, not added to the list of recommendations.⁶

Based on the predicted ratings, we compute *precision*, *recall*, and the F_1 -measure to assess the top-10 accuracy [11]. We evaluate the 10 top ranked tracks as too many track recommendations might provoke choice overload and hence, is not feasible. The problem of choice overload has been addressed by Bollen et al. [5] who state that user satisfaction is highest when presenting the user with Top-5 to Top-20 items—naturally assuming that the recommendation list contains a sufficient number of relevant items for the user. For assessing the overall *precision*, *recall*, and F_1 -measure of the evaluated recommender systems, we compute the measures for each individual user and compute the average among all users. For computing the *recall* measure, all relevant items in the test set are con-

sidered, independent of the number of recommendations. Thus, there is a natural cap for *recall*, namely the number of recommendations divided by the number of relevant items in the test set.

For the tuning of XGBoost parameters, we did a preliminary cross-evaluation aiming to optimize precision values for the proposed models and hence, set the number of maximum trees to learn the models to 2,000. For all other parameters, we relied on the default settings. For the training and tuning of k-means and GMM for the creation of the user models, we performed the following steps. For k-means, estimated the number of clusters by utilizing the elbow method based on the within-cluster sum of squares. For the given dataset, we estimated the number of clusters to be 5. For the GMM, we performed a training phase based on expectation maximization and determined the number of components using the Bayesian Information Criterion (BIC), which resulted in a total of 9 components for the GMM.

6. RESULTS

We present the results of our evaluation for a recommendation list of size ten in Table 3 and in a precision-recall plot depicted in Figure 1.

The best results are obtained by the GMM + Content avg, sd model, reaching a precision@10 of 0.771 and a recall@10 of 0.427 and hence, achieving substantially higher precision and recall scores than any other model. Comparing the results of this model to the GMM model (relying on solely the assignments to GMM components) and the Content avg, sd baseline model shows that those two models individually perform substantially worse than when combined. When inspecting the results of the GMM model, we find that solely relying on the GMM density functions does not suffice to represent a user's musical taste. Particularly, all content-based GMM or k-means models achieve higher performance when applied in isolation. However, combining a simple content-based approach that provides acoustic features regarding the user's general preferences, with GMM, provides us with a representative user model. This suggests that the GMM model captures a user's diverse preferences regarding the detected components and hence, his/her distribution in preference towards specific types of music, while his/her general preferences are captured by the average acoustic features and the according standard deviation.

Model	Prec	Rec	F ₁
GMM + Content avg, sd	0.771	0.427	0.632
Content k-means weighted	0.606	0.316	0.400
Content k-means binary	0.573	0.300	0.383
Content binary GMM	0.569	0.298	0.381
Content weighted GMM	0.569	0.298	0.381
GMM	0.231	0.122	0.226
Content avg, sd	0.161	0.089	0.241
Content avg	0.159	0.087	0.241

Table 3. Precision, Recall and F₁@10, ordered by F₁.

⁶ This distinction between the two classes is also utilized by XGBoost for binary classification tasks based on logistic regression.

Our results also show that the user models based on k-means clusters slightly outperform the methods based on GMM components (1.8% in recall, 3.7% in precision). Please note that for k-means we determined the number of clusters to be five, whereas we created nine GMM components (as described in Section 5). Our findings regarding the number of clusters are also in line with previous analyses on playlists [30], where the authors found that clustering the tracks within playlists into five clusters allows for cohesive and homogeneous clusters.

The weighted k-means approach achieves better results than the binary k-means approach. This seems natural as the former incorporates the user's membership in all clusters, whereas the latter does a majority vote and utilizes the resulting (single) cluster to characterize the user. However, this does not hold for the GMM-based approaches. While the differences between the weighted and binary k-means approaches are marginal, for GMM there is no difference between weighted and binary Content GMM.

The proposed baseline model Content avg achieves the lowest values regarding recall, precision and F_1 . Adding the standard deviation to this model hardly impacts the results. We initially suspected that adding the SD to the model may allow mitigating the effects of aggregating possibly highly different tracks as we aggregate across all tracks of a user (regarding their acoustic features), however, this is not confirmed by our experiments. In preliminary experiments, we also used different representations of clusters: while we now utilize the mean acoustic features and the according SDs, we also used only the mean features. We found that the SD contributes only marginally as the dispersion of tracks in regards to acoustic features is already captured by the individual clusters/components and hence, the tracks contained in a single cluster/component are more homogeneous. We also experimented with models that utilize user-cluster assignments for k-means, however, those models achieved inferior results. In contrast, representing those clusters by the average acoustic features across all contained tracks seems to be representative. Combining k-means cluster assignments with content-based models also lead to inferior results, which we lead back to the fact that the GMM probability densities provide more information than sheer cluster-assignments.

Generally, we conclude that content features strongly contribute to user models and that grouping tracks into clusters (k-means) or components (GMM) and solely relying on the assignment to those clusters or components is not sufficient for a representative user model. Finding groups of similar tracks to represent users by user-group assignments via the tracks a user listened to is not expressive enough. Naturally, utilizing content features allows to compute higher-dimensional similarities between users and their tracks (in our experiments, 8 dimensions) and hence, a more fine-grained notion of similarity.

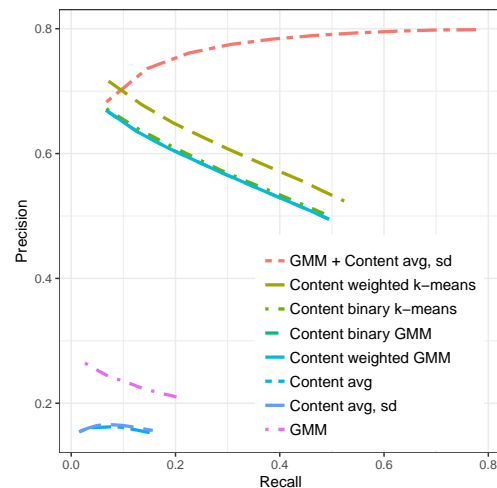


Figure 1. Precision-Recall curves for all models.

7. DISCUSSION

We find that a GMM that captures the specific preferences of a user towards a set of nine types of music (captured by nine GMM components) complemented by the general musical preference of a user (captured by the avg. acoustic features of his/her tracks) provides the best results.

Regarding the limitations of this study, we note that the content descriptors utilized are aggregated high-level features. This allowed us to keep the feature space smaller and to specifically focus on the user modeling aspects. Furthermore, this evaluation is solely based on aspects related to the content of tracks and no further user-related aspects as e.g., proposed by Schedl et al. [34]. Lastly, while the proposed models characterize users based on their interest in different clusters/components and hence, are able to build more specific user models, we still represent each cluster/component by the mean acoustic features of the tracks contained, which naturally limits the user model's specificity. However, we believe that our findings are a valuable contribution to advance user modeling for MIR and MRS and to foster further research in this direction.

8. CONCLUSION AND FUTURE WORK

We proposed and evaluated a set of user models for describing the musical preference of users by leveraging content descriptors of tracks the user has listened to. We find that a GMM complemented by the user's general musical preferences describes a user's different musical preferences best. We believe that our findings can contribute to improved user models for music recommender systems and generally, MIR tasks. In future work, we aim to investigate methods to combine the models evaluated by e.g., ensemble methods. Furthermore, we aim to tackle the problem that our current model still computes average acoustic features across a large number of tracks.

9. REFERENCES

- [1] Jesper Steen Andersen. Using the echo nest’s automatically extracted music features for a musicological purpose. In *2014 4th International Workshop on Cognitive Information Processing (CIP)*, pages 1–6, 2014.
- [2] Takashi Ayaki, Hidekazu Yanagimoto, and Michifumi Yoshioka. Recommendation from access logs with ensemble learning. *Artificial Life and Robotics*, 22(2):163–167, 2017.
- [3] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. Matrix factorization techniques for context aware recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, pages 301–304. ACM, 2011.
- [4] Dmitry Bogdanov, Martín Haro, Ferdinand Fuhrmann, Emilia Gómez, and Perfecto Herrera. Content-based music recommendation based on user preference examples. In *4th ACM Conference on Recommender Systems. Workshop on Music Recommendation and Discovery (Womrad 2010)*, page 33, 2010.
- [5] Dirk Bollen, Bart P. Knijnenburg, Martijn C. Willemssen, and Mark Graus. Understanding choice overload in recommender systems. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 63–70, New York, NY, USA, 2010. ACM.
- [6] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, UAI’98*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.
- [7] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.
- [8] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.
- [9] Parag Chordia, Mark Godfrey, and Alex Rae. Extending content-based recommendation: The case of indian classical music. In *Eight International Society for Music Information Retrieval Conference*, pages 571–576, 2008.
- [10] P. Cremonesi, R. Turrin, E. Lentini, and M. Matteucci. An evaluation methodology for collaborative recommender systems. In *2008 International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, pages 224–231, 2008.
- [11] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 39–46, New York, NY, USA, 2010. ACM.
- [12] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, pages 278–288, 2015.
- [13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182, 2017.
- [14] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. IEEE, 2008.
- [15] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 79–86. ACM, 2010.
- [16] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer US, Boston, MA, 2011.
- [17] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.
- [18] Mark Levy and Mark Sandler. Learning latent semantic models for music from social tags. *Journal of New Music Research*, 37(2):137–150, 2008.
- [19] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764 – 766, 2013.
- [20] Ning-Han Liu. Comparison of content-based music recommendation using different distance estimation methods. *Applied Intelligence*, 38(2):160–174, Mar 2013.
- [21] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the 1st International Symposium on Music Information Retrieval*, volume 270, pages 1–11, 2000.
- [22] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer US, Boston, MA, 2011.
- [23] L. McInnes and J. Healy. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, 2018.

- [24] Geoffrey McLachlan and David Peel. *Finite mixture models*. Wiley Series in Probability and Statistics, 2004.
- [25] Matt McVicar, Tim Freeman, and Tijl De Bie. Mining the correlation between lyrical and audio features and the emergence of mood. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 783–788, 2011.
- [26] Joshua L Moore, Shuo Chen, Thorsten Joachims, and Douglas Turnbull. Learning to embed songs and tags for playlist prediction. In *Twelfth International Society on Music Information Retrieval Conference*, pages 349–354, 2012.
- [27] Joshua L Moore, Thorsten Joachims, and Douglas Turnbull. Taste space versus the world: an embedding analysis of listening habits and geography. In *Fourteenth International Society for Music Information Retrieval Conference*, pages 439–444, 2014.
- [28] Andrzej Pacuk, Piotr Sankowski, Karol Wegrzycki, Adam Witkowski, and Piotr Wygocki. Job recommendations based on preselection of offers and gradient boosting. In *Proceedings of the Recommender Systems Challenge*, pages 10:1–10:4. ACM, 2016.
- [29] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 502–511, 2008.
- [30] Martin Pichl, Eva Zangerle, and Gnther Specht. Understanding playlist creation on music streaming platforms. In *IEEE International Symposium on Multimedia*, pages 475–480. IEEE Computer Society, 2016.
- [31] Markus Schedl. The LFM-1B Dataset for Music Retrieval and Recommendation. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 103–110, New York, NY, USA, 2016. ACM.
- [32] Markus Schedl and Arthur Flexer. Putting the user in the center of music information retrieval. In *Twelfth International Society for Music Information Retrieval Conference*, pages 385–390. Citeseer, 2012.
- [33] Markus Schedl, Arthur Flexer, and Julián Urbano. The neglected user in music information retrieval research. *Journal of Intelligent Information Systems*, 41(3):523–539, 2013.
- [34] Markus Schedl, Peter Knees, and Fabien Gouyon. New paths in music recommender systems research. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 392–393. ACM, 2017.
- [35] B. Shao, D. Wang, T. Li, and M. Ogihara. Music recommendation based on acoustic features and user access patterns. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(8):1602–1611, Nov 2009.
- [36] Xinxi Wang, David Rosenblum, and Ye Wang. Context-aware mobile music recommendation for daily activities. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pages 99–108, New York, NY, USA, 2012. ACM.
- [37] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 627–636, New York, NY, USA, 2014. ACM.
- [38] Kazuyoshi Yoshii, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *Seventh International Society for Music Information Retrieval Conference*, 2006.
- [39] Bingjun Zhang, Jialie Shen, Qiaoliang Xiang, and Ye Wang. Compositemap: A novel framework for music similarity measure. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 403–410, New York, NY, USA, 2009. ACM.

REPRESENTATION LEARNING OF MUSIC USING ARTIST LABELS

Jiyoung Park^{1*} Jongpil Lee^{2*} Jangyeon Park¹ Jung-Woo Ha¹ Juhan Nam²

¹ NAVER Corp.

² Graduate School of Culture Technology, KAIST

{j.y.park, jangyeon.park, jungwoo.ha}@navercorp.com, {richter, juhannam}@kaist.ac.kr

ABSTRACT

In music domain, feature learning has been conducted mainly in two ways: unsupervised learning based on sparse representations or supervised learning by semantic labels such as music genre. However, finding discriminative features in an unsupervised way is challenging and supervised feature learning using semantic labels may involve noisy or expensive annotation. In this paper, we present a supervised feature learning approach using artist labels annotated in every single track as objective meta data. We propose two deep convolutional neural networks (DCNN) to learn the deep artist features. One is a plain DCNN trained with the whole artist labels simultaneously, and the other is a Siamese DCNN trained with a subset of the artist labels based on the artist identity. We apply the trained models to music classification and retrieval tasks in transfer learning settings. The results show that our approach is comparable to previous state-of-the-art methods, indicating that the proposed approach captures general music audio features as much as the models learned with semantic labels. Also, we discuss the advantages and disadvantages of the two models.

1. INTRODUCTION

Representation learning or feature learning has been actively explored in recent years as an alternative to feature engineering [1]. The data-driven approach, particularly using deep neural networks, has been applied to the area of music information retrieval (MIR) as well [14]. In this paper, we propose a novel audio feature learning method using deep convolutional neural networks and artist labels.

Early feature learning approaches are mainly based on unsupervised learning algorithms. Lee et al. used convolutional deep belief network to learn structured acoustic patterns from spectrogram [19]. They showed that the learned features achieve higher performance than Mel-Frequency Cepstral Coefficients (MFCC) in genre and artist classification. Since then, researchers have applied various

unsupervised learning algorithms such as sparse coding [12, 24, 29, 31], K-means [8, 24, 30] and restricted Boltzmann machine [24, 26]. Most of them focused on learning a meaningful dictionary on spectrogram by exploiting sparsity. While these unsupervised learning approaches are promising in that it can exploit abundant unlabeled audio data, most of them are limited to single or dual layers, which are not sufficient to represent complicated feature hierarchy in music.

On the other hand, supervised feature learning has been progressively more explored. An early approach was mapping a single frame of spectrogram to genre or mood labels via pre-trained deep neural networks and using the hidden-unit activations as audio features [11, 27]. More recently, this approach was handled in the context of transfer learning using deep convolutional neural networks (DCNN) [6, 20]. Leveraging large-scaled datasets and recent advances in deep learning, they showed that the hierarchically learned features can be effective for diverse music classification tasks. However, the semantic labels that they use such as genre, mood or other timbre descriptions tend to be noisy as they are sometimes ambiguous to annotate or tagged from the crowd. Also, high-quality annotation by music experts is known to be highly time-consuming and expensive.

Meanwhile, artist labels are the meta data annotated to songs naturally from the album release. They are objective information with no disagreement. Furthermore, considering every artist has his/her own style of music, artist labels may be regarded as terms that describe diverse styles of music. Thus, if we have a model that can discriminate different artists from music, the model can be assumed to explain various characteristics of the music.

In this paper, we verify the hypothesis using two DCNN models that are trained to identify the artist from an audio track. One is the basic DCNN model where the softmax output units corresponds to each of artist. The other is the Siamese DCNN trained with a subset of the artist labels to mitigate the excessive size of the output layer in the plain DCNN when a large-scale dataset is used. After training the two models, we regard them as a feature extractor and apply artist features to three different genre datasets in two experiment settings. First, we directly find similar songs using the artist features and K-nearest neighbors. Second, we conduct transfer learning to further adapter the features to each of the datasets. The results show that proposed approach captures useful features for unseen audio datasets

* Equally contributing authors.



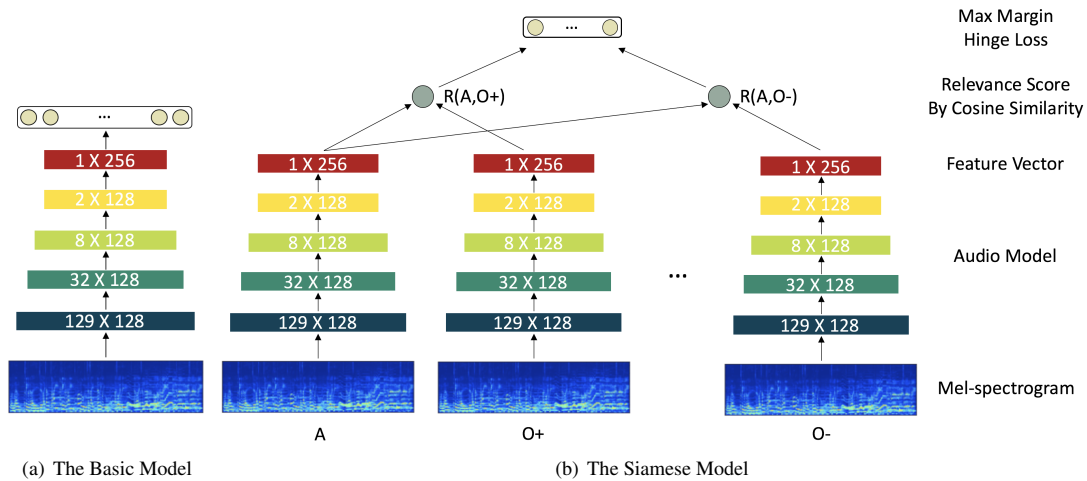


Figure 1. The proposed architectures for the model using artist labels.

and the proposed models are comparable to those trained with semantic labels in performance. In addition, we discuss the advantages and disadvantages of the two proposed DCNN models.

2. LEARNING MODELS

Figure 1 shows the two proposed DCNN models to learn audio features using artist labels. The basic model is trained as a standard classification problem. The Siamese model is trained using pair-wise similarity between an anchor artist and other artists. In this section, we describe them in detail.

2.1 Basic Model

This is a widely used 1D-CNN model for music classification [5, 9, 20, 25]. The model uses mel-spectrogram with 128 bins in the input layer. We configured the DCNN such that one-dimensional convolution layers slide over only a single temporal dimension. The model is composed of 5 convolution and max pooling layers as illustrated in Figure 1(a). Batch normalization [15] and rectified linear unit (ReLU) activation layer are used after every convolution layer. Finally, we used categorical cross entropy loss in the prediction layer.

We train the model to classify artists instead of semantic labels used in many music classification tasks. For example, if the number of artists used is 1,000, this becomes a classification problem that identifies one of the 1,000 artists. After training, the extracted 256-dimensional feature vector in the last hidden layer is used as the final audio feature learned using artist labels. Since this is the representation from which the identity is predicted by the linear softmax classifier, we can regard it as the highest-level artist feature.

2.2 Siamese Model

While the basic model is simple to train, it has two main limitations. One is that the output layer can be excessively large if the dataset has numerous artists. For example, if a

dataset has 10,000 artists and the last hidden layer size is 100, the number of parameters to learn in the last weight matrix will reach 1M. Second, whenever new artists are added to the dataset, the model must be trained again entirely. We solve the limitations using the Siamese DCNN model.

A Siamese neural network consists of twin networks that share weights and configuration. It then provides unique inputs to the network and optimizes similarity scores [3, 18, 22]. This architecture can be extended to use both positive and negative examples at one optimization step. It is set up to take three examples: anchor item (query song), positive item (relevant song to the query) and negative item (different song to the query). This model is often called *triplet networks* and has been successfully applied to music metric learning when the relative similarity scores of song triplets are available [21]. This model can be further extended to use several negative samples instead of just one negative in the triplet network. This technique is called *negative sampling* and has been popularly used in word embedding [23] and latent semantic model [13]. By using this technique, they could effectively approximate the full softmax function when the output class is extremely large (i.e. 10,000 classes).

We approximate the full softmax output in the basic model with the Siamese neural networks using negative sampling technique. Regarding the artist labels, we set up the negative sampling by treating identical artist's song to the anchor song as positive sample and other artists' songs as negative samples. This method is illustrated in Figure 1(b). Following [13], the relevance score between the anchor song feature and other song feature is measured as:

$$R(A, O) = \cos(y_A, y_O) = \frac{y_A^T y_O}{|y_A| |y_O|} \quad (1)$$

where y_A and y_O are the feature vectors of the anchor song and other song, respectively.

Meanwhile, the choice of loss function is important in this setting. We tested two loss functions. One is the softmax function with categorical cross-entropy loss to max-

imize the positive relationships. The other is the max-margin hinge loss to set only margins between positive and negative examples [10]. In our preliminary experiments, the Siamese model with negative sampling was successfully trained only with the max-margin loss function between the two objectives, which is defined as follows:

$$\text{loss}(A, O) = \sum_{O^-} \max[0, \Delta - R(A, O^+) + R(A, O^-)] \quad (2)$$

where Δ is the margin, O^+ and O^- denotes positive example and negative examples, respectively. We also grid-searched the number of negative samples and the margin, and finally set the number of negative samples to 4 and the margin value Δ to 0.4. The shared audio model used in this approach is exactly the same configuration as the basic model.

2.3 Compared Model

In order to verify the usefulness of the artist labels and the presented models, we constructed another model that has the same architecture as the basic model but using semantic tags. In this model, the output layer size corresponds to the number of the tag labels. Hereafter, we categorize all of them into *artist-label model* and *tag-label model*, and compare the performance.

3. EXPERIMENTS

In this section, we describe source datasets to train the two artist-label models and one tag-label model. We also introduce target datasets for evaluating the three models. Finally, the training details are explained.

3.1 Source Tasks

All models are trained with the Million Song Dataset (MSD) [2] along with 30-second 7digital¹ preview clips. Artist labels are naturally annotated onto every song, thus we simply used them. For the tag label, we used the Last.fm dataset augmented on MSD. This dataset contains tag annotation that matches the ID of the MSD.

3.1.1 Artist-label Model

The number of songs that belongs to each artist may be extremely skewed and this can make fair comparison among the three models difficult. Thus, we selected 20 songs for each artist evenly and filtered out the artists who have less than this. Also, we configured several sets of the artist lists to see the effect of the number of artists on the model performances (500, 1,000, 2,000, 5,000 and 10,000 artists). We then divided them into 15, 3 and 2 songs for training, validation and testing, respectively for the sets contain less than 10,000 artists. For the 10,000 artist sets, we partitioned them in 17, 1 and 2 songs because once the artists reach 10,000, the validation set already become 10,000 songs even when we only use 1 song from each artist which is already sufficient for validating the model performance.

¹ <https://www.7digital.com/>

We also should note that the testing set is actually not used in the whole experiments in this paper because we used the source dataset only for training the models to use them as feature extractors. The reason we filtered and split the data in this way is for future work².

3.1.2 Tag-label Model

We used 5,000 artists set as a baseline experiment setting. This contains total 90,000 songs in the training and validation set with a split of 75,000 and 15,000. We thus constructed the same size set for tagging dataset to compare the artist-label models and the tag-label model. The tags and songs are first filtered in the same way as the previous works [4, 20]. Among the list with the filtered top 50 used tags, we randomly selected 90,000 songs and split them into the same size as the 5,000 artist set.

3.2 Target Tasks

We used 3 different datasets for genre classification.

- GTZAN (fault-filtered version) [17, 28]: 930 songs, 10 genres. We used a “fault-filtered” version of GTZAN [17] where the dataset was divided to prevent artist repetition in training/validation/test sets.
- FMA small [7]: 8,000 songs, 8 balanced genres.
- NAVER Music³ dataset with only Korean artists: 8,000 songs, 8 balanced genres. We filtered songs with only have one genre to clarify the genre characteristic.

3.3 Training Details

For the preprocessing, we computed the spectrogram using 1024 samples for FFT with a Hanning window, 512 samples for hop size and 22050 Hz as sampling rate. We then converted it to mel-spectrogram with 128 bins along with a log magnitude compression.

We chose 3 seconds as a context window of the DCNN input after a set of experiments to find an optimal length that works well in music classification task. Out of the 30-second long audio, we randomly extracted the context size audio and put them into the networks as a single example. The input normalization was performed by dividing standard deviation after subtracting mean value across the training data.

We optimized the loss using stochastic gradient descent with 0.9 Nesterov momentum with $1e^{-6}$ learning rate decay. Dropout 0.5 is applied to the output of the last activation layer for all the models. We reduce the learning rate when a valid loss has stopped decreasing with the initial learning rate 0.015 for the basic models (both artist-label and tag-label) and 0.1 for the Siamese model. Zero-padding is applied to each convolution layer to maintain its size.

² All the data splits of the source tasks are available at the link for reproducible research <https://github.com/jiyoungpark527/msd-artist-split>.

³ <http://music.naver.com>

Our system was implemented in Python 2.7, Keras 2.1.1 and Tensorflow-gpu 1.4.0 for the back-end of Keras. We used NVIDIA Tesla M40 GPU machines for training our models. Code and models are available at the link for reproducible research⁴.

4. FEATURE EVALUATION

We apply the learned audio features to genre classification as a target task in two different approaches: feature similarity-based retrieval and transfer learning. In this section, we describe feature extraction and feature evaluation methods.

4.1 Feature Extraction Using the DCNN Models

In this work, the models are evaluated in three song-level genre classification tasks. Thus, we divided 30-second audio clip into 10 segments to match up with the model input size and the 256-dimension features from the last hidden layer are averaged into a single song-level feature vector and used for the following tasks. For the tasks that require song-to-song distances, cosine similarity is used to match up with the Siamese model's relevance score.

4.2 Feature Similarity-based Song Retrieval

We first evaluated the models using mean average precision (MAP) considering genre labels as relevant items. After obtaining a ranked list for each song based on cosine similarity, we measured the MAP as following:

$$AP = \frac{\sum_{k \in rel} precision_k}{number\ of\ relevant\ items} \quad (3)$$

$$MAP = \frac{\sum_{q=1}^Q AP(q)}{Q} \quad (4)$$

where Q is the number of queries. $precision_k$ measures the fraction of correct items among first k retrieved list.

The purpose of this experiment is to directly verify how similar feature vectors with the same genre are in the learned feature space.

4.3 Transfer Learning

We classified audio examples using the k-nearest neighbors (k-NN) classifier and linear softmax classifier. The evaluation metric for this experiment is classification accuracy. We first classified audio examples using k-NN to classify the input audio into the largest number of genres among k nearest to features from the training set. The number of k is set to 20 in this experiment. This method can be regarded as a similarity-based classification. We also classified audio using a linear softmax classifier. The purpose of this experiment is to verify how much the audio features of unseen datasets are linearly separable in the learned feature space.

⁴<https://github.com/jongpillee/ismir2018-artist>.

MAP	Artist-label Basic Model	Artist-label Siamese Model	Tag-label Model
GTZAN (fault-filtered)	0.4968	0.5510	0.5508
FMA small	0.2441	0.3203	0.3019
NAVER Korean	0.3152	0.3577	0.3576

Table 1. MAP results on feature similarity-based retrieval.

KNN	Artist-label Basic Model	Artist-label Siamese Model	Tag-label Model
GTZAN (fault-filtered)	0.6655	0.6966	0.6759
FMA small	0.5269	0.5732	0.5332
NAVER Korean	0.6671	0.6393	0.6898

Table 2. KNN similarity-based classification accuracy.

Linear Softmax	Artist-label Basic Model	Artist-label Siamese Model	Tag-label Model
GTZAN (fault-filtered)	0.6721	0.6993	0.7072
FMA small	0.5791	0.5483	0.5641
NAVER Korean	0.6696	0.6623	0.6755

Table 3. Classification accuracy of a linear softmax.

5. RESULTS AND DISCUSSION

5.1 Tag-label Model vs. Artist-label Model

We first compare the artist-label models to the tag-label model when they are trained with the same dataset size (90,000 songs). The results are shown in Table 1, 2 and 3. In feature similarity-based retrieval using MAP (Table 1), the artist-based Siamese model outperforms the rest on all target datasets. In the genre classification tasks (Table 2 and 3), Tag-label model works slightly better than the rest on some datasets and the trend becomes stronger in the classification using the linear softmax. Considering that the source task in the tag-based model (trained with the Last.fm tags) contains genre labels mainly, this result may attribute to the similarity of labels in both source and target tasks. Therefore, we can draw two conclusions from this experiment. First, the artist-label model is more effective in similarity-based tasks (1 and 2) when it is trained with the proposed Siamese networks, and thus it may be more useful for music retrieval. Second, the semantic-based model is more effective in genre or other semantic label tasks and thus it may be more useful for human-friendly music content organization.

5.2 Basic Model vs. Siamese Model

Now we focus on the comparison of the two artist-label models. From Table 1, 2 and 3, we can see that the Siamese model generally outperforms the basic model. However, the difference become attenuated in classification tasks and the Siamese model is even worse on some datasets. Among them, it is notable that the Siamese model is significantly worse than the basic model on the NAVER Music dataset

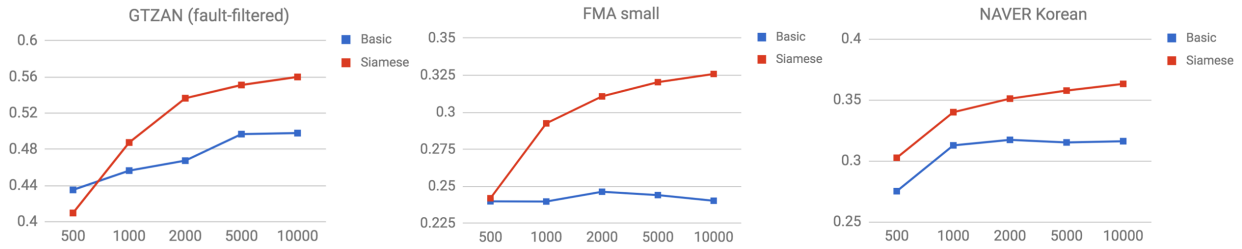


Figure 2. MAP results with regard to different number of artists in the feature models.

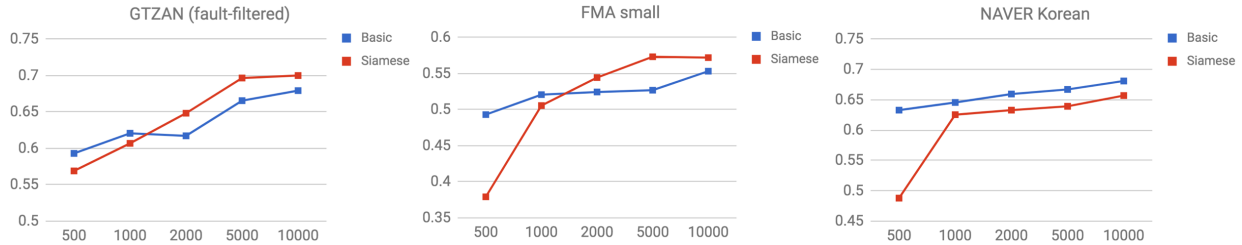


Figure 3. Genre classification accuracy using k-NN with regard to different number of artists in the feature models.

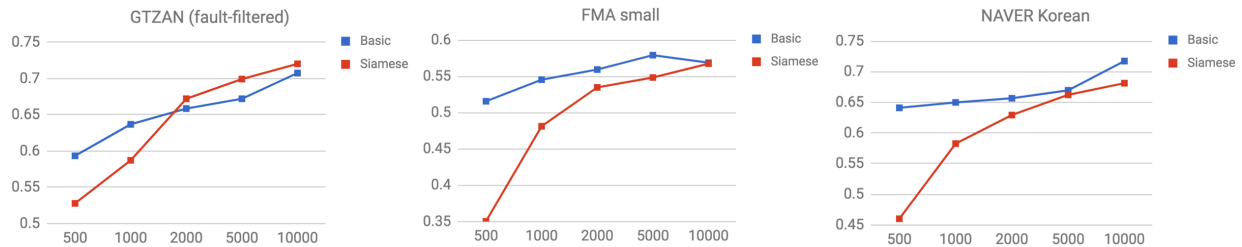


Figure 4. Genre classification accuracy using linear softmax with regard to different number of artists in the feature models.

in the genre classification using k-NN even though they are based on feature similarity. We dissected the result to see whether it is related to the cultural difference between the training data (MSD, mostly Western) and the target data (the NAVER set, only Korean). Figure 5 shows the detailed classification accuracy for each genre of the NAVER dataset. In three genres, ‘Trot’, ‘K-pop Ballad’ and ‘Kids’ that do not exist in the training dataset, we can see that the basic model outperforms the Siamese model whereas the results are opposite in the other genres. This indicates that the basic model is more robust to unseen genres of music. On the other hand, the Siamese model slightly over-fits to the training set, although it effectively captures the artist features.

5.3 Effect of the Number of Artists

We further analyze the artist-label models by investigating how the number of artists in training the DCNN affects the performance. Figure 2, 3 and 4 are the results that show similarity-based retrieval (MAP) and genre classification (accuracy) using k-NN and linear softmax, respectively, according to the increasing number of training artists. They show that the performance is generally proportional to the number of artists but the trends are quite different between the two models. In the similarity-based retrieval, the MAP of the Siamese model is significantly higher than that of the basic model when the number of

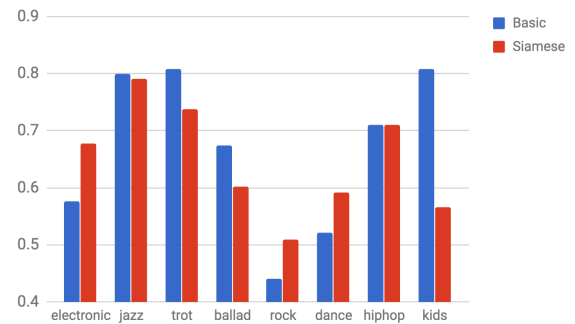


Figure 5. The classification results of each genre for the NAVER dataset with only Korean music.

artists is greater than 1,000. Also, as the number of artists increases, the MAP of the Siamese model consistently goes up with a slight lower speed whereas that of the basic model saturates at 2,000 or 5,000 artists. On the other hand, the performance gap changes in the two classification tasks. On the GTZAN dataset, while the basic model is better for 500 and 1,000 artists, the Siamese model reverses it for 2,000 and more artists. On the NAVER dataset, the basic model is consistently better. On the FMA small, the results are mixed in two classifiers. Again, the results may be explained by our interpretation of the models in Section 5.2. In summary, the Siamese model seems

Models	GTZAN (fault-filtered)	FMA small
2-D CNN [17]	0.6320	-
Temporal features [16]	0.6590	-
Multi-level Multi-scale [20]	0.7200	-
SVM [7]	-	0.5482 [†]
Artist-label Basic model	0.7076	0.5687
Artist-label Siamese model	0.7203	0.5673

Table 4. Comparison with previous state-of-the-art models: classification accuracy results. Linear softmax classifier is used and features are extracted from the artist-label models trained with 10,000 artists. [†] This result was obtained using the provided code and dataset in [7].

to work better in similarity-based tasks and the basic model is more robust to different genres of music. In addition, the Siamese model is more capable of being trained with a large number of artists.

5.4 Comparison with State-of-the-arts

The effectiveness of artist labels is also supported by comparison with previous state-of-the-art models in Table 4. For this result, we report two artist-label models trained with 10,000 artists using linear softmax classifier. In this table, we can see that the proposed models are comparable to the previous state-of-the-art methods.

6. VISUALIZATION

We visualize the extracted feature to provide better insight on the discriminative power of learned features using artist labels. We used the DCNN trained to classify 5,000 artists as a feature extractor. After collecting the feature vectors, we embedded them into 2-dimensional vectors using t-distributed stochastic neighbor embedding (t-SNE).

For artist visualization, we collect a subset of MSD (apart from the training data for the DCNN) from well-known artists. Figure 6 shows that artists' songs are appropriately distributed based on genre, vocal style and gender. For example, artists with similar genre of music are closely located and female pop singers are close to each other except Maria Callas who is a classical opera singer. Interestingly, some songs by Michael Jackson are close to female vocals because of his distinctive high-pitched tone.

Figure 7 shows the visualization of features extracted from the GTZAN dataset. Even though the DCNN was trained to discriminate artist labels, they are well clustered by genre. Also, we can observe that some genres such as disco, rock and hip-hop are divided into two or more groups that might belong to different sub-genres.

7. CONCLUSION AND FUTURE WORK

In this work, we presented the models to learn audio feature representation using artist labels instead of semantic labels. We compared two artist-label models and one tag-label model. The first is a basic DCNN consisting of a softmax output layer to predict which artist they belong to out of all artists used. The second is a Siamese-style architecture that maximizes the relative similarity score be-

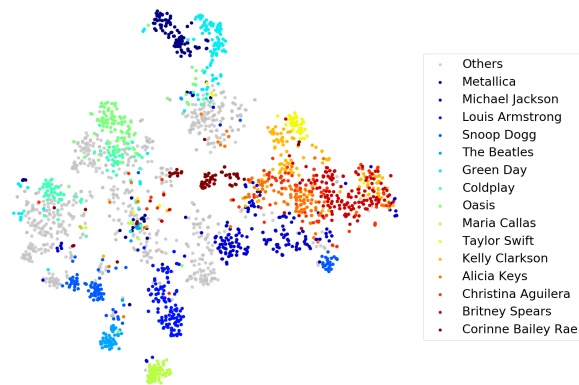


Figure 6. Feature visualization by artist. Total 22 artists are used and, among them, 15 artists are represented in color.

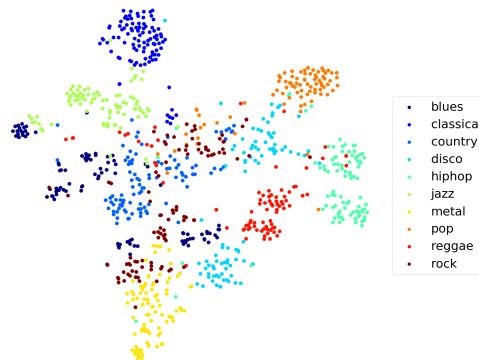


Figure 7. Feature visualization by genre. Total 10 genres from the GTZAN dataset are used.

tween a small subset of the artist labels based on the artist identity. The last is a model optimized using tag labels with the same architecture as the first model. After the models are trained, we used them as feature extractors and validated the models on song retrieval and genre classification tasks on three different datasets. Three interesting results were found during the experiments. First, the artist-label models, particularly the Siamese model, is comparable to or outperform the tag-label model. This indicates that the cost-free artist-label is as effective as the expensive and possibly noisy tag-label. Second, the Siamese model showed the best performances on song retrieval task in all datasets tested. This can indicate that the pair-wise relevance score loss in the Siamese model helps the feature similarity-based search. Third, the use of a large number of artists increases the model performance. This result is also useful because the artists can be easily increased to a very large number.

As future work, we will investigate the artist-label Siamese model more thoroughly. First, we plan to investigate advanced audio model architecture and diverse loss and pair-wise relevance score functions. Second, the model can easily be re-trained using new added artists because the model does not have fixed output layer. This property will be evaluated using cross-cultural data or using extremely small data (i.e. one-shot learning [18]).

8. ACKNOWLEDGEMENT

This work was supported by Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Science, ICT & Future Planning (2015R1C1A1A02036962) and by NAVER Corp.

9. REFERENCES

- [1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- [2] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, volume 2, pages 591–596, 2011.
- [3] Jane Bromley, Isabelle Guyon, Yann LeCun, Edward Säckinger, and Roopak Shah. Signature verification using a “siamese” time delay neural network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 737–744, 1994.
- [4] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 805–811, 2016.
- [5] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2392–2396, 2017.
- [6] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Transfer learning for music classification and regression tasks. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 141–149, 2017.
- [7] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 316–323, 2017.
- [8] Sander Dieleman and Benjamin Schrauwen. Multi-scale approaches to music audio feature learning. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 116–121, 2013.
- [9] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968, 2014.
- [10] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems (NIPS)*, pages 2121–2129, 2013.
- [11] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 339–344, 2010.
- [12] Mikael Henaff, Kevin Jarrett, Koray Kavukcuoglu, and Yann LeCun. Unsupervised learning of sparse features for scalable audio classification. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 681–686, 2011.
- [13] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proc. of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM, 2013.
- [14] Eric Humphrey, Juan Bello, and Yann LeCun. Feature learning and deep architectures: new directions for music informatics. *Journal of Intelligent Information Systems*, 41(3):461–481, Dec 2013.
- [15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.
- [16] Il-Young Jeong and Kyogu Lee. Learning temporal features using a deep neural network and its application to music genre classification. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 434–440, 2016.
- [17] Corey Kereliuk, Bob L Sturm, and Jan Larsen. Deep learning and music adversaries. *IEEE Transactions on Multimedia*, 17(11):2059–2071, 2015.
- [18] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.
- [19] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems (NIPS)*, pages 1096–1104, 2009.
- [20] Jongpil Lee and Juhan Nam. Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging. *IEEE Signal Processing Letters*, 24(8):1208–1212, 2017.
- [21] Rui Lu, Kailun Wu, Zhiyao Duan, and Changshui Zhang. Deep ranking: Triplet matchnet for music metric learning. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–125, 2017.

- [22] Pranay Manocha, Rohan Badlani, Anurag Kumar, Ankit Shah, Benjamin Elizalde, and Bhiksha Raj. Content-based representations of audio using siamese neural networks. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.
- [23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NIPS)*, pages 3111–3119, 2013.
- [24] Juhan Nam, Jorge Herrera, Malcolm Slaney, and Julius O. Smith. Learning sparse feature representations for music annotation and retrieval. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 565–570, 2012.
- [25] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *Proc. of the International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6, 2016.
- [26] Jan Schlüter and Christian Osendorfer. Music Similarity Estimation with the Mean-Covariance Restricted Boltzmann Machine. In *Proc. of the International Conference on Machine Learning and Applications*, pages 118–123, 2011.
- [27] Erik M. Schmidt and Youngmoo E. Kim. Learning emotion-based acoustic features with deep belief networks. In *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 65–68, 2011.
- [28] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.
- [29] Yonatan Vaizman, Brian McFee, and Gert Lanckriet. Codebook-based audio feature representation for music information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(10):1483–1493, 2014.
- [30] Jan Wülfing and Martin Riedmiller. Unsupervised learning of local features for music classification. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 139–144, 2012.
- [31] Chin-Chia Yeh, Li Su, and Yi-Hsuan Yang. Dual-layer bag-of-frames model for music genre classification. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 246–250, 2013.

StructureNet: INDUCING STRUCTURE IN GENERATED MELODIES

Gabriele Medeot¹
Samer Abdallah¹

Srikanth Cherla¹
Marco Selvi¹

Katerina Kosta¹
Ed Newton-Rex¹

Matt McVicar¹
Kevin Webster²

¹Jukedeck Ltd., London, United Kingdom

²Imperial College London, London, United Kingdom

{gabriele, srikanth, katerina, matt, samer, marco, ed}@jukedeck.com

kevin.webster@imperial.ac.uk

ABSTRACT

We present the StructureNet - a recurrent neural network for inducing structure in machine-generated compositions. This model resides in a musical structure space and works in tandem with a probabilistic music generation model as a modifying agent. It favourably biases the probabilities of those notes that result in the occurrence of structural elements it has learnt from a dataset. It is extremely flexible in that it is able to work with any such probabilistic model, it works well when training data is limited, and the types of structure it can be made to induce are highly customisable. We demonstrate through our experiments on a subset of the Nottingham dataset that melodies generated by a recurrent neural network based melody model are indeed more structured in the presence of the StructureNet.

1. INTRODUCTION

Automated generation of symbolic music using computers involves the application of computer algorithms to the creation of novel musical scores. The natural predisposition of computers to quickly enumerate and choose from a large set of compositional alternatives makes them suitable candidates for discovering novelty in the vast space of musical possibilities that could be daunting to a human composer. Leveraging computing power for this purpose has the potential to aid and accelerate the creative process, thus lowering the bar for composition and democratising it. So-called *machine-generated music* has been a subject of steady interest since the pioneering work of a few musically inclined information theorists [5, 8]. This interest has surged during the past decade or so within academia and especially outside it with the rise of certain industry players (such as Jukedeck¹ and the Magenta project²).

¹<https://www.jukedeck.com/>

²<https://magenta.tensorflow.org/>

The roughly seven decade-long history of machine-generated symbolic music has seen the application of a plethora of algorithms to varying degrees of success [8]. With the increasing digitisation of musical scores, those relying on machine learning have gained importance in recent times. The relatively successful approaches among these have been Probabilistic Grammars [9], (Hidden) Markov models [19, 21], and Connectionist architectures [2, 18]. The latter in particular have proven to be highly effective at representing musical information and modelling long-term dependencies which are crucial to generating good-quality music [3].

This paper addresses the issue of long-term structure in machine-generated symbolic monophonic music. Structure is a key aspect of music composed by humans that plays a crucial role in giving a piece of music a sense of overall coherence and intentionality. It appears in a piece as a collection of musical patterns, variations of these patterns, literal or motivic repeats and transformations of sections of music that have occurred earlier in the same piece. Hampshire underlines that a piece can be conceived as a work of art if and only if the listener's mind is actively tracing the structure of the work using her own natural imagery and musical memory [7, p. 16].

Here we introduce StructureNet - a recurrent neural network that induces structure in machine-generated melodies. It learns about structure from a dataset consisting of structural elements and their occurrence statistics, which is created using a structure-tagging algorithm from an existing dataset of melodies. Once trained, StructureNet works in tandem with a melody model which generates a probability distribution over a set of musical notes. Given the melody model's prediction at any given time during generation, StructureNet uses the structural elements implied by the melody so far to alter the prediction, leading to a more structured melody in the future. Our experiments reveal that music generated with StructureNet contains significantly better structure, even when it is trained on a relatively small dataset. We provide musical examples that highlight this fact.

The next section introduces relevant state-of-the-art. Some preliminaries and a description of StructureNet follow in Sections 3 and 4 respectively. Based on the results presented in Section 5, we summarise our findings and suggest potential future work in Section 6.



© Gabriele Medeot, Srikanth Cherla, Katerina Kosta, Matt McVicar, Samer Abdallah, Marco Selvi, Ed Newton-Rex, Kevin Webster. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Gabriele Medeot, Srikanth Cherla, Katerina Kosta, Matt McVicar, Samer Abdallah, Marco Selvi, Ed Newton-Rex, Kevin Webster. "StructureNet: Inducing Structure in Generated Melodies", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

2. RELATED WORK

In order to repeat verbatim or with variations sections that have occurred previously in a piece of machine-generated music, i.e. to induce structure in it, the model must be able to encode and recall in some way what has happened in the past. This can be achieved in a variety of ways. In a first instance, improving structure simply involves making the generation model more powerful. An example of this is the RNN-RBM [2] that was enhanced purely by replacing its components - the Recurrent Neural Network (RNN) by a Long-Short Term Memory (LSTM) Network to improve its temporal memory, making it the LSTM-RTRBM [16], and the Restricted Boltzmann Machine (RBM) by a Deep Belief Network (DBN) to improve its output layer, making it the RNN-DBN [10]. Similarly, it was demonstrated in [4] that connectionist models outperform Markov models in modelling melodic sequences. Closely related to these is a musically informed improvement that enriches the feature encoding to include those features that have the potential to add more information about structure [6, 19]. Along similar lines, the Magenta Project proposed two neural network architectures to model higher-level structure in music - the Lookback RNN and Attention RNN [25]. While the former augments the model's feature vector with information about notes from previous measures, repeat information and metrical location, the latter adopts an attention-based mechanism [1] wherein a weighted sum of the model's outputs in the previous n locations is used in addition to its current state to make better predictions. Such approaches address the overall quality of music, of which high-level structure is just one aspect. Moreover, the improvements afforded by the former kind are highly dependent on the training loss, which does not explicitly take into account structure of the kind observed in music. So while an improvement in the model or feature representation does tend to improve the overall quality of music in a piece, improvement is often observed over short time-spans and not necessarily in the higher-level structure.

Alternatively, one can explicitly address the issue of high-level structure in machine-generated compositions. One simple solution involves dividing the generation task between multiple models. The MELONET system [13], whose goal is to produce variations of a given melodic theme, achieves structural coherence by dividing the effort between two mutually interacting neural networks operating at different time-scales. The first network learns to recognise musical structure while the second network predicts the musical notes. Similarly, Todd [24] proposed two cascaded networks that allow the explicit representation of structure in a hierarchy. The first network generates a sequence of plans which correspond to descriptions of melodic chunks, and the second a sequence of notes given a plan. More recently, Roig et al. [22] devised a system in which melodic and rhythmic patterns existing in the dataset are concatenated according to statistically governed rules to form new patterns that are not too distant from those occurring in the dataset. In the system known as MorpheuS [11] music generation is formulated

as a combinatorial optimisation problem in which a template of musical structure acts as a hard-constraint, and solved using a meta-heuristic search algorithm known as Variable Neighbourhood Search. Patterns contained in the dataset of pieces are discovered using an existing pattern-detection algorithm [17]. In a similar vein, [20] control the generation of chord sequences and melodies using steerable constraints Markov chains. Lattner et al. [14] adopt a similar approach where a Convolutional Restricted Boltzmann Machine is combined with a constraint optimisation technique to constrain the music sampled from the C-RBM according to the musical structure of a given template.

3. BACKGROUND

StructureNet is a Recurrent Neural Network (RNN) that operates in the space of musical structure and learns sequences of features that denote the presence or absence of repeats at a point in time and their type, if present. Here we give an overview of the Long Short-Term Memory (LSTM) RNN that underlies StructureNet and the definition of structural repeats that we rely on.

3.1 Long Short-Term Memory

The RNN is a type of neural network for modelling sequences and its basic architecture consists of an input layer, a hidden layer and an output layer. The *state* of its hidden layer acts as a memory of the past information it encounters while traversing a sequence. At each location in the sequence, the RNN makes use of both the input and the state of its hidden layer from the previous location to predict an output. Here we use a special case of the RNN known as the Long-Short Term Memory (LSTM) network [12] that, owing to the presence of purpose-built *memory cells* to augment its hidden layer, boasts a greater temporal memory than the standard RNN. Given an input vector x_t at sequence location t , the output of the LSTM h_{t-1} and its memory cell c_{t-1} (collectively, its state) from the previous location, the output of the LSTM layer h_t is computed and further propagated into another layer of a larger model.

3.2 Modelling Melodies and Structure Elements

The output layer of the note-based (as opposed to frame-based) melody model in the present work contains two groups of softmax units. Each group of softmax units models a single probability distribution over a set of mutually exclusive possibilities. The first of these denotes the musical pitch of the note, and the second its duration. Given the output of the LSTM layer h_t at any given location t in the sequence, this is transformed into two independent probability distributions \mathbf{p}_t and \mathbf{d}_t that together make up the output layer of the network. From these two distributions, the probability of a certain note (with pitch and duration) can be obtained simply by multiplying the probabilities of its corresponding pitch and duration respectively. Note that the output layer of StructureNet contains three groups of softmax units. Although these represent different quantities that define aspects of structure (explained in detail in



Figure 1. A 16-measure melody generated by our LSTM melody model together with the StructureNet. A selection of repeats in this melody are as follows: measures 9-12 are a duration-interval repeat of measures 5-8, as are measures 13-14 of measures 9-10; and measures 15 and 16 are both duration repeats of measure 12.

Sections 4.1 and 4.2), the manner in which these are combined to generate the probabilities of structural elements is identical to the melody model. Also note that the choice of the LSTM as the melody model is arbitrary and it can be replaced by any other probabilistic prediction model.

3.3 A Definition of Structure

There are various types of structure present in music. Composers use techniques such as instrumental variation, changes and repeats in timbre, and dynamics to induce a feeling of familiarity in the listener. In the present work, however, we focus on the score-level repeat information. In a score, perhaps the two most obvious types of repeat are of (1) duration (rhythmic), and (2) pitches (melodic).

A duration repeat is a section of the melody, the durations of whose notes are the same as those of a previous section. Examples of duration repeats can be found in the melody of Figure 1. These are determined purely by the sequences of crotchets and quavers contained in these measures. When it comes to pitch, it is helpful to think of these repeats in terms of *intervals rather than absolute pitch*. The interval between two notes can be defined in a number of ways, but in this work we use the scale degree distance between notes. For instance, in the key of C major, the scale degree between a C note and subsequent E note would be the same as the scale degree between a D note and subsequent F note. Given this definition of an interval, a duration-interval repeat is a section of the melody that holds the same relationship to a previous section as a duration repeat, and additionally the intervals between whose consecutive notes are the same as those between the consecutive notes of that previous section. Figure 1 also illustrates duration-interval repeats. In the present work, we consider duration repeats as well as repeats of both durations and intervals. Purely interval repeats were found to be very few in our chosen dataset and were thus ignored.

4. StructureNet

StructureNet is only able to produce structural repeat information that biases the predictions of an accompanying music (in the present case melody) model. In Section 4.2

we will outline a methodology whereby it modifies the probability of notes that the melody model produces, thus encouraging structure but not enforcing it. Crucially, this means that the structure network is able to *suggest* repeats of certain types, but if the melody network assigns very low probability to notes that would form these repeats, it is free to “override” the structure network’s suggestions in a probabilistic and flexible manner. The specifics of how StructureNet achieves these goals is outlined in the remainder of this section, beginning with the type of structure we capture and how we identify it.

4.1 A Dataset of Structure

StructureNet operates in a space of musical structure. In order to train the model, we first create this structure dataset by processing a dataset of melodies with a musical repeat-detection algorithm. The algorithm encodes each melody into a sequence of binary feature vectors in the semi-quaver temporal resolution (although this resolution is not a strict requirement: if the dataset contains no notes shorter than a quaver, one may use a quaver as the minimal resolution). The feature vector itself is a concatenation of three one-hot sub-vectors. The first is given by

$$[f, d, di_{tr}, di_{nt}]$$

wherein each bit of the first sub-vector indicates which of four categories a given frame of music belongs to. These are (1) f - free music, (2) d - duration repeat, (3) di_{tr} - duration-interval repeat with transposition and (4) di_{nt} - duration-interval repeat without transposition. The only distinction between the two types of duration-interval repeats is that in the case of di_{tr} the section to which the frame belongs is a transposed version of the original section whereas in the case of di_{nt} the section to which the frame belongs is at the same musical pitch as the original section. The free music bit f indicates that the frame is a part of a section that is not a repeat of any previous section of the melody. The second one-hot sub-vector is given by

$$[f, l_{0.5}, l_{0.75}, l_{1.0}, l_{1.5}, l_{2.0}, l_{3.0}, l_{4.0}, l_{8.0}, l_{16.0}]$$

and contains bits that indicate the *lookback*, i.e. the distance (in crotchets) between the original section and the section containing the current frame, if the section containing the current frame is a repeat of the original section. If it is not a repeat, the free music bit f is on. Note that the value of the free music bits in both sub-vectors is identical and hence uses the same notation. Also note that the choice of the set of lookbacks is completely open to change, and highlights another flexible aspect of the model; it may even be possible to learn the optimal set of lookbacks for a given dataset. Finally, the third 8-dimensional one-hot vector ϕ encodes the location of a frame via its *beat strength* β and its *measure strength* ρ . The beat strength [15] encodes the strength of each metrical location in a measure. In a measure divided into 16 semi-quaver beats (as in the present work), its values are $\beta = [0, 4, 3, 4, 2, 4, 3, 4, 1, 4, 3, 4, 2, 4, 3, 4]$. The measure

strength extends the notion of beat strength to a sequence of measures. The strengths associated with beats in a measure are associated with measures in a piece, beginning with the first measure. In the present work, we choose a cycle of 8 measures that correspond to the following sequence of measure strengths $\rho = [0, 3, 2, 3, 1, 3, 2, 3]$. In both cases, a lower value indicates a higher strength. Our encoding is defined as

$$\phi_t = \begin{cases} \rho(\text{mod}(t, 8)) & \text{if } \text{mod}(t, 16) = 0 \\ \max(\rho) + \beta(\text{mod}(t, 16)) & \text{otherwise} \end{cases} \quad (1)$$

Note that just as the beat strength, the measure cycle duration for the measure length can also be varied as desired.

StructureNet models the vector that is a concatenation of these three sub-vectors as three groups of softmax units in its output layer. As noted earlier in Section 3.2, the manner in which one combines the probability distributions represented by these two groups of softmax units (for instance, a duration-interval repeat of lookback 8.0, or an interval repeat of lookback 1.5) is by multiplying the corresponding probabilities one from each group.

The repeat-detection algorithm works by first converting a sequence of notes into two strings - one corresponding to durations and the other to intervals. In each of these strings, it then uses a string matching algorithm to find substrings that repeat. Single-note repeats are trivial and thus discarded, and only those repeats that correspond to the above listed lookbacks are retained. Any note that is longer than 2 measures is split into multiple notes of the same pitch to limit the number of characters required to represent the piece as a string. Then the list of duration repeats are filtered such that only the longest repeats remain and all overlapping and shorter repeats are discarded. At this stage, the duration-interval repeats are nothing but duration repeats with coinciding interval repeats. So from the list of interval repeats only those are retained that coincide exactly with the current list of duration repeats with the same lookbacks. These are tagged as duration-interval repeats, replacing the corresponding duration repeats to give the final list of duration repeats and duration-interval repeats. While it is indeed possible to look for other types of repeats, we limit ourselves in this paper to the above as it is sufficient to demonstrate the efficacy of StructureNet. This also highlights the flexibility of the model wherein one may change the type of repeats detected and also customise the number of lookbacks as needed.

4.2 Influencing Event Probabilities

Once trained on the above described structure dataset, StructureNet is then put to use with the probabilistic melody prediction model \mathcal{M}_m . At time t (that is, given the history of notes generated up to time t), the model \mathcal{M}_m predicts a probability distribution P_t over a set of notes N . At the same time, given the history of repeats generated so far, the structure model \mathcal{M}_s predicts a probability distribution Q_t over a set of possible repeats Π , which includes an element π_f , representing ‘free music’. Each note $\nu \in N$

can be consistent with a subset Π_t^ν of these repeats, which will always include π_f , meaning that every note is consistent with ‘free music’.

StructureNet influences the prediction P_t by modifying the probability of each note according to the probabilities of the repeats with which it is consistent. Let $\phi_t : N \times \Pi \rightarrow \{0, 1\}$ be a function such that $\phi_t(\nu, \pi) = 1$ when note ν is consistent with repeat π at time t and 0 otherwise. In terms of this we can express Π_t^ν as $\{\pi \in \Pi | \phi_t(\nu, \pi) = 1\}$, and further define $N_t^\pi = \{\nu \in N | \phi_t(\nu, \pi) = 1\}$, which is the set of notes consistent with π . Each note ν is then assigned a weight

$$W_t(\nu) = P_t(\nu) \sum_{\pi \in \Pi_t^\nu} \frac{Q_t(\pi)}{\mu_t^\pi}, \quad (2)$$

where $\mu_t^\pi = \sum_{\nu \in N_t^\pi} P_t(\nu)$. In this way, the relative probability of a note ν is increased when it is consistent with repeat(s) to which \mathcal{M}_s has assigned high probability.

It is important to note that \mathcal{M}_m and \mathcal{M}_s operate at different temporal resolutions—note-level and semiquaver frame-level respectively—and that this difference becomes significant here. Suppose note ν is of duration $\Delta_\nu = \tau_\nu \delta$, where δ is the frame duration and τ_ν is the number of frames occupied by ν . Ideally, in order to get an accurate estimate of the joint probability of the note ν and the repeat π , one should consider the probability that \mathcal{M}_s assigns to τ_ν consecutive frames of π . This would be expressed as

$$W_t(\nu) = P_t(\nu) \sum_{\pi \in \Pi_t^\nu} \prod_{k=0}^{\tau_\nu-1} \frac{Q_{t+k}(\pi)}{\mu_{t+k}^\pi}. \quad (3)$$

However, we found in our experiments that the single-step approximation (2) works well in practice and is less computationally intensive than (3).

Next, the weight distribution W_t is normalised to obtain a probability distribution R_t :

$$R_t(\nu) = \frac{W_t(\nu)}{\sum_{\nu \in N} W_t(\nu)}. \quad (4)$$

We may now sample a note ν_t from this distribution and update the internal state of the melodic model \mathcal{M}_m with this observation.

It remains to update the state of the structure model \mathcal{M}_s with some observed repeat. The note ν_t sampled at time t could be associated with any of the repeats that were consistent with it. We choose one by sampling π_t from a distribution S_t over $\Pi_t^{\nu_t}$ defined as

$$S_t(\pi) = \frac{Q_t(\pi)}{\sum_{\pi' \in \Pi_t^{\nu_t}} Q_t(\pi')}. \quad (5)$$

At this point the two models are misaligned due to the different time-scales they operate in, with \mathcal{M}_m being τ semiquaver frames ahead of \mathcal{M}_s . Since each update of the state of \mathcal{M}_s takes it ahead by just one semiquaver frame, it is necessary to update \mathcal{M}_s τ times repeatedly with the same structure vector so that it is once again aligned with \mathcal{M}_m .

At the end of the process described above, we have a melody note sampled from our melody model that has been

influenced by StructureNet. StructureNet has also updated its own state according to the sampled note and is ready to influence the choice of next note.

5. EXPERIMENTS

We demonstrate the efficacy of StructureNet on a well-known dataset of melodies by comparing statistics over several musical quantities computed both on the dataset and compositions generated by the melody model alone and the melody and structure models combined. The results show that the presence of StructureNet leads to music that is more structured and closer in the statistics to the dataset. We also share the generated music to allow the reader to her or himself be the judge of our claims.

5.1 Dataset

We evaluate StructureNet on the cleaned Nottingham folk melody dataset that was released by the Jukedeck Research Team [23]. This publicly available dataset facilitates reproducibility. We carry out our experiments on the subset of 450 4/4 time-signature pieces out of the 1,548 contained in it. Each piece of the dataset was truncated to its first 16 measures and transposed into the Key of C, and all upbeats at the beginning of each piece were removed prior to training. We used 20% (90 segments) of the data as the validation and the rest (360 segments) for training the models. StructureNet is also trained on the same dataset following the application of the repeat-tagging algorithm. However, one must note that this is not a requirement and a different dataset may be used for learning structure and could potentially lead to interesting results. StructureNet successfully induces structure in the generated melodies despite the few examples contained in the training data.

5.2 Training methodology

As mentioned earlier, both the structure network and the melody network are LSTMs and contain a single hidden layer. A Bayesian Optimisation based method was employed to carry out model selection. In the case of the melody model, the single best outcome of the grid search was used. As for StructureNet, ten models with the same best set of hyperparameters as determined by the model selection step, and different initial conditions, were trained and used in tandem with the melody model. This was done in order to be able to compute confidence intervals in the figures. The hidden layer size of each network was varied between 50 and 1000 in steps of 50 during model selection, which led to $n_{hid}^s = 950$ in the former and $n_{hid}^m = 250$ in the latter. Early stopping was used as a regulariser, such that the training was stopped and the best models thus far retrieved after no improvement in the validation cost for 25 epochs. The models were trained using the ADAM optimiser with an initial learning rate $\eta_{init} = 0.001$, and parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

5.3 Evaluation

Our hypothesis is two-fold: (1) that repeat-related statistics computed over the melodies generated with StructureNet are closer to those over the dataset melodies than those over melodies generated without StructureNet, and (2) that non repeat-related statistics do not differ between the melodies generated by the melody model with and without StructureNet. This would demonstrate that the use of StructureNet leads to more structured melodies than are generated by the melody model on its own, and that are musically at least as similar to the original data as the melody model alone achieves. The statistics are:

1. **Repeat Count:** Number of repeats corresponding to various lookback values (in crotchets).
2. **Repeat Duration:** Number of repeats of various durations (in crotchets).
3. **Repeat Onsets:** Number of repeats beginning at various locations (in crotchets) in a piece.
4. **Pitch, start time and duration distributions:** Occurrence statistics of pitches, start times in measure, and durations.

The first three are repeat-related statistics and the rest are not. A histogram of each is first computed per collection of melodies (dataset, generations with and without StructureNet), and then normalised by the count of melodies in the collection to generate a probability distribution (as the counts vary between the different collections of melodies). The KL-Divergences (KLD) $\kappa_{data,SN}$ and $\kappa_{data,NoSN}$ between the distribution pairs (dataset, StructureNet) and (dataset, No StructureNet) respectively highlight the effect of introducing StructureNet (Table 1). Ideally, among the structure-related distributions, we would want $\kappa_{data,SN} < \kappa_{data,NoSN}$. And among the non-repeat-related distributions, we wish for $\kappa_{data,SN} \leq \kappa_{data,NoSN}$.

	$\kappa_{data,NoSN}$	$\kappa_{data,SN}$
Repeat Count (D)	0.0356 \pm 0.0022	0.0069 \pm 0.0043
Repeat Duration (D)	0.1071 \pm 0.0047	0.0389 \pm 0.0168
Repeat Onset (D)	0.0844 \pm 0.0038	0.0357 \pm 0.0094
Repeat Count (DI)	0.0511 \pm 0.0049	0.0173 \pm 0.0095
Repeat Duration (DI)	0.2402 \pm 0.0069	0.0634 \pm 0.0352
Repeat Onset (DI)	0.1209 \pm 0.0073	0.0639 \pm 0.0194
Repeat Count (all)	0.0483 \pm 0.0035	0.0083 \pm 0.0045
Repeat Duration (all)	0.0996 \pm 0.0033	0.025 \pm 0.0081
Repeat Onset (all)	0.0875 \pm 0.0036	0.031 \pm 0.0103
Pitch	0.0079 \pm 0.0011	0.0061 \pm 0.0012
Duration	0.0049 \pm 0.0016	0.0042 \pm 0.0014
Onset	0.058 \pm 0.0081	0.0275 \pm 0.0082

Table 1. KL-divergences between the training data and melodies generated with and without StructureNet (computed over 10 sets of 450 melodies generated with each trained StructureNet) for the Duration (D), Duration-Interval (DI) and all repeat types.

5.4 Observations

In Table 1, the KLD values show a greater match between the dataset and the set of generated melodies in the presence of StructureNet than in its absence. This holds true for both duration and duration-interval repeats. Figure 2 illustrates such similarities (over all repeat types) visually. One will see here that overall StructureNet (a) is conducive to the creation of longer repeats while generally having a positive effect on shorter ones as well, (b) is conducive to the creation of repeats that have lookback values similar to those in the dataset, particularly larger lookbacks (encouraging distant repeats), and (c) encourages repeats to begin on those metrical locations in a generated piece where they tend to occur in the dataset.

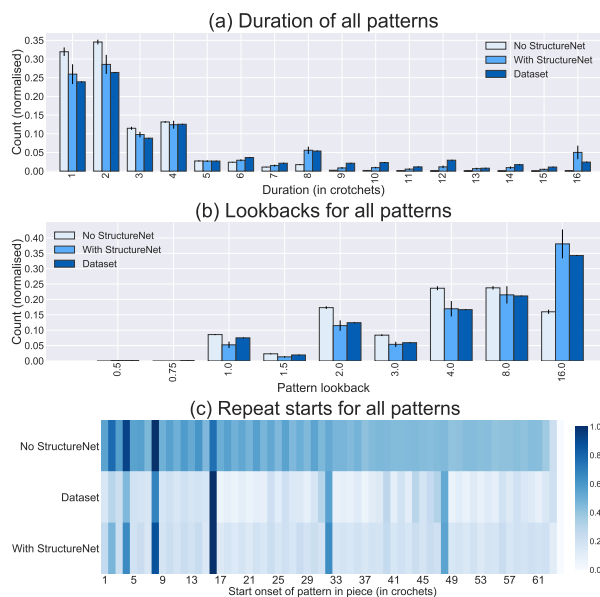


Figure 2. Repeat-related statistics of the dataset to the two generation modes (with/without StructureNet).

It is also evident from the set of three non-repeat-related statistics of Figure 3 that the presence of StructureNet has, more often than not, led to a better match of the generated melody statistics to the dataset. This is also supported by the very similar KLD values (often lower in the $\kappa_{data, NoSN}$ column) for these three musical quantities at the bottom of Table 1. And finally, each plot in Figure 4 shows the percentage of generated melodies with various degrees of free music in them. The three plots together reveal that using StructureNet reduces the proportion of free music (and thus increases the proportion of repeats) in the generated melodies in a way that more closely matches the proportions of free music and repeats in the dataset. Note that the statistics in Figures 2, 3 and 4 have been computed over the same number of melodies (of the same duration in measures). We have made a representative subset of melodies generated with and without StructureNet in the MIDI format available for scrutiny³.

³ <https://goo.gl/hL9RhZ>

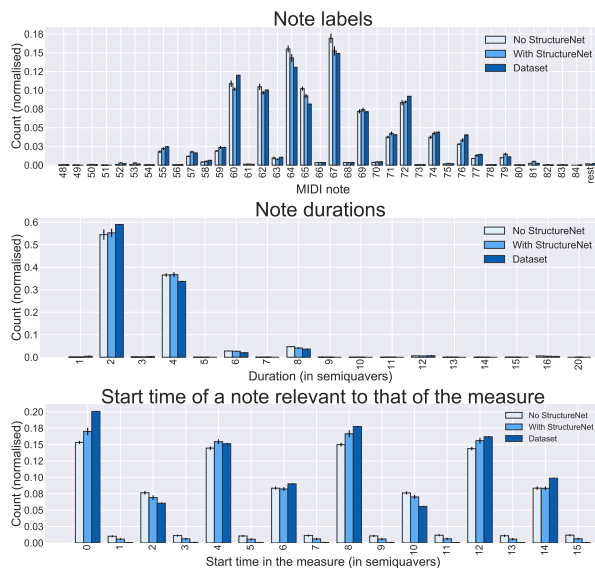


Figure 3. Non repeat-related statistics of the dataset to the two generation modes (with/without StructureNet).

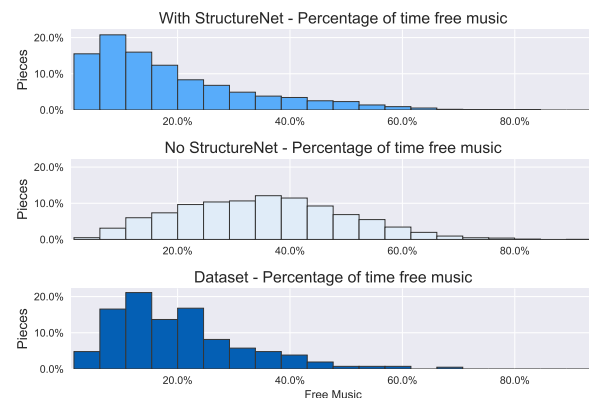


Figure 4. The amount of generated melodies with various degrees of free music in them.

6. CONCLUSIONS & FUTURE WORK

We introduced StructureNet - an RNN that influences the predictions of a melody model so as to give the generated melodies greater structure. We demonstrated using statistics, as well as several musical examples, that this model does indeed increase the probability of encountering longer and more distant (greater lookback) patterns in music generated by a melody model. Given these initially successful results, we foresee some interesting directions for future work. Firstly, we are interested in experimenting with a more evolved pattern detection algorithm such as SIATEC and COSIATEC [17]. This will lead to new feature representations over and beyond just repeats that can perhaps provide a better insight into musical structure to StructureNet. We would like to expand the three musical quantities introduced in Section 5.3 into a more comprehensive set of quantities that can lead to a more thorough evaluation of musical structure.

7. REFERENCES

- [1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [2] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: application to polyphonic music generation and transcription. In *Intl. Conf. on Machine Learning*, pages 1881–1888. Omnipress, 2012.
- [3] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation—a survey. *arXiv preprint arXiv:1709.01620*, 2017.
- [4] Srikanth Cherla, Son N Tran, Artur d’Avila Garcez, and Tillman Weyde. Discriminative learning and inference in the recurrent temporal rbm for melody modelling. In *Intl. Joint Conf. on Neural Networks*, pages 1–8. IEEE, 2015.
- [5] Joel E Cohen. Information theory and music. *Systems Research and Behavioral Science*, 7(2):137–163, 1962.
- [6] Darrell Conklin and Ian H Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.
- [7] Nicholas Cook. *Music, imagination, and culture*. Oxford University Press, 1992.
- [8] Jose David Fernandez and Francisco Vico. AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48:513–582, 2013.
- [9] Jon Gillick, Kevin Tang, and Robert M Keller. Machine learning of jazz grammars. *Computer Music Journal*, 34(3):56–66, 2010.
- [10] Kratarth Goel, Raunaq Vohra, and JK Sahoo. Polyphonic music generation by modeling temporal dependencies using a rnn-dbn. In *Intl. Conf. on Artificial Neural Networks*, pages 217–224. Springer, 2014.
- [11] Dorien Herremans and Elaine Chew. Morpheus: generating structured music with constrained patterns and tension. *IEEE Trans. on Affective Computing*, 2017.
- [12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [13] Dominik Hörnel and Wolfram Menzel. Learning musical structure and style with neural networks. *Computer Music Journal*, 22(4):44–62, 1998.
- [14] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints. *arXiv preprint arXiv:1612.04742*, 2016.
- [15] Fred Lerdahl and Ray S Jackendoff. *A generative theory of tonal music*. MIT press, 1985.
- [16] Qi Lyu, Zhiyong Wu, and Jun Zhu. Polyphonic music modelling with lstm-rtrbm. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 991–994. ACM, 2015.
- [17] David Meredith. Cosatec and siateccompress: Pattern discovery by geometric compression. In *Intl. Society for Music Information Retrieval Conf. Intl. Society for Music Information Retrieval*, 2013.
- [18] Michael C Mozer. Connectionist music composition based on melodic, stylistic and psychophysical constraints. *Music and connectionism*, pages 195–211, 1991.
- [19] Francois Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, 2003.
- [20] François Pachet and Pierre Roy. Markov constraints: steerable generation of markov sequences. *Constraints*, 16(2):148–172, 2011.
- [21] Jean-Francois Paiement, Yves Grandvalet, and Samy Bengio. Predictive models for music. *Connection Science*, 21(2-3):253–272, 2009.
- [22] Carles Roig, Lorenzo J Tardón, Isabel Barbancho, and Ana M Barbancho. Automatic melody composition based on a probabilistic model of music style and harmonic rules. *Knowledge-Based Systems*, 71:419–434, 2014.
- [23] Jukedeck R&D Team. “Releasing a cleaned version of the Nottingham Dataset.” Web blog post. *Jukedeck Research*, 7 Mar. 2017. Web. 30 Mar. 2018.
- [24] Peter M Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43, 1989.
- [25] Elliot Waite. “Generating Long-Term Structure in Songs and Stories.” Web blog post. *Magenta*, 15 Jul. 2016. Web. 30 Mar. 2018.

SUMMARIZING AND COMPARING MUSIC DATA AND ITS APPLICATION ON COVER SONG IDENTIFICATION

Diego Furtado Silva

Departamento de Computação
Universidade Federal de São Carlos
São Carlos, Brazil
diegofs@ufscar.br

Felipe Vieira Falcão, Nazareno Andrade

Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Brazil
{felipev,nazareno}@lisd.ufcg.edu.br

ABSTRACT

While there is a multitude of music information retrieval algorithms that have distance functions as their core procedure, comparing the similarity between recordings is a costly procedure. At the same, the recent growth of digital music repositories makes necessary the development of novel time- and memory-efficient algorithms to deal with music data. One particularly interesting idea on the literature is transforming the music data into reduced representations, improving the memory usage and reducing the time necessary to assess the similarity. However, these techniques usually add other issues, such as an expensive preprocessing or a reduced retrieval performance. In this paper, we propose a novel method to summarize a recording in small snippets based on its self-similarity information. Besides, we present a simple way to compare other recordings to these summaries. We demonstrate, in the scenario of cover song identification, that our method is more than one order of magnitude faster than state-of-the-art adversaries, at the same time that the retrieval performance is not affected significantly. Additionally, our method is incremental, which allows the easy and fast update of the database when a new song needs to be inserted into the retrieval system.

1. INTRODUCTION

With the arising of digital music platforms and the consequent growth of music data repositories, we have witnessed an increasing interest in fast methods for mining this kind of data. Organizing, searching, and finding patterns in large repositories require algorithms that are efficient in memory and time while providing an accurate performance.

Several algorithms proposed for different music mining and information retrieval rely on comparing (dis)similarities among the recordings of interest. One is-

sue regarding this approach is the scalability of these methods, since comparing distances is usually a costly procedure.

The cover song identification (CSI) is one task that usually is assessed by similarity-based methods. Most work on advancing the knowledge in CSI is based on creating or adapting new similarity measures and algorithms for comparing the recordings [8, 11, 19] or fusing features or distances to improve the retrieval performance [6, 17, 25]. While some efforts point to the direction of improving CSI runtime [4, 10, 18, 24], the majority of papers on CSI rely on quadratic algorithms to compare each pair of songs [3, 21, 26–28], which may difficult its application on large databases.

One particular idea on speeding up the CSI was presented by Silva, Souza and Batista [24]. The authors proposed a training phase to find what they called “triplets”, which are three short excerpts of each original recording that *summarize* them, i.e., that represent the songs with a reduced amount of data. When comparing a new query against these summarized data, the runtime for the distance calculations drastically reduces, since it is proportional to the length of the feature vectors under comparison.

While summarizing tracks can significantly improve retrieval runtime, the triplets technique has some contrivances that make its use difficult. Most importantly, its training phase is costly in time and memory. Also, it considers that more than one original or authorized version of each song is available: the method depends on measuring the distance of each candidate excerpt to several other segments of the same “class label.” Finally, once a new recording is added to the dataset, the training phase must be recomputed, since the method to choose the summaries also relies on comparing songs from different labels to analyze the class separability.

In this work, we also leverage the idea of summarizing recordings for fast retrieval. However, we proposed a new suite of methods for summarizing and comparing music data that makes these two usually costly steps simpler and faster. The methods put forward are based on a fast subsequence similarity join algorithm that achieves good retrieval performances and can easily and quickly increment the reference dataset when a new original recording is presented.



© Diego Furtado Silva, Felipe Vieira Falcão, Nazareno Andrade. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Diego Furtado Silva, Felipe Vieira Falcão, Nazareno Andrade. “Summarizing and Comparing Music Data and Its Application on Cover Song Identification”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

The suit of methods proposed in this work has the following contributions:

- considerably higher speed to summarize recordings compared to the state-of-the-art;
- a reduction of an order of magnitude in the runtime of the comparison and retrieval phase compared to recent proposals of scalable algorithms;
- no significant loss of retrieval performance is incurred in process of speeding up summarization; and
- because our summarization method solely relies on the recording being summarized, the method is naturally parallelizable and incremental.

Figure 1 illustrates the pipeline of our method.

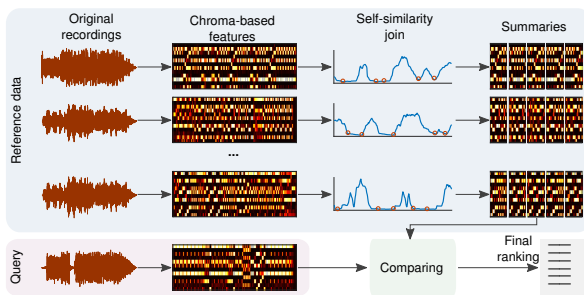


Figure 1: The pipeline of our method consists of summarizing the reference dataset using similarity joins and, for each query, comparing it to the summaries to achieve a ranking by similarity.

This paper is organized as follow: Section 2 introduces a background on the task of cover song recognition and a few related work. Section 3 presents our summarizing and comparing techniques, which composes the proposed suite for fast similarity recover. Section 4 presents the experimental evaluation of our method. Section 5 discuss some ideas on how further improve our proposal. Finally, Section 6 concludes this work.

2. BACKGROUND AND RELATED WORK

The main focus of this paper is the cover song identification (CSI) task. A cover song is a generic name to refer for any recording that is a new version of an original recording. While it may represent an attempt to make a faithfully reproduction of the original work, covers usually widely differs on many characteristics, such as key, timbre, structure and tempo, which makes CSI a difficult task.

To deal with this variation, several methods provide invariance to these issues to CSI algorithms. One example is the Optimal Transposition Index (OTI) [20], which provides key invariance. This algorithm starts by calculating and storing a global pitch profile of the recordings under comparison. Using these profiles, it estimates the difference in key of the songs and transpose one of their feature vectors so that the tracks have the same (estimated) key.

Tempo differences also motivate the need for invariance. Several similarity methods for CSI proposed in the literature are based on a dynamic programming algorithm to dynamically align the compared recordings [6, 8, 21, 25]. This kind of algorithm provides invariance to tempo at the cost of relying on a costly alignment algorithm.

For this reason, techniques which are chiefly concerned with the runtime of CSI systems usually apply lock-step measures such as the Euclidean distance. In these cases, providing tempo invariance in the feature level is a common approach. One option for that is smoothing the feature vectors, an approach that is adopted by some chroma-based features definitions, such as the Chroma Energy Normalized Statistics (CENS) [16].

Many methods for CSI in the literature, if not all, use a pipeline that includes techniques to provide invariances and a distance measure calculation. One example is the already mentioned Triplets [24]. Specifically, this method uses CENS and OTI in its process.

Triplets summarizes the CENS from each reference (original) recording in three short excerpts that are maximally close to excerpts (subsequences) from the same song and far from the excerpts from other pieces. Once a query is presented to the CSI system, it rotates the query according OTI and compares it to the summaries. On the one hand, the summarization significantly improves the runtime to assess a query. On the other hand, the step of finding the triplets is prohibitive thanks to the high number of distance calculations it requires.

Another work from the same research group proposes to identify covers assessing subsequence similarity joins by using the similarity matrix profile, or SiMPle [23]. The SiMPle is a representation of the subsequence similarity join, which is the task of finding the nearest neighbor of each subsequence from a frame-level feature vector among all the subsequences of another vector. Particularly, this operation is called AB-join. The operation of calculating of the best match between a subsequence of a song to itself (disregarding trivial matches) is referred to as self-join.

The join operation returns two pieces of information: the SiMPle and the SiMPle index. While the SiMPle stores the distance of each subsequence to its nearest neighbor, the SiMPle index indicate which subsequence is such neighbor.

The main intuition behind SiMPle in the CSI domain is that comparing a query to its corresponding original version tends to return small distances. Conversely, comparing the query to a recording of another song tends to produce high subsequence distances. As such, the authors defined the distance between a query B and a candidate original recording A as:

$$dist(A, B) = median(SiMPle(B, A)) \quad (1)$$

While in the SiMPle paper the SiMPle-based CSI is performed over the AB-join operation, the authors demonstrate other applications relying on the self-join procedure. For instance, it is possible to use the SiMPle and the SiMPle index to find music thumbnails (most repeated subse-

quence, given by the neighbors stored in the index) and patterns that are faithfully reproduced in different times of the song or are the most different excerpts in the recording (small and high values in the SiMPle, respectively).

However, computing the distance between a high number of subsequence pairs is a costly operation. To speed up the SiMPle calculation, the authors used MASS, a Fast Fourier Transform-based algorithm to perform a fast subsequence similarity search under the Euclidean distance [14]. The Euclidean distance (ED) between two vectors of n elements is calculated as:

$$ED(A, B) = \sqrt{\sum_{i=1}^n (a_i)^2 + \sum_{i=1}^n (b_i)^2 - 2(A \cdot B)} \quad (2)$$

When using the ED to find the best match between a subsequence and a long feature vector, we may slide the short sequence along the longer one. The main idea behind MASS is substituting the required dot product by a FFT-based algorithm for calculating the cross-correlation between the compared vectors, often referred to as sliding dot-product. Besides, we can pre-calculate the quadratic sums required by the ED and reuse it when necessary.

Consider n the length of the long feature vector and m the length of the assessed subsequence. The main advantage of using MASS is that it reduces the subsequence similarity search's complexity from $\mathcal{O}(nm)$ to $\mathcal{O}(n \log n)$ by using the FFT to find the windowed dot-products instead a brute-force approach [29]. Moreover, these dot-products can be reused to calculate SiMPle even faster [22].

In this paper, we propose a new method that sums up the advantages of Triplets and SiMPle in a single solution. Our algorithm is described in the next section.

3. SUMMARIZING AND COMPARING RECORDINGS

We propose the Summarizing and Comparing (SuCo) method. As the name suggests, it is split into two main procedures. The first one summarizes the reference recordings based on SiMPle. To ensure the time efficiency of our method, we use the faster version of SiMPle's algorithm [22]. The second part refers to the way that a query is compared to these summaries to estimate a distance value between the query and each reference recording.

3.1 Summarization

Summarizing music files is not a novel procedure. Although a few algorithm use it as a intermediate step (e.g. Triplets for CSI), it is usually the final procedure in some specific tasks, such as thumbnailing [1]. In this work, we use the SiMPle to summarize music data as the first step of our algorithm. Using this representation of subsequences similarities, we summarize the music files in five excerpts¹ using two different approaches, which are described in the next sub-sections.

¹ The number of summaries per song is a parameter, which we set to 5. For details, please refer to Section 4.2.

The summarization step can be seen as a training phase, similar to what is done by Triplets. However, while summarizing in Triplets depends on comparing each recording with the entire dataset, our approach processes each recording independently. This implies that (i) summarization in SuCo is naturally parallelizable, and (ii) once a new original recording is added, it is only necessary to summarize it and add this summary to our set of summaries. The latter operation takes only hundredths of a second.

3.1.1 Thumbnailing

Thumbnailing relies on summarizing a recording in one short segment that best represents it. A thumbnail enables for example a listener to quickly identify a song or its marked characteristics.

A possible definition of a good thumbnail is the excerpt of the song that is most times repeated [1]. Based on this definition, Silva et al. [22] use the subsequence that most appears in the SiMPle index as the thumbnail of a track. This is the subsequence that is most times considered the nearest neighbor of other fragments. In practical terms, the thumbnail is the mode of the SiMPle index. In case of a tie, the subsequence chosen is the one with lowest mean distance to the segments that point at it in the SiMPle index.

In SuCo we use this same step as our first summary, and combine it with four other in a set of five segments that summarizes each recording. The extra segments are considered that is desirable to extract summaries that faithfully describe the song but they need to be diverse. In other words, we avoid describing music with similar excerpts, as this aggregates little information to the retrieval step.

That said, after we choose a summary, we exclude from the choices of next summaries all the subsequences that have it as the nearest neighbor. From a practical standpoint, we keep the count of times that each subsequence is denoted as the nearest neighbor in the SiMPle index and use this information to decide the next summary. When we select a subsequence as a summary, we turn to zero the count regarding each of the subsequences that point at the current summary in the SiMPle index.

Similarly, we make subsequences around each picked summary also ineligible for next summaries. Let p be the position of the current thumbnail and w a constant defined as one-quarter of the assessed subsequences' length. We turn to zero the neighbor count for all subsequences starting at $p_i \in [p - w, p + w]$.

3.1.2 Diverse repeated pattern

While the thumbnail-based summaries rely on the SiMPle index, we also propose a summarization method based on the distances stored by SiMPle. We count on the fact that small distances mean faithfully repeated patterns. These excerpts are very likely to be more precisely repeated because they are more significant to describe the song. For instance, a guitar solo is not similar to other points of the song. Also, that is not a good summary for the cover song recognition, since many covers skip, modify or poorly perform these parts of the song.

As in the thumbnail version, this summarization process also aims to pick diverse patterns. For this, we use a technique based on that proposed by Dau and Keogh [7]. The main idea of this process is to, after picking a subsequence as a summary, increase the value in the SiMPle of the subsequences that are similar to the current summary. This procedure reduces the chance that we choose similar summaries to describe a song.

More precisely, after choosing a summary, we calculate the distance from it to all the subsequences in the song, using MASS. This provides us a vector of distance which has the same length than SiMPle. Then, we normalize this vector in the interval $[0, 1]$, being that the position storing 1 represents the most distant subsequence and 0 appears at the position of the current summary. Finally, we perform a point-by-point division of the SiMPle by this vector. As similar subsequences have a (normalized) distance close to zero, the division will make its relative positions in SiMPle significantly increase its values. Consequently, they will unlikely be chosen as a summary in the next iterations. We refer the reader to the paper that first proposed this procedure [7] for a formal definition of it.

In addition to the described procedure, we also make ineligible the subsequences that are around the picked summaries. Similarly to the thumbnail, we set a region $p_i \in [p - w, p + w]$ of ineligible subsequences. The difference here is that we set as infinite the values at these positions in the SiMPle.

3.1.3 Pitch profiles

In addition to the summaries, the global pitch class profile are also stored in our procedure. This profile is the normalized sum of each bin of the chroma vectors that describe the recording [20]. This is necessary to apply OTI and, consequently, provide invariance to key differences when calculating the distances for a new query recording.

3.2 Distance Calculation

When a new query is presented to the CSI system, SuCo must compare it to each song in the reference database and return a ranking by similarity. In our proposal, we match the query with each summary of each original recording. For this, we again take advantage of the algorithm MASS.

Given a query q , the steps to compare q with the original recordings are:

1. Calculate the global pitch profile of q and the statistics required by MASS, i.e., its sliding quadratic sums and its FFT (which is used to calculate the sliding dot-product). We only need to calculate these values once and, then, use them in every posterior distance calculations.
2. From an original recording r , compare its pitch profile with the profile obtained from q . Then, rotate the chroma vector of each summary of r accordingly.
3. Using the values calculate in the previous steps, calculate the distances between q and each summary of

r . Store the lowest distance value, i.e., the distance between the summary and the subsequence from q that best matches it.

4. The final distance between q and r is given by the geometric mean of the distances stored in the previous step. The geometric mean benefits low values in its calculation, favoring the match between q and the reference songs with one or more summaries with a good approximate match.

4. EXPERIMENTAL EVALUATION

This section presents an experimental evaluation of the proposed methods. For the sake of reproducibility, all code and detailed results are provided in a supplementary website².

This section is split in distinct topics regarding different phases of our evaluation. First, we describe the datasets we used. Next, we present the experimental setup regarding feature extraction, parameter setting, evaluation measures, and adversary methods. Finally, we present the results of experiments regarding time and retrieval performances.

For simplicity, we refer to our summarizing and comparing methods by thumbnails and diverse repeated patterns as SuCo-thumb and SuCo-repeat, respectively.

4.1 Datasets

The datasets used in our experiments include popular and classical music with different sizes. We opted to use the same data as in the paper that proposed SiMPle, so that results are directly comparable. The datasets are:

- **YouTubeCovers:** This dataset is composed of 50 popular songs of different genres, with seven recordings each. The data is split in pre-defined reference/training and test partitions. The reference set comprises the original (studio) recording and a live version performed by the same artist for each song. The test set is, therefore, composed of five different versions of each song in the dataset.
- **Mazurkas:** This dataset is a collection of classical music. It comprises 2914 distinct recordings of 49 Chopin's Mazurkas obtained from the Mazurka Project³. The number of performances of each piece varies between 41 and 95. Unlike the YouTubeCovers dataset, the Mazurkas is not split into default partitions. We therefore assess this dataset using the leave-one-out approach.

4.2 Experimental Setup

To detail our experimental setup, we next describe the applied feature sets, the parameters of our method, and how we compare results.

²<https://sites.google.com/view/sucomusic>

³www.mazurka.org.uk/

4.2.1 Feature Extraction

Although some work explores the combination of different features to improve retrieval performance [17, 25], the usual procedure in the literature is to apply chroma-based features [8, 9, 12, 21, 23] that describe pitch perceived over time. More recently, deep learning-based methods have been used to extract cleaner chroma features from audio. In this work, we extract deep-chroma features [13] to describe the recordings in our datasets, using the Madmom tool library [5].

Since the Euclidean distance is sensitive to tempo differences, features are smoothed to provide robustness against this issue. Specifically, we used the technique applied by CENS, which uses a Hann window to smooth features in the time axis. Moreover, we reduced the dimensionality of the temporal axis of the chroma vectors by a factor of five. At the end of this procedure, each recording is represented by a vector containing two (smoothed) deep-chroma values per second of audio.

4.2.2 Parameter Settings

The two parameters of our method are the number and length of subsequences used to describe reference songs. We assessed three values of relative summary length: 10, 20, and 30 seconds (i.e., 20, 40, and 60 consecutive features). Using 10 seconds provide the worst results, and these results are not shown, while they point that using too small windows hampers performance.

Similarly, we assessed results using 1, 3 and 5 subsequences as the summaries set. We notice that while varying the set size does not significantly affect runtime, but the retrieval performance is clearly superior when using five segments.

Given the results of this parameter exploration, we henceforth present the results using five summaries of 30 seconds for the YouTubeCovers data. Because some recordings in the Mazurkas dataset are too short to apply summarization with 30 seconds per summary, we present results on this dataset using 20-second summaries.

4.2.3 Evaluation Measures and Compared Algorithms

Our evaluation consider three common evaluation measures: mean average precision (MAP); precision at 10 (P@10); and mean rank of first correct match (MR1). These measures allow us to compare SuCo against results presented in the literature.

For the YouTubeCovers, our experiments compare SuCo, Triplets [24], SiMPle [23], and a recent technique based on the 2-D Fourier Transform (which we refer as 2D-FT) [19]. The Mazurkas dataset has been less often used in the literature, so it is only possible to compare SuCo against SiMPle with this dataset.

Because previous evaluations of SiMPle did not use deep-chroma features, to isolate whether accuracy improvements in SuCo compared to previous ideas in SiMPle happen due to its use of deep-chromas or algorithmic improvements, we also run SiMPle with the same deep-chroma feature vector used by SuCo.

4.3 Runtime Performance

A central goal of SuCo is to create a fast method for similarity-based music information retrieval. We thus first focus on evaluating its runtime in our datasets⁴.

Note that this evaluation does not consider the duration of feature extraction, since it is common to all methods. Also, although we report the total runtime of SuCo's summarizing and comparing procedure, in practice the summarization is only performed once. For the SiMPle-based CSI, the reported runtime regards only the retrieval phase, as it does not rely on a training phase.

In the YouTubeCovers dataset, SuCo-thumb and SuCo-repeat run in 136 and 134 seconds, respectively. On the other hand, the SiMPle-based CSI took 4,192 seconds to assess the same dataset. That is, while our method takes a little more than 2 minutes to run, SiMPle (which is considered a fast algorithm) needs more than one hour. SuCo is more than 30 times faster than SiMPle.

This difference can be further observed in the Mazurkas dataset. While SuCo-thumb and SuCo-repeat take around 10 and 13 hours, respectively, to run the complete process, SiMPle only assess around 240 queries – out of 2914 – in the same runtime. This shows that in this larger dataset, SuCo is two order of magnitudes faster than SiMPle. Indeed, we aborted the execution of SiMPle for this dataset.

To break down the runtime for summarization and comparison in the SuCo pipeline, we isolate the runtime for summarizing in the YouTubeCovers dataset. This dataset has 100 reference recordings, and the summarization step for it takes around 20 seconds. This means that summarizing a new reference track takes around 0.2 seconds, and therefore that incrementing the training set using SuCo is nearly instantaneous.

4.4 Retrieval Performance

After efficiency, our second evaluation criteria is accuracy. Table 1 presents the results for our accuracy evaluation measures in the YouTubeCovers dataset. The SiMPle-deep refers to running regular SiMPle algorithm on the deep-chroma features.

Algorithm	MAP	P@10	MR1
Triplets [24]	0.48	0.13	8.49
SiMPle [23]	0.59	0.14	7.91
2D-FT [19]	0.65	0.14	8.27
SiMPle-Deep	0.78	0.17	3.66
SuCo-thumb	0.65	0.15	5.13
SuCo-repeat	0.74	0.17	3.80

Table 1: Results on the YouTubeCovers dataset

The results of SuCo and SiMPle-deep are a significant improvement over previous results presented in the literature for this dataset. SiMPle-deep presented the best results

⁴ This version of SuCo is implemented in Matlab. The experiments were carried out in a desktop computer with 16 Intel(R) Core(TM) i7 – 2600K CPU @ 3.20GHz and 64Gb of memory running Ubuntu 16.04. Also, at any time, there was only one process computing SuCo.

in this experiment, while SuCo-repeat achieved a very similar performance.

Table 2 presents the results for the Mazurkas classical music dataset.

Algorithm	MAP	P@10	MR1
SiMPle [23]	0.88	0.95	2.33
SuCo-thumb	0.83	0.93	2.83
SuCo-repeat	0.85	0.94	2.77

Table 2: Results on the Mazurkas dataset

Like in the YouTubeCovers data, SiMPle displays a slightly better performance than SuCo, in this case even without the deep-learned chroma features.

Taken together, the results on the two datasets point that SuCo is able to attain an accuracy very close to the best performing method while providing a much higher performance, with much lower runtime.

Besides, we notice that we may spend an extra few time to enhance our distance calculation, improving our retrieval performance and not significantly affecting the runtime. We discuss this topic in the next section.

5. ON REFINING THE EUCLIDEAN DISTANCE

The main purpose of using the Euclidean distance is its time efficiency and the possibility of exploring algorithms to further speeding up its application. However, we understand that it has a negative impact on the efficacy, since ED is sensitive to different distortions in the data.

While we reduce the impact of tempo variances by smoothing the feature vectors, our method is still sensitive to major differences. Applying a distance measure which is more robust to tempo differences in the entire SuCo pipeline could completely compromise our runtime performance. However, we believe that “refining” the distance calculation by some of these functions can improve our results.

To assure this argument, we made a subtle modification of the comparison algorithm. Once the best subsequence match is found using ED, we re-calculate the distance of the matched pair using the Open-End Dynamic Time Warping (DTW) [15] with a relative warping window of 10% of the subsequence length. For simplicity, we will refer to this optimization as SuCo-DTW.

Before presenting the results, we discuss another characteristic that may affect the ED calculation: the complexity of the data. Batista et al. [2] show that more complex time series, i.e., with high variations between consecutive observations, tends to present higher distances to its neighbors. Figure 2 illustrates an example of a simpler and a more complex chroma pattern.

To circumvent this issue, we estimate the complexity of each summary by the standard deviation of its chroma dimensions. The mean complexity of the twelve dimensions is taken as the summary’s complexity estimate. Finally, we adjust the distance of a query to each summary by diving it

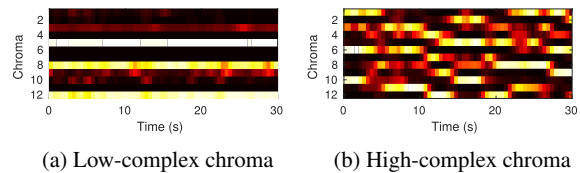


Figure 2: Examples of two different summaries with clearly different complexities

by the complexity estimate. For simplicity, we refer to this approach as SuCo-complexity.

To test our assumptions, we ran an experiment using these strategies on the YouTubeCovers dataset. Table 3 presents the results. In this experiment both SuCo-DTW and SuCo-complexity use the diverse repeated patterns as the summarizing method. As previously noted, we did not run the complete SiMPle-deep for the Mazurkas data, due to its impracticable runtime.

Algorithm	MAP	P@10	MR1
SiMPle-Deep	0.78	0.17	3.66
SuCo-DTW	0.80	0.18	3.42
SuCo-complexity	0.78	0.17	5.09

Table 3: Results on the YouTubeCovers dataset

Finally, refining the distance calculation does not severely affect the algorithm runtime. For instance, calculating the estimate complexities and “fix” the whole distance matrix in the YouTubeCovers dataset takes only 0.4 seconds. Also, the total runtime for running SuCo-DTW takes 459 seconds for the entire pipeline. Although it is slower than SuCo-thumb and SuCo-repeat, it is still around ten times faster than SiMPle and presents better retrieval performance. Investigating this kind of efficiency-precision trade-offs is part of our future works, presented with other concluding remarks in the next section.

6. CONCLUDING REMARKS

This paper presents and evaluates SuCo, a suite of methods for summarizing and comparing music data for fast content-based information retrieval. The techniques developed focus on the identification of cover songs. Our results demonstrate that it is possible to achieve results that are close to state-of-the-art algorithms while performing up to two orders of magnitudes faster depending on the dataset. Further improvements on the precision of SuCo with simple post-processing methods were also explored.

Future work may explore the SuCo pipeline in varied applications which rely on similarity comparisons. It also seems promising to further investigate methods to be added to this pipeline to improve precision. Future work may for example experiment with varied features and similarity measures, as well as with fusing different approaches to improve the retrieval efficacy [22].

7. ACKNOWLEDGEMENT

This work was funded by grants #2013/26151-5 and #2018/11755-6 São Paulo Research Foundation (FAPESP).

8. REFERENCES

- [1] Mark A Bartsch and Gregory H Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104, 2005.
- [2] Gustavo EAPA Batista, Eamonn J Keogh, Oben Moses Tataw, and Vinicius MA De Souza. Cid: an efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, 28(3):634–669, 2014.
- [3] Juan Pablo Bello. Audio-based cover song retrieval using approximate chord sequences: Testing shifts, gaps, swaps and beats. In *International Society for Music Information Retrieval Conference*, volume 7, pages 239–244, 2007.
- [4] Thierry Bertin-Mahieux and Daniel PW Ellis. Large-scale cover song recognition using the 2d Fourier transform magnitude. In *International Society for Music Information Retrieval Conference*, pages 241–246, 2012.
- [5] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. Madmom: A new Python audio and music signal processing library. In *ACM Multimedia Conference*, pages 1174–1178. ACM, 2016.
- [6] Ning Chen, Wei Li, and Haidong Xiao. Fusing similarity functions for cover song identification. *Multimedia Tools and Applications*, 77(2):2629–2652, 2018.
- [7] Hoang Anh Dau and Eamonn Keogh. Matrix profile v: A generic technique to incorporate domain knowledge into motif discovery. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 125–134. ACM, 2017.
- [8] Daniel PW Ellis and Graham E Poliner. Identifying-cover songs’ with chroma features and dynamic programming beat tracking. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages IV–1429. IEEE, 2007.
- [9] Jiunn-Tsair Fang, Yu-Ruey Chang, and Pao-Chi Chang. Deep learning of chroma representation for cover song identification in compression domain. *Multidimensional Systems and Signal Processing*, 29(3):887–902, 2018.
- [10] Eric J Humphrey, Oriol Nieto, and Juan Pablo Bello. Data driven and discriminative projections for large-scale cover song identification. In *International Society for Music Information Retrieval Conference*, pages 149–154, 2013.
- [11] Jesper Hojvang Jensen, Mads G Christensen, Daniel PW Ellis, and Soren Holdt Jensen. A tempo-insensitive distance measure for cover song identification based on chroma features. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2209–2212. IEEE, 2008.
- [12] Maksim Khadkevich and Maurizio Omologo. Large-scale cover song identification using chord profiles. In *International Society for Music Information Retrieval Conference*, pages 233–238, 2013.
- [13] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: The deep chroma extractor. In *International Society for Music Information Retrieval Conference*, pages 37–43, 2016.
- [14] Abdullah Mueen, Yan Zhu, Michael Yeh, Kaveh Kamgar, Krishnamurthy Viswanathan, Chetan Gupta, and Eamonn Keogh. The fastest similarity search algorithm for time series subsequences under euclidean distance, August 2017. <http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html>.
- [15] M Muller. Dynamic time warping (DTW). *Information Retrieval for Music and Motion*, pages 70–83, 2007.
- [16] Meinard Muller, Frank Kurth, and Michael Clausen. Chroma-based statistical audio features for audio matching. In *Applications of Signal Processing to Audio and Acoustics, 2005. IEEE Workshop on*, pages 275–278. IEEE, 2005.
- [17] Julien Osmalsky, Jean-Jacques Embrechts, Peter Foster, and Simon Dixon. Combining features for cover song identification. In *International Society for Music Information Retrieval Conference*, pages 462–468, 2015.
- [18] Julien Osmalskyj, Sébastien Piérard, Marc Van Droogenbroeck, and Jean-Jacques Embrechts. Efficient database pruning for large-scale cover song recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 714–718. IEEE, 2013.
- [19] Prem Seetharaman and Zafar Rafii. Cover song identification with 2d Fourier transform sequences. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 616–620. IEEE, 2017.
- [20] Joan Serra, Emilia Gómez, and Perfecto Herrera. Transposing chroma representations to a common key. In *IEEE CS Conference on The Use of Symbols to Represent Music and Multimedia Objects*, pages 45–48, 2008.
- [21] Joan Serra, Emilia Gómez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(6):1138–1151, 2008.

- [22] D. F. Silva, C. M. Yeh, Y. Zhu, E. Keogh, and G. E. A. P. A. Batista. Fast similarity matrix profile for music analysis and exploration. *IEEE Transactions on Multimedia*, 2018 (in press).
- [23] Diego F Silva, Chin-Chia M Yeh, Gustavo Enrique de Almeida Prado Alves Batista, and Eamonn Keogh. Simple: Assessing music similarity using subsequences joins. In *International Society for Music Information Retrieval Conference*, pages 23–29, 2016.
- [24] Diego Furtado Silva, Vinícius Mourão Alves de Souza, and Gustavo Enrique de Almeida Prado Alves Batista. Music shapelets for fast cover song recognition. In *International Society for Music Information Retrieval Conference*, pages 441–447, 2015.
- [25] Christopher J Tralie. Early MFCC and HPCP fusion for robust cover song identification. In *International Society for Music Information Retrieval Conference*, pages 294–301, 2017.
- [26] Christopher J Tralie and Paul Bendich. Cover song identification with timbral shape sequences. In *International Society for Music Information Retrieval Conference*, pages 38–44, 2015.
- [27] Wei-Ho Tsai, Hung-Ming Yu, Hsin-Min Wang, and Jorng-Tzong Horng. Using the similarity of main melodies to identify cover versions of popular songs for music document retrieval. *Journal of Information Science & Engineering*, 24(6), 2008.
- [28] Fan Yang and Ning Chen. Cover song identification based on cross recurrence plot and local alignment. *J. East China Univ. Sci. Technol*, 42(2):247–253, 2016.
- [29] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Zachary Zimmerman, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. *Data Mining and Knowledge Discovery*, 32(1):83–123, 2018.

TRANSFERRING THE STYLE OF HOMOPHONIC MUSIC USING RECURRENT NEURAL NETWORKS AND AUTOREGRESSIVE MODELS

Wei-Tsung Lu and Li Su

Institute of Information Science, Academia Sinica, Taiwan

s603122001@gmail.com, lisu@iis.sinica.edu.tw

ABSTRACT

Utilizing deep learning techniques to generate musical contents has caught wide attention in recent years. Within this context, this paper investigates a specific problem related to music generation, music style transfer. This practical problem aims to alter the style of a given music piece from one to another while preserving the essence of that piece, such as melody and chord progression. In particular, we discuss the style transfer of homophonic music, composed of a predominant melody part and an accompaniment part, where the latter is modified through Gibbs sampling on a generative model combining recurrent neural networks and autoregressive models. Both objective and subjective test experiment are performed to assess the performance of transferring the style of an arbitrary music piece having a homophonic texture into two different distinct styles, Bachs chorales and Jazz.

1. INTRODUCTION

Automatic music generation is gaining traction in the music industry because of its potential in mass producing music according to a user-assigned *style*, such as genre or mood. For example, the artificial intelligence (AI) music composition service, Jukedeck, supports four genre options (i.e., folk, rock, electronic, and ambient) and allows users to choose how the music feels (i.e., ambient, sparse, meditate, and sci-fi) [1]. Most of the existing advanced techniques employ deep learning techniques to perform end-to-end generative modeling of a music style based on a symbolic music format such as the musical instrument digital interface (MIDI) [3]. Various kinds of model configurations were explored in this fashion, such as the encoder-decoder framework [16], generative adversarial networks (GAN) [6], autoregressive models [13], variational autoencoders (VAE) [8], long-short-term memory (LSTM) networks [7, 12], recurrent Boltzmann machines (RBM) [2] and tied parallel networks [10]. These models are designed for two slightly different scenarios of music generation: one is to simply generate music by taking noise as

input [16], while the other is to generate adapted accompaniment or voices for a given melody or a lead sheet, also known as *reharmonization* [7] or *reorchestration* [14].

In this paper, we discuss the *music style transfer* problem. More specifically, we aim at *rearranging* the elements that highly affects style of a given music piece (e.g., rhythm patterns in the accompaniment) while preserving the essence (e.g., melody and chord progression) of that piece. This problem has been of interest for a long time; previous related studies include the use of genetic algorithm (GA) [15] and optimization approaches [19]. In contrast to reharmonization, the style transfer problem in this paper uses the entirety of a music piece, including all voices and accompaniment, as the input of a model. In other words, we aim to obtain a system that can automatically determine which part of an input music is to be preserved and which part is to be adapted to another style.

Besides, by leveraging the end-to-end modeling capability of deep learning, we investigate the potential of a single neural network to model two or more distinct styles based on training data of each style. To solve the style transfer problem, there are two main challenges, model complexity and the diversity over various musical genres. For model complexity, since the DeepBach model is designed for generating *polyphonic* music that only has a fixed number of voices (i.e., number of polyphony) such as Bach's four-part chorales instead of music having varying numbers of polyphony, such as Jazz music, the number of voices in the DeepBach model needs to be increased to accommodate the maximum number of voices. For example, the DeepBach model can be extended to 10 voices or more, but doing so also increases the model size and complexity considerably. For the second challenge, the diversity of various musical genres means that different music styles correspond to different preferable model setting, making it virtually impossible to develop a universal framework applicable to all kinds of music. For example, [14] proposed a solution to generate music having different styles but had to adopt different models for those styles.

This paper proposes a solution of music style transfer with one single generalized model. It assumes that input music is *homophonic* music decomposable into a *predominant melody* and an *accompaniment* part. Therefore, we only need to consider style transfer of the accompaniment; the melody, while being unaltered in the output, can be used as a condition of the network. We employ a DeepBach-based model to model temporal in-



© Wei-Tsung Lu and Li Su. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Wei-Tsung Lu and Li Su. "Transferring the Style of Homophonic Music Using Recurrent Neural Networks and Autoregressive Models", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

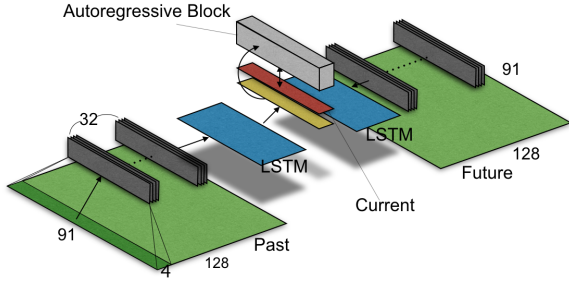


Figure 1: The proposed model for music style transfer. The past and future part of the input data (in green) is first transferred to another feature mapping (in dark gray) through a shared fully-connected network. The LSTM network then takes this feature mapping as input and outputs a 150-by-2 matrix (in yellow), which acts as the condition of the autoregressive model (in light gray). Finally, the autoregressive model predicts the activation of the interested note given the current part of the input data (in red).

formation, and combine this model with an autoregressive model (Wavenet [13]) to model the pitch information without restricting the number of note co-occurrence in a single frame. Experiment results of transferring the style of an arbitrary homophonic music piece to Bachs chorale and Jazz styles are provided in this paper since these two styles are arguably the two most extensively investigated music styles in the literature of automatic composition. The source code and listening samples are available on-line.¹

2. MODEL

Our proposed model of an input music piece is shown in Fig. 1. The model contains two LSTM networks, with one taking data preceding a reference time as input and another taking data following the reference time as input. In addition, a specific autoregressive model, WaveNet, is used to process the data of the reference time. Details of the proposed model are discussed below.

2.1 Data representation

We represent a music score as a *piano-roll* matrix $S \in \mathbb{R}_+^{I \times J}$ and a *metadata* matrix $M \in \mathbb{R}_+^{3 \times J}$. The concatenated matrix $[S; M]$ is then used as the input of the system. The element at the i -th row and the j -th column of S is denoted as S_{ij} , representing the pitch activation at pitch i and at time j , where $i \in [1, I]$, $j \in [1, J]$, $I = 88$, and J is the number of time steps of the music piece. $S_{:j} \in \mathbb{R}^I$ is then the j -th column of S , representing the pitch profile at time j . To represent homophonic music data, a note activation, S_{ij} for $i \in [1, 88]$, is represented as a Bernoulli random variable, and $S_{ij} = 1$ if there is a note activation at pitch i and time j , and $S_{ij} = 0$ if no note activation occurs at time j . Since the number of polyphony of homophonic music

varies with time, $S_{:j}$ becomes a multi-hot vector, where its number of non-zero elements varies with j .

The metadata matrix M describes the time grids, the start and the end symbol of the music piece, thereby forming a 3-by- J matrix. The time grids used in this paper are the same as the ones in DeepBach: each beat interval is divided into four subdivisions, and are indexed by 1, 2, 3, and 4 respectively. The starting time and ending time are denoted as 1 and others as 0. As a result, the dimension of the input, $[S_{:j}; M_{:j}]$, is 91.

To facilitate our discussion, we simplify the input data in the following two ways. First, sustained note are considered as repeating notes with the same pitch. Secondly, any two notes with the same time and pitch are considered as one note.

2.2 Model Architecture

The proposed model with parameterization θ is obtained by the following optimization problem:

$$\max_{\theta} \sum_{ij} \log p(S_{ij} = 1 | S_{\setminus ij}, M, \theta). \quad (1)$$

The formulation (1) can be viewed as a generalized version of the original DeepBach network discussed in [7], where the number of voices is fixed at 4 and the pitch of each voice is modeled individually:

$$\max_{\theta_i} \sum_j \log p_i(V_{ij} | V_{\setminus ij}, M, \theta_i), \text{ for } i \in [1, 4]. \quad (2)$$

The data representation $V_{ij} \in \mathbb{R}^{4 \times J}$ in (2) is different from S_{ij} ; V_{ij} is the pitch number of the i -th voice (i.e., 1 for soprano, 2 for alto, 3 for tenor, and 4 for bass) at time j . The four networks in DeepBach have the same structure, each processing only one part of the four-part Bach's chorales and producing an output limited to monophonic music. By using the four networks together, the conditional probability of notes occurred simultaneously in different parts can be covered.

We tackle our task on the basis of DeepBach because accessing both past and future parts of a score mitigates the problem of transition modeling [18], a major obstacle for music modeling. Besides, the temporal feature of music can be well captured with this model (shown in Section 3.4). Although DeepBach succeeds in handling Bachs 4-part chorales, it is not readily suitable for our problem scenario. To adapt the original DeepBach to accommodate more music types, especially for the music having a homophonic texture with varying numbers of polyphony, we remove the restriction of voicing and abandon the original four-network structure, and adopt one single network to process the multi-hot piano-roll representation.

Reducing the number of networks to one causes our generalized DeepBach to lose the ability to model the joint distribution of notes articulated simultaneously at a given time step. Besides, [7] indicated that using the piano-roll representation causes the generated result to be trapped in isolated regions during the Gibbs sampling process (see

¹ <https://github.com/s603122001/Music-Style-Transfer>

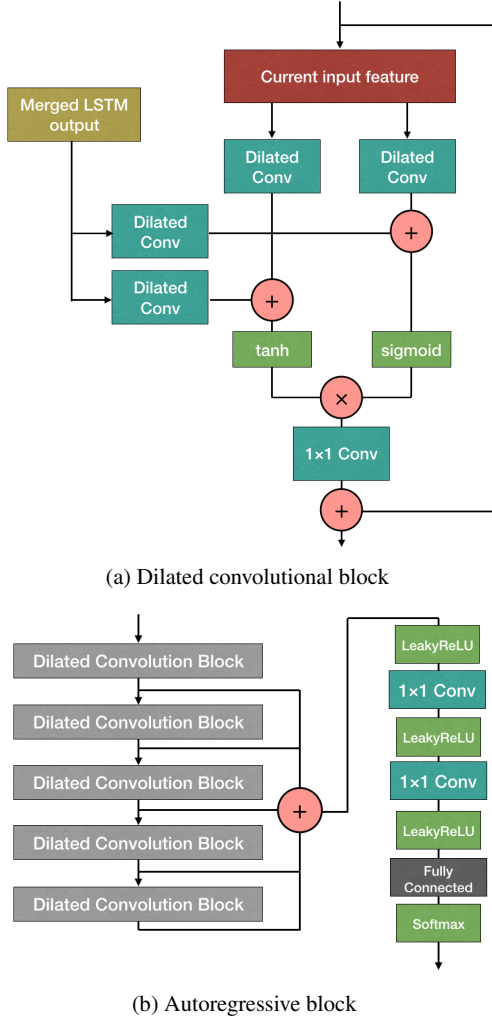


Figure 2: Illustration of the autoregressive block and dilated convolutional block of a conditional Wavenet.

Section 2.4). In order to overcome these issues, we employ an autoregressive model, Wavenet [13], to control the relationship among the 88 possible pitch activations at a given time step j . The joint distribution of $S_{:j} = \{S_{1j}, \dots, S_{88j}\}$ can be written as the product of conditional probabilities of all pitches:

$$p(S_{:j}) = \prod_{i=1}^{88} P(S_{ij} | S_{1j}, \dots, S_{(i-1)j}, S_{\setminus S_{:j}}). \quad (3)$$

The Wavenet models the conditional probabilities by stacking dilated causal convolution layers [13]. We then employ the output of the generalized DeepBach model as a constraint. This is implemented by a dilated convolutional block with constraint (see Fig. 2a) and an autoregressive block to predict the output by running from $i = 1$ to $i = 88$ (see Fig. 2b). Implementation details of them can be seen in [13].

In summary, the output of the generalized DeepBach represents the temporal context of music, and it constrains the Wavenet model to ensure that the output is musically

reasonable in terms of the harmony progression and other contextual structures.

2.3 Implementation details

The model is implemented using the Keras [4] library with tensorflow as the back end. First, input data is divided, such that each unit of the input data contains a segment of four time steps, i.e., a 91×4 matrix, and the segments do not overlap (see the dark green part in Fig. 1). Every segment is first flattened, and the flattened segment is embedded into a 150-D vector with a shared fully connected layer for dimension reduction (see the dark gray part in Fig. 1), so as to incorporate information over a larger temporal range with a smaller model capacity. Similar to the original DeepBach model, two LSTM models are employed, one dealing with the past embedded feature mappings and the other dealing with the future ones. Both LSTM networks take a series of embedded features with 32 time steps, i.e., a 150-by-32 matrix, as the input. Both networks contain 4 LSTM layers, each having 150 hidden units (see the blue block in Fig. 1). The outputs of the two networks are concatenated and then transformed to an 88-D vector with another fully-connected layer. This merged LSTM output is then used as the condition of the Wavenet that employs the original input feature map at the current time step, as illustrated in Fig. 2a. The Wavenet consists of five dilated convolution layers as shown in Fig. 2b, where only the top two of the five layers are conditioned by the merged LSTM output and the other three are not conditioned. A dropout rate of 30% is adopted for each layer, and batch normalization is added after the activation of each convolutional layer. The model is optimized using ADAM [11].

2.4 Style transfer

The algorithm of style transfer is shown in Algorithm 1, and the number of pitch range p is 88 in this paper. Inspired by the idea in [7], the style transfer is conducted by using Gibbs sampling to sample the model and then performing an iterative update on the elements of the input score matrix. In contrast to those models using noise as input [7, 16], the input of our model is the music score to be transferred, and this initialization enables the resulting musical structure to follow the original one. In every iteration of the optimization process, all the elements at the same time step in the input matrix (including both note activation and silence) are visited and updated iteratively. It is important to point out that all notes at the same time step are updated together for the chosen target time step. While doing this partly violates the original concept of Gibbs sampling, it produces stable results in the experiments.

Sampling from an autoregressive model is inefficient due to the sequential property that every output is conditioned on all the previous ones. To speed up, we adopt the strategy of independent Gibbs sampling [9, 17]. Independent Gibbs sampling uses an *annealed* masking probability α that controls the percentage of the elements in the matrix that are to be updated independently, making the input approach a stable condition in a short time [9, 17]. In the n -th

Algorithm 1 Music Style Transfer

Input: I by J score matrix S , number of pitch range P , maximum number of iteration N , maximum and minimum annealed masking probability $[\alpha_{max} \ \alpha_{min}]$, annealed masking ratio η

```

1:  $\alpha \leftarrow \alpha_{max}$ ,  $\hat{S} \leftarrow S$ ,  $c \leftarrow 0$ 
2: for  $n$  from 1 to  $N$  do
3:   Choose time index  $j$  in the range of  $J$ 
4:   if  $\alpha > \alpha_{min}$  then
5:     Update  $\{S_{ij}\}_{i=1}^P$  by  $p(S_{ij}|\hat{S}_{1j}, \dots, \hat{S}_{(i-1)j}, \hat{S}_{(i+1)j}, \dots, \hat{S}_{Jj})$ 
6:      $c \leftarrow c + P$ 
7:     if  $c > (\alpha \cdot P \cdot J)$  then
8:        $c \leftarrow 0$ 
9:        $\hat{S} \leftarrow S$ 
10:    end if
11:     $\alpha \leftarrow \alpha - \frac{\alpha_{max} - \alpha_{min}}{\eta \cdot N}$ 
12:  else
13:    Update  $\{S_{ij}\}_{i=1}^P$  by  $p(S_{ij}|S_{1j}, \dots, S_{(i-1)j}, S_{(i+1)j}, \dots, S_{Jj})$ 
14:  end if
15: end for
Output: Transferred score matrix  $\hat{S}$ 

```

iteration of such a sampling process, and for some maximal and minimal probabilities α_{max} and α_{min} , α is updated by the following formula:

$$\alpha_n = \max \left(\alpha_{min}, \alpha_{max} - \frac{n(\alpha_{max} - \alpha_{min})}{\eta N} \right), \quad (4)$$

where N and η represent the total number of Gibbs steps and the annealed masking ratio controlling the required time for α approaching α_{min} . As α is reduced to the minimum, the procedure approximates the standard Gibbs sampling, which updates only one element at one time and compensates the poor result produced in the independent phase. The advantage of using independent Gibbs sampling is its efficiency. Besides, independent Gibbs sampling gives more stable outcomes, especially when transferring to challenging genres like Jazz.

This research also discovers that during the transfer process, the melody in the original music tends to vanish during the iteration. As a result, we currently apply a constraint on the melody part to address this issue, and leave the style transfer of the melody part to future work.

3. EXPERIMENTS AND DISCUSSION

3.1 Datasets

Two datasets, Bach’s four-part chorales and Jazz music, are employed as the training data. The Bach dataset contains 357 Bach four-part chorales included in the music21 toolkit [5]. The Jazz dataset contains 487 songs either collected manually on-line or generated on our own according to the scores in the well-known Real Book. To simplify the experiment, we did not distinguish among the sub-genres of Jazz, and all of the Jazz pieces are played in Jazz trio,

containing one piano, one double bass and one drum. The drummer part is ignored since we consider only the harmonic part of music in this paper. In the training process, we perform data augmentation, by pitch-shifting each song in the two datasets up and down by at most 6 semitones in order to cover all possible keys. As a result, we have 4858 pieces and 6331 pieces in the Bach dataset and the Jazz dataset, respectively. In addition, the two datasets are compiled in different time resolutions. In the Bach dataset, a sixteenth note is defined as one time step, while in the Jazz dataset, a thirty-second note is defined as one time step, as the latter one contains faster note groups.

Four songs with different styles were selected as the testing data: *Rocky Raccoon* by Beatles, *Paranoid Android* by Radiohead, *Live and Let Die* by Paul McCartney, and Beethoven’s *Moonlight Sonata*, Op. 27, No. 2. Each of the song was cropped to 30 seconds long. These four testing songs will be used in both the objective evaluation and the subjective test.

3.2 Experiment settings

Experiments are conducted to demonstrate the effect of our solution to the task of transferring the style of an input music piece to Bach or Jazz style, and we employ the proposed models trained from the afore-mentioned datasets of Bachs chorales and Jazz, respectively. To verify the capability of the network in modeling signals with a varying number of voices, we employed two different versions of the network, one without the autoregressive model (denoted as “LSTM only”), and the other incorporating the autoregressive model (denoted as “LSTM-WN”). We compare the following three models:

1. LSTM-to-Bach: transfer the style of input music to Bach’s chorale using the LSTM network only.
2. LSTM-WN-to-Bach: transfer the style of input music to Bach’s chorale using the LSTM combined with the Wavenet.
3. LSTM-WN-to-Jazz: transfer the style of input music to Jazz using the LSTM combined with the Wavenet.

LSTM-to-Jazz is excluded from the experiment results because our pilot study showed that without the autoregressive model, the generated outputs appear to be composed of random note groups due to the wide diversity of the Jazz dataset. Since a subjective test is hard to be performed with such output samples, we eliminate this case in the following experiments. For further comparison, we also compare our model with the original DeepBach model, under the scenario of reharmonizing given melodies using a pre-trained DeepBach model.

3.3 Metrics for objective evaluation

To analyze the performance of a style transfer system and to compare the songs before and after a transferring process, we divide the style transfer task into 3 sub-parts and evaluate them with different metrics.

1. *Content preservation of the original style.* According to our definition of music style transfer, a good music style transfer system should preserve the backbone of an input song. This means that the overall structure of a song, such as chord progression, should not be changed. Therefore, the content similarity between the original song and the transformed one should be considered. To evaluate the content similarity between two music pieces, we first transfer the MIDI representation (in piano roll) of every time step into a chroma vector with the 12 pitch classes, then compute the moving average of the chroma vector within a frame size of half bar, which is 8 time steps in the Bach's dataset and 16 time steps in the Jazz dataset. Finally, the cosine similarity between every time step in the two score matrices is calculated to measure the content preservation degree of the transfer process.
2. *Harmony structure similarity to the transferal target style.* Common harmony sets (i.e., combination of notes) vary across different music styles. For example, chords with an extension note like 9th and 11th are more often used in Jazz music than in Bach's chorales. This characteristic is utilized to visualize how such distribution changes after a style transfer. First the transfer target styles of interest in this paper are represented by collecting all existing harmony combinations in the two datasets, and then mapping them to a 2-D plane by using the principal component analysis (PCA) and then the t-distributed stochastic neighbor embedding (t-SNE) method. By visualizing such 2-D features of the original and transferred songs on the plane, we could observe the difference of locations between them, and how the transferred song moves toward the target dataset.
3. *Temporal structure imitating the transferred style.* Rhythmic patterns are an important characteristic in distinguish different music styles. To model this property, we utilize the fact that rhythm has a strong correlation with the timing of harmony changes. For example, it is common that a chord change coincides with a strong beat. Therefore, we define the harmonic transition point of a song to be a time step where at least three notes change in comparison to the previous time step, and then we plot the distribution of these points within every bar with the temporal unit being an eighth note. The result is a 32-D vectors representing the major pattern of rhythm and harmonic transitions. We examine how similar such pattern of the transferred music pieces is to the pattern of a target dataset.

3.4 Objective evaluation

Table 1 shows the performance index of content preservation, the average cosine similarity, of the four testing songs before and after style transfer. We list the results of the proposed models and the original DeepBach [7], being used as

	To Bach	To Jazz
DeepBach	0.39	N/A
LSTM-to-Bach	0.86	N/A
LSTM-WN-to-Bach	0.76	N/A
LSTM-WN-to-Jazz	N/A	0.56

Table 1: Evaluation results of Content Preservation in terms of cosine similarity.

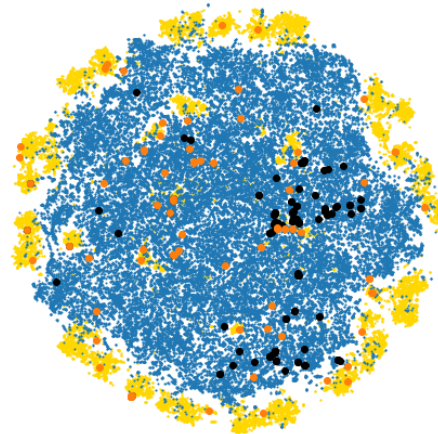


Figure 3: Result of the Harmony Distribution. Data points are the piano roll features mapped to a 2-D space through PCA and tSNE. Blue: Jazz dataset. Yellow: Bach dataset. Black: testing clips of Jazz music. Orange: testing clips transferred to Bach's style.

a baseline. We do not use the original DeepBach model for style transfer to Jazz because its number of voice is fixed at 4. Notably, the original DeepBach, which uses only the main melody to perform reharmonization, is less effective in following the structure of the original pieces than the proposed models designed for homophonic music.

Fig. 3 illustrates the harmony distribution of the cropped segments of five pieces in the Jazz dataset, before and after a style transfer using the LSTM-WN-to-Bach model. Here we illustrate the result of Jazz data instead of the testing songs because this is a genre-to-genre comparison. As shown in Fig. 3, the resulting harmony distributions indicate that all data points of the Jazz data are originally within the distribution of the Jazz dataset. After style transfer, most of the data points move toward the distribution of the Bach dataset, and some of the transferred data points are even located within the Bach distribution.

Fig. 4 shows the results of temporal structure similarity of the four testing songs before and after style transfer. We used the same songs and models in the content preservation part. Results show that for the transfer-to-Bach case, all models fit fairly well to the pattern representing the Bach dataset, with the original DeepBach model producing a few extra transitions unseen in the Bach dataset. For the transfer-to-Jazz task, the proposed model also fits the pattern of the target dataset well.



Figure 4: Evaluation results on temporal structures.

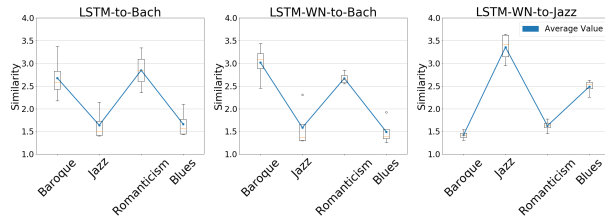


Figure 5: Result of the subjective test. The scores represent the subjects' evaluation on how similar the style of music to the genre listed in the questionnaire.

3.5 Subjective tests

To evaluate the performance of our model from a human perception perspective, a listening test was conducted with 63 participants. Among the participants, 51 of them have the experience of being a music performer, and 17 of those 51 participants receive formal music education or have work experience in related fields.

Each of the four testing songs was transferred using the three afore-mentioned models: LSTM-to-Bach, LSTM-WN-to-Bach, and LSTM-WN-to-Jazz, respectively. As a result, three different versions were produced for each song, and every participant evaluated a total of 12 songs. For each song to be transferred, the participants were asked to determine the style of the main melody from 4 music styles: Baroque music (i.e., Bachs), Jazz, Romanticism, and Blues. This question aims to direct their attention to both melody and accompaniment parts. Note that Romanticism and Blues are extra options added to avoid a possible bias in the questionnaire. After answering the question, the participants then evaluated the degree of similarity between the transferred songs and the 4 music styles above, based on their music knowledge and personal perception. The evaluation was in the scale from 1 (low) to 5 (high).

The results of the subjective test are shown in Fig. 5. The results are averaged for different models. It can be seen that, for the cases of transferring to Bach's style, the rated degrees of similarity of the transferred songs to *Baroque* and *Romanticism* are both high. According to a

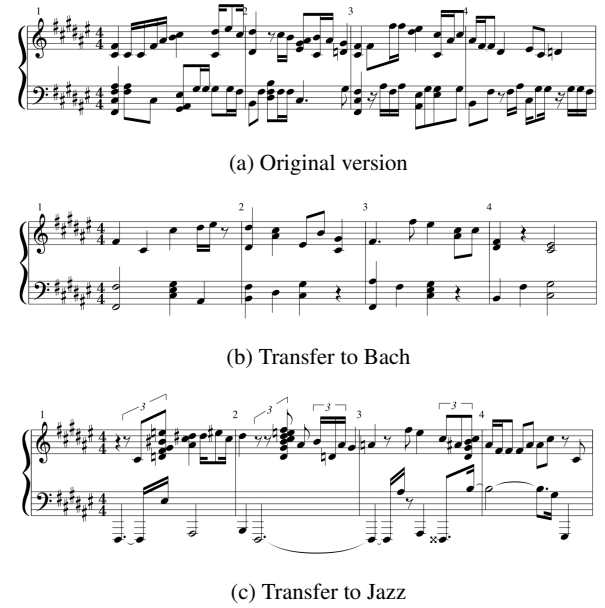


Figure 6: Transfer results of *Live and Let Die* using the proposed model. The score is output by LogicPro.

participant who is a major in music, this phenomenon is related to the main melodies and the original songs we pick. However, the model with Wavenet produced transferred songs rated with the highest similarity degree to *Baroque*, demonstrating the necessity of the Wavenet component in our model. For the case of transferring to Jazz style, the degree of similarity to *Jazz* surpasses other types of music.

One test sample used in the subjective test is outputted as music scores illustrated in Fig. 6 to give us some insights into the capability of the proposed models. In Fig. 6(b), the harmony and music contents are simplified with respect to the original version since the contents of Bach's 4-part chorales are usually not complicated. Apart from this, the difference between the temporal structure of the two scores is a clear example that our model has learned the temporal feature of the music style. In Fig. 6(c), we find many non-chord notes and some taste of syncopated rhythm, both marking the characteristics of Jazz music.

4. CONCLUSION AND FUTURE WORK

We have demonstrated the capability of our model in transferring arbitrary homophonic music scores into the styles of Bach's 4-part chorales and Jazz, and both objective and subjective tests are conducted. The advantage of our method is that it does not pose any restrictions on input music scores, and thus it can be easily applied in other scenarios. Besides, different styles of music can be modeled using the same framework, simplifying the process when we want to expand the collection of target music genres. Future work will focus on the style transfer of a melody, which is not considered in this paper, as well as further investigation into complicated music styles and extensive applications based on the proposed model.

5. ACKNOWLEDGEMENT

We would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work is partially supported by MOST Taiwan, under the contract MOST 106-2218-E-001-003-MY3.

6. REFERENCES

- [1] Jukedeck. <https://www.jukedeck.com>. [Online; accessed 26-Nov-2017].
- [2] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proc. International Conference on Machine Learning (ICML)*, 2012.
- [3] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation—a survey. *arXiv preprint arXiv:1709.01620*, 2017.
- [4] François Chollet et al. Keras. <https://github.com/fchollet/keras>, 2015.
- [5] Cuthbert, Michael Scott, and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. 2010.
- [6] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Symbolic-domain music generation and accompaniment with multi-track sequential generative adversarial networks. *arXiv preprint arXiv:1709.06298*, 2017.
- [7] Gaëtan Hadjeres and François Pachet. Deepbach: a steerable model for bach chorales generation. In *Proc. International Conference on Machine Learning (ICML)*, 2017.
- [8] Jay A Hennig, Akash Umakantha, and Ryan C Williamson. A classifying variational autoencoder with application to polyphonic music generation. *arXiv preprint arXiv:1711.07050*, 2017.
- [9] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. In *ISMIR*, 2017.
- [10] Daniel D Johnson. Generating polyphonic music using tied parallel networks. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 128–143. Springer, 2017.
- [11] Kingma, Diederik, and Jimmy Ba. Adam: A method for stochastic optimization. 2014.
- [12] Feynman Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. Automatic stylistic composition of bach chorales with deep lstm. In *ISMIR*, 2017.
- [13] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [14] François Pachet. A joyful ode to automatic orchestration. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):18, 2016.
- [15] Dimitrios Tzimeas and Eleni Mangina. Jazz sebastian bach: A ga system for music style modification. In *Systems and Networks Communications, 2006. ICSNC'06. International Conference on*. IEEE, 2006.
- [16] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. In *ISMIR*, 2017.
- [17] Li Yao, Sherjil Ozair, Kyunghyun Cho, and Yoshua Bengio. On the equivalence between deep nade and generative stochastic networks. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 322–336, 2014.
- [18] Adrien Ycart and Emmanouil Benetos. A study on lstm networks for polyphonic music sequence modelling. In *ISMIR*, 2017.
- [19] Frank Zalkow, Stephan Brand, and Benjamin Graf. Musical style modification as an optimization problem. In *Proc. Int Conf. Computer Music (ICMC)*, 2016.

MIDI-VAE: MODELING DYNAMICS AND INSTRUMENTATION OF MUSIC WITH APPLICATIONS TO STYLE TRANSFER

Gino Brunner Andres Konrad Yuyi Wang Roger Wattenhofer

Department of Electrical Engineering and Information Technology

ETH Zurich

Switzerland

brunnegi, konradan, yuwang, wattenhofer@ethz.ch

ABSTRACT

We introduce MIDI-VAE, a neural network model based on Variational Autoencoders that is capable of handling polyphonic music with multiple instrument tracks, as well as modeling the dynamics of music by incorporating note durations and velocities. We show that MIDI-VAE can perform style transfer on symbolic music by automatically changing pitches, dynamics and instruments of a music piece from, e.g., a Classical to a Jazz style. We evaluate the efficacy of the style transfer by training separate style validation classifiers. Our model can also interpolate between short pieces of music, produce medleys and create mixtures of entire songs. The interpolations smoothly change pitches, dynamics and instrumentation to create a harmonic bridge between two music pieces. To the best of our knowledge, this work represents the first successful attempt at applying neural style transfer to complete musical compositions.

1. INTRODUCTION

Deep generative models do not just allow us to generate new data, but also to change properties of existing data in principled ways, and even transfer properties between data samples. Have you ever wanted to be able to create paintings like Van Gogh or Monet? No problem! Just take a picture with your phone, run it through a neural network, and out comes your personal masterpiece. Being able to generate new data samples and perform style transfer requires models to obtain a deep understanding of the data. Thus, advancing the state-of-the-art in deep generative models and neural style transfer is not just important for transforming horses into zebras,¹ but lies at the very core of Deep (Representation) Learning research [2].

While neural style transfer has produced astonishing results especially in the visual domain [21, 37], the progress

for sequential data, and in particular music, has been slower. We can already transfer sentiment between restaurant reviews [30, 36], or even change the instrument with which a melody is played [33], but we have no way of knowing how our favorite pop song would have sounded if it were written by a composer who lived in the classical epoch or how a group of jazz musicians would play the Overture of Mozart’s Don Giovanni. In this work we take a step towards this ambitious goal. To the best of our knowledge, this paper presents the first successful application of unaligned style transfer to musical compositions. Our proposed model architecture consists of parallel Variational Autoencoders (VAE) with a shared latent space and an additional style classifier. The style classifier forces the model to encode style information in the shared latent space, which then allows us to manipulate existing songs, and effectively change their style, e.g., from Classic to Jazz. Our model is capable of producing harmonic polyphonic music with multiple instruments. It also learns the dynamics of music by incorporating note durations and velocities.

2. RELATED WORK

Gatys et al. [14] introduce the concept of neural style transfer and show that pre-trained CNNs can be used to merge the style and content of two images. Since then, more powerful approaches have been developed [21, 37]; these allow, for example, to render an image taken in summer to look like it was shot in winter. For sequential data, autoencoder based methods [30, 36] have been proposed to change the sentiment or content of sentences. Van den Oord et al. [33] introduce a VAE model with discrete latent space that is able to perform speaker voice transfer on raw audio data. Mor et al. [26] develop a system based on a WaveNet autoencoder [12] that can translate music across instruments, genres and styles, and even create music from whistling. Malik et al. [23] train a model to add note velocities (loudness) to sheet music, resulting in more realistic sounding playback. Their model is trained in a supervised manner, with the target being a human-like performance of a music piece in MIDI format, and the input being the same piece but with all note velocities set to the same value. While their model can indeed play music in a more human-like manner, it can only change note velocities, and does not

¹ <https://junyanz.github.io/CycleGAN/>



© Gino Brunner, Andres Konrad, Yuyi Wang, Roger Wattenhofer. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Gino Brunner, Andres Konrad, Yuyi Wang, Roger Wattenhofer. “MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

learn the characteristics of different musical styles/genres. Our model is trained on unaligned songs from different musical styles. Our model can not only change the dynamics of a music piece from one style to another, but also automatically adapt the instrumentation and even the note pitches themselves. Apart from style transfer, our model can also generate short pieces of music, medleys, interpolations and song mixtures. At the core of our model thus lies the capability to produce music. In the following we will therefore discuss related work in the domains of symbolic and raw audio generation. For a more comprehensive overview we refer the interested readers to these surveys: [4, 13, 16].

People have been trying to compose music with the help of computers for decades. One of the most famous early examples is “Experiments in Musical Intelligence” [9], a semi-automatic system based on Markov models that is able to create music in the style of a certain composer. Soon after, the first attempts at music composition with artificial neural networks were made. Most notably, Todd [31], Mozer [27] and Eck et al. [11] all used Recurrent Neural Networks (RNN). More recently, Boulanger-Lewandowski et al. [3] combined long short term memory networks (LSTMs) and Restricted Boltzmann Machines to simultaneously model the temporal structure of music, as well as the harmony between notes that are played at the same time, thus being capable of generating polyphonic music. Chu et al. [7] use domain knowledge to model a hierarchical RNN architecture that produces multi-track polyphonic music. Brunner et al. [5] combine a hierarchical LSTM model with learned chord embeddings that form the Circle of Fifths, showing that even simple LSTMs are capable of learning music theory concepts from data. Hadjeres et al. [15] introduce an LSTM-based system that can harmonize melodies by composing accompanying voices in the style of Bach Chorales, which is considered a very difficult task even for professionals. Johnson et al. [18] use parallel LSTMs with shared weights to achieve transposition-invariance (similar to the translation-invariance of CNNs). Chuan et al. [8] investigate the use of an image-based Tonnetz representation of music, and apply a hybrid LSTM/CNN model to music generation.

Generative models such as the Variational Autoencoder (VAE) and Generative Adversarial Networks (GANs) have been increasingly successful at modeling music. Roberts et al. introduce MusicVAE [29], a hierarchical VAE model that can capture long-term structure in polyphonic music and exhibits high interpolation and reconstruction performance. GANs, while very powerful, are notoriously difficult to train and have generally not been applied to sequential data. However, Mogren [25], Yang et al. [34] and Dong et al. [10] have recently shown the efficacy of CNN-based GANs for music composition. Yu et al. [35] were the first to successfully apply RNN-based GANs to music by incorporating reinforcement learning techniques.

Researchers have also worked on generating raw audio waves. Van den Oord et al. [32] introduce WaveNet, a CNN-based model for the conditional generation of

speech. The authors also show that it can be used to generate pleasing sounding piano music. More recently, Engel et al. [12] incorporated WaveNet into an Autoencoder structure to generate musical notes and different instrument sounds. Mehri et al. [24] developed SampleRNN, an RNN-based model for unconditional generation of raw audio. While these models are impressive, the domain of raw audio is very high dimensional and it is much more difficult to generate pleasing sounding music. Thus most existing work on music generation uses symbolic music representations (see e.g., [3, 5, 7–10, 15, 18, 23, 25, 27, 29, 31, 34, 35]).

3. MODEL ARCHITECTURE

Our model is based on the Variational Autoencoder [20] (VAE) and operates on a symbolic music representation that is extracted from MIDI [1] files. We extend the standard piano roll representation of note pitches with velocity and instrument rolls, modeling the most important information contained in MIDI files. Thus, we term our model MIDI-VAE. MIDI-VAE uses separate recurrent encoder/decoder pairs that share a latent space. A style classifier is attached to parts of the latent space to make sure the encoder learns a compact latent style label that we can then use to perform style transfer. The architecture of MIDI-VAE is shown in Figure 1, and will be explained in more detail in the following.

3.1 Symbolic Music Representation

We use music files in the MIDI format, which is a symbolic representation of music that resembles sheet music. MIDI files have multiple tracks. Tracks can either be *on* with a certain pitch and velocity, *held* over multiple time steps or be *silent*. Additionally, an instrument is assigned to each track. To feed the note pitches into the model we represent them as a tensor $P \in \{0, 1\}^{n_P \cdot n_B \cdot n_T}$ (commonly known as piano roll and henceforth referred to as pitch roll), where n_P is the number of possible pitch values, n_B is the number of beats and n_T is the number of tracks. Thus, each song in the dataset is split into pieces of length n_B . We choose n_B such that each piece corresponds to one bar. We include a “silent” note pitch to indicate when no note is played at a time step. The note velocities are encoded as tensor $V \in [0, 1]^{n_P \cdot n_B \cdot n_T}$ (velocity roll). Velocity values between 0.5 and 1 signify a note being played for the first time, whereas a value below 0.5 means that either no note is being played, or that the note from the last time step is being held. The note velocity range defined by MIDI (0 to 127) is mapped to the interval $[0.5, 1]$. We model the assignment of instruments to tracks as matrix $I = \{0, 1\}^{n_T \cdot n_I}$ (instrument roll), where n_I is the number of possible instruments. The instrument assignment is a global property and thus remains constant over the duration of one song. Finally, each song in our dataset belongs to a certain style, designated by the style label $S \in \{\textit{Classic}, \textit{Jazz}, \textit{Pop}, \textit{Bach}, \textit{Mozart}\}$.

In order to generate harmonic polyphonic music it is important to model the joint probability of simultane-

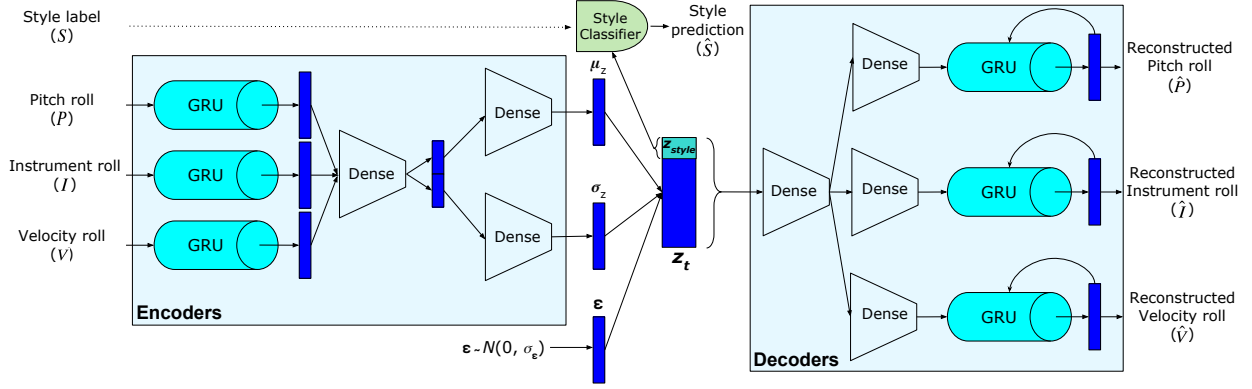


Figure 1. MIDI-VAE architecture. GRU stands for Gated Recurrent Unit [6].

ously played notes. A standard recurrent neural network model already models the joint distribution of the sequence through time. If there are multiple outputs to be produced per time step, a common approach is to sample each output independently. In the case of polyphonic music, this can lead to dissonant and generally “wrong” sounding note combinations. However, by unrolling the piano rolls in time we can let the RNN learn the joint distribution of simultaneous notes as well. Basically, instead of one n_T -hot vector for each beat, we input n_T 1-hot vectors per beat to the RNN. This is a simple but effective way of modeling the joint distribution of notes. The drawback is that the RNN needs to model longer sequences. We use the `pretty_midi` [28] Python library to extract information from MIDI files and convert them to piano rolls.

3.2 Parallel VAE with Shared Latent Space

MIDI-VAE is based on the standard VAE [20] with a hyperparameter β to weigh the Kullback-Leibler divergence in the loss function (as in [17]). A VAE consists of an encoder $q_\theta(z|x)$, a decoder $p_\phi(x|z)$ and a latent variable z , where q and p are usually implemented as neural networks parameterized by θ and ϕ . In addition to minimizing the standard autoencoder reconstruction loss, VAEs also impose a prior distribution $p(z)$ on the latent variables. Having a known prior distribution enables generation of new latent vectors by sampling from that distribution. Furthermore, the model will only “use” a new dimension, i.e., deviate from the prior distribution, if it significantly lowers the reconstruction error. This encourages disentanglement of latent dimensions and helps learning a compact hidden representation. The VAE loss function is

$$\mathcal{L}_{VAE} = \mathbb{E}_{q_\theta(z|x)} [\log p_\phi(x|z)] - \beta D_{KL} [q_\theta(z|x) || p(z)],$$

where the first term corresponds to the reconstruction loss, and the second term forces the distribution of latent variables to be close to a chosen prior. D_{KL} is the Kullback-Leibler divergence, which gives a measure of how similar two probability distributions are. As is common practice, we use an isotropic Gaussian distribution with unit variance as our prior, i.e., $p(z) = \mathcal{N}(0, I)$. Thus, both $q_\theta(z|x)$

and $p(z)$ are (isotropic) Gaussian distributions and the KL divergence can be computed in closed form.

As described in Section 3.1, we represent multi-track music as a combination of note pitches, note velocities and an assignment of instruments to tracks. In order to generate harmonic multi-track music, we need to model a joint distribution of these input features instead of three marginal distributions. Thus, our model consists of three encoder/decoder pairs with a shared latent space that captures the joint distribution. For each input sample (i.e., a piece of length n_B beats), the pitch, velocity and instrument rolls are passed through their respective encoders, implemented as RNNs. The output of the three encoders is concatenated and passed through several fully connected layers, which then predict σ_z and μ_z , the parameters of the approximate posterior $q_\theta(z|x) = \mathcal{N}(\mu_z, \sigma_z)$.² Using the reparameterization trick [20], a latent vector z is sampled from this distribution as $z \sim \mathcal{N}(\mu_z, \sigma_z * \epsilon)$ where $*$ stands for element-wise multiplication. This is necessary because it is generally not possible to backpropagate gradients through a random sampling operation, since it is not differentiable. ϵ is sampled from an isotropic Gaussian distribution $\mathcal{N}(0, \sigma_\epsilon * I)$, where we treat σ_ϵ as a hyperparameter (see Section 4.2 for more details). This shared latent vector is then fed into three parallel fully connected layers, from which the three decoders try to reconstruct the pitch, velocity and instrument rolls. The note pitch and instrument decoders are trained with cross entropy losses, whereas for the velocity decoder we use MSE.

3.3 Style Classifier

Having a disentangled latent space might enable some control over the style of a song. If for example one dimension in the latent space encodes the dynamics of the music, then we could easily change an existing piece by only varying this dimension. Choosing a high value for β (the weight of the KL term in the VAE loss function) has been shown to increase disentanglement of the latent space in the visual domain [17]. However, increasing β has a negative effect

² We use notation σ for both a variance vector and the corresponding diagonal variance matrix.

Dataset	#Songs	#Bars	Artists
Classic	477	60523	Beethoven, Clementi, ...
Jazz	554	72190	Sinatra, Coltrane, ...
Pop	659	65697	ABBA, Bruno Mars, ...
Bach	156	16213	Bach
Mozart	143	17198	Mozart

Table 1. Properties of our dataset.

on the reconstruction performance. Therefore, we introduce additional structure into the latent space by attaching a softmax style classifier to the top k dimensions of the latent space (z_{style}), where k equals the number of different styles in our dataset. This forces the encoder to write a “latent style label” into the latent space. Using only k dimensions and a weak classifier encourages the encoder to learn a compact encoding of the style. In order to change a song’s style from S_i to S_j , we pass the song through the encoder to get z , swap the values of dimensions z_{style}^i and z_{style}^j , and pass the modified latent vector through the decoder. As style we choose the music genre (e.g., Jazz, Pop or Classic) or individual composers (Bach or Mozart).

3.4 Full Loss Function

Putting all parts together, we get the full loss function of our model as

$$\begin{aligned} \mathcal{L}_{tot} = & \lambda_P H(P, \hat{P}) + \lambda_I H(I, \hat{I}) \\ & + \lambda_V MSE(V, \hat{V}) + \lambda_S H(S, \hat{S}) - \beta D_{KL}(q||p), \end{aligned} \quad (1)$$

where $H(\cdot, \cdot)$, $MSE(\cdot, \cdot)$ and $D_{KL}(\cdot||\cdot)$ stand for cross entropy, mean squared error and KL divergence respectively. The hats denote the predicted/reconstructed values. The weights λ and β can be used to balance the individual terms of the loss functions.

4. IMPLEMENTATION

In this section we describe our dataset and pre-processing steps. We also give some insight into the training of our model and justification for hyperparameter choices.

4.1 Dataset and Pre-Processing

Our dataset contains songs from the genres Classic, Jazz and Pop. The songs were gathered from various online sources;³ a summary of the properties is shown in Table 1. Note that we excluded symphonies from our Classic, Bach and Mozart datasets due to their complexity and high number of simultaneously playing instruments. We use a train/test split of 90/10. Each song in the dataset can contain multiple instrument tracks and each track can have multiple notes played at the same time. Unless stated otherwise, we select $n_T = 4$ instrument tracks from each song by first picking the tracks with the highest number of

played notes, and from each track we choose the highest voice, meaning picking the highest notes per time step. If a song has fewer than n_T instrument tracks, we pick additional voices from the tracks until we have n_T voices in total. We exclude drum tracks, since they do not have a pitch value. We choose the 16th note as smallest unit. In the most widely used time signature $\frac{4}{4}$ there are 16 16th notes in a bar. 91% of Jazz and Pop songs in our dataset are in $\frac{4}{4}$, whereas for Classic the fraction is 34%. For songs with time signatures other than $\frac{4}{4}$ we still designate 16 16th notes as one bar. All songs are split into samples of one bar and our model auto-encodes one sample at a time. During training we shuffle the songs for each epoch, but keep the bars of a song in the correct order and do not reset the RNN states between samples. Thus, our model is trained on a proper sequence of bars, instead of being confused by random bar progressions.

There are 128 possible pitches in MIDI. Since very low and high pitches are rare and often do not sound pleasing, we only use $n_P = 60$ pitch values ranging from 24 (C_1) to 84 (C_6).

4.2 Model (Hyper-)Parameters

Our model is generally not sensitive to most hyperparameters. Nevertheless we continuously performed local hyperparameter searches based on good baseline models, only varying one hyperparameter at a time. We use the reconstruction accuracy of the pitch roll decoder as evaluation metric. Using Gated Recurrent Units (GRUs) [6] instead of LSTMs increases performance significantly. Using bidirectional GRUs did not improve the results. The pitch roll encoder/decoder uses two GRU layers, whereas the rest uses only one layer. All GRU state sizes as well as the size of the latent space z are set to 256. We use the ADAM optimizer [19] with an initial learning rate of 0.0002. For most layers in our architecture, we found tanh to work better than sigmoid or rectified linear units. We train on batches of size 256. The loss function weights λ_P , λ_I , λ_V and λ_S were set to 1.0, 1.0, 0.1 and 0.1 respectively. λ_P was set to 1.0 to favor high quality note pitch reconstructions over the rest. λ_V was also set to 1.0 because the MSE magnitude is much smaller than the cross entropy loss values.

During our experiments, we realized that high values of β generally lead to very poor performance. We further found that setting the variance of ϵ to the value of $\sigma_\epsilon = 1$, as done in all previous work using VAEs, also has a negative effect. Therefore we decided to treat σ_ϵ as a hyperparameter as well. Figure 2 shows the results of the grid search. σ_ϵ is the variance of the distribution from which the ϵ values for the reparameterization trick are sampled, and is thus usually set to the same value as the variance of the prior. However, especially at the beginning of learning, this introduces a lot of noise that the decoder needs to handle, since the values for μ_z and σ_z , output by the encoder, are small compared to ϵ . We found that by reducing σ_ϵ , we can improve the performance of our model significantly, while being able to use higher values for β . An annealing strategy for both β and σ_ϵ might produce better

³ Pop: www.midiworld.com / Jazz: http://midkar.com/jazz/jazz_01.html / Classic (including Bach, Mozart): www.reddit.com/r/WeAreTheMusicMakers/comments/3ajwe4/

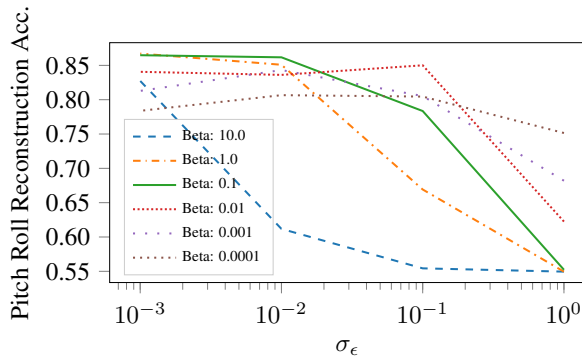


Figure 2. Test reconstruction accuracy of pitch roll for different β and σ_ϵ .

	Pitch		Instrument		Style		Velocity	
	Train	Test	Train	Test	Train	Test	Train	Test
CvJ	0.90	0.85	0.99	0.87	0.98	0.92	0.008	0.029
CvP	0.96	0.88	0.99	0.89	0.96	0.91	0.017	0.036
JvP	0.88	0.80	0.99	0.86	0.94	0.69	0.043	0.048
BvM	0.91	0.75	0.99	0.82	0.94	0.74	0.010	0.033

Table 2. Train and test performance of our final models. The velocity column shows MSE loss values, whereas the rest are accuracies.

results, but we did not test this. In the final models we use $\beta = 0.1$ and $\sigma_\epsilon = 0.01$. Note that during generation we sample z from $\mathcal{N}(0, \sigma_z)$, where σ_z is the empirical variance obtained by feeding the entire training dataset through the encoder. The empirical mean μ_z is very close to zero.

4.3 Training

All models are trained on single GPUs (GTX 1080) until the pitch roll decoder converges. This corresponds to around 400 epochs, or 48 hours. We train one model for each genre/composer pair to make learning easier. This results in four models that we henceforth call CvJ (trained on Classic and Jazz), CvP (Classic and Pop), JvP (Jazz and Pop) and BvM (Bach and Mozart). The train/test accuracies/losses of all final models are shown in Table 2. The columns correspond to the terms in our model’s full loss function (Equation 1).

5. EXPERIMENTAL RESULTS

In this section we evaluate the capabilities of MIDI-VAE. Wherever mentioned, corresponding audio samples can be found on YouTube.⁴

5.1 Style Transfer

To evaluate the effectiveness of MIDI-VAE’s style transfer, we train three separate style evaluation classifiers. The input features are the pitch, velocity and instrument rolls re-

	Train Songs			Test Songs		
	Before	After	Diff.	Before	After	Diff.
CvJ	0.92	0.38	0.54	0.87	0.39	0.48
CvP	0.94	0.43	0.51	0.92	0.45	0.47
JvP	0.72	0.60	0.12	0.72	0.62	0.10
BvM	0.77	0.45	0.32	0.66	0.47	0.19

Table 3. Style transfer performance (ensemble classifier accuracies before and after) between all style pairs.

spectively. The three style classifiers are also combined to output a voting based ensemble prediction. The accuracy of the classifiers is computed as the fraction of correctly predicted styles per bar in a song. We predict the likelihood of the source style *before* and *after* the style change. If the style transfer works, the predicted likelihood of the source style decreases. The larger the difference, the stronger the effect of the style transfer. Note that for all experiments presented in this paper we set the number of styles $k = 2$, that is, one MIDI-VAE model is trained on two styles, e.g., Classic vs. Jazz. Therefore, the style classifier is binary and a reduction in probability of the source style is equivalent to an increase in probability of the target style of the same magnitude. All style classifiers use two-layer GRUs with a state size of 256. Table 3 shows the performance of MIDI-VAE’s style transfer when measured by the ensemble style classifier. We trained a separate MIDI-VAE for each style pair. For each pair of styles we perform a style change on all songs in both directions and average the results. The style transfer works for all models, albeit to varying degrees. In all cases except for JvP, the predictor is even skewed below 0.5, meaning that the target style is now considered more likely than the source style.

Table 4 shows the style transfer results measured by each individual style classifier. We can see that pitch and velocity contribute equally to the style change, whereas instrumentation seems to correlate most with the style. For CvJ and CvP, switching the style heavily changes the instrumentation. Figure 3 illustrates how the instruments of all songs in our Jazz test set are changed when switching the style to Classic. Only few instruments are rarely changed (piano, ensemble, reed), whereas most others are mapped to one or multiple different instruments. The instrument switch between genres with highly overlapping instrumentation (JvP, BvM) is much less pronounced. Classifying style based on the note pitches and velocities of one bar is more difficult, as shown by the “before” accuracies in Table 4, which are generally lower than the ones of the instrument roll based classifier. Nevertheless, the style transfer changes pitch and velocity towards the target style. MIDI-VAE retains most of the original melody, while often changing accompanying instruments to suit the target style. This is generally desirable, since we do not want to change the pitches so thoroughly that the original song cannot be recognized anymore. We provide examples of style transfers on a range of songs from our training and test sets on YouTube (see *Style transfer songs*).

⁴ <https://goo.gl/vb8Yrh>

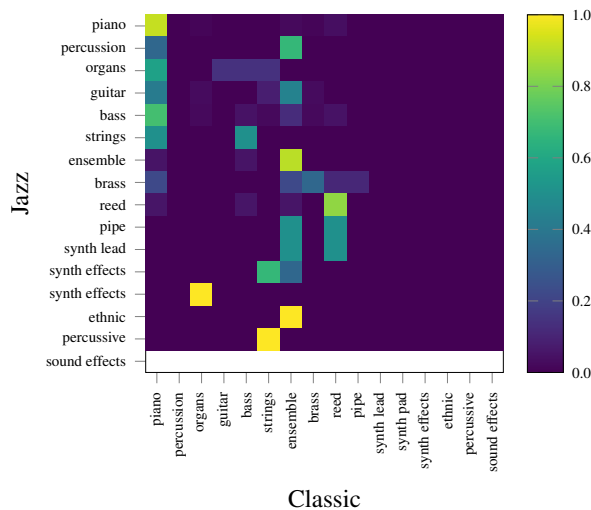


Figure 3. The matrix visualizes how the instruments are changed when switching from Jazz to Classic, averaged over all Jazz songs in the test set.

	Pitch		Velocity		Instrument	
	Bf.	Af.	Bf.	Af.	Bf.	Af.
CvJ Test	0.77	0.66	0.67	0.57	0.90	0.20
CvP Test	0.77	0.67	0.71	0.60	0.91	0.27
JvP Test	0.65	0.63	0.67	0.64	0.67	0.55
BvM Test	0.55	0.47	0.60	0.49	0.64	0.47

Table 4. Average before and after classifier accuracies for all classifiers (pitch/instrument/velocity) for the test set.

5.2 Latent Space Evaluation

Figure 4 shows a t-SNE [22] plot of the latent vectors for all bars of 20 Jazz and 20 Classic pieces. The darker the color, the more “jazzy” or “classical” a song is according to the ensemble style classifier. The genres are well separated, and most songs have all their bars clustered closely together (likely thanks to the instrument roll being constant). Some classical pieces are bleeding over into the Jazz region and vice versa. As can be seen from the light color, the ensemble style classifier did not confidently assign these pieces to either style.

We further perform a sweep over all 256 latent dimensions on randomly sampled bars to check whether changing one dimension has a measurable effect on the generated music. We define 27 metrics, among which are total number of (held) notes, mean/max/min/range of (specific or all) pitches/velocities, and style changes. Besides the obvious dimensions where the style classifier is attached, we find that some dimensions correlate with the total number of notes played in a song, the highest pitch in a bar, or the occurrence of a specific pitch. The changes can be seen when plotting the pitches, but are difficult to hear. Furthermore, the dimensions are very entangled, and changing one dimension has multiple effects. Higher values for $\beta \in \{1, 2, 3\}$ slightly improve the disentanglement of la-

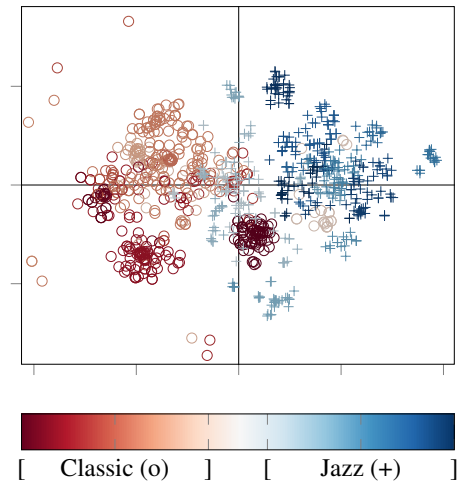


Figure 4. t-SNE plot of latent vectors for bars from 20 Jazz and Classic songs. Bars from the same song were given the same color. Lighter colors mean that the ensemble style classifier was less certain in its prediction.

tent dimensions, but strongly reduce reconstruction accuracy (see Figure 2). We added samples to YouTube to show the results of manipulating individual latent variables.

5.3 Generation and Interpolation

MIDI-VAE is capable of producing smooth interpolations between bars. This allows us to generate medleys by connecting short pieces from our dataset. The interpolated bars form a musically consistent bridge between the pieces, meaning that, e.g., pitch ranges and velocities increase when the target bar has higher pitch or velocity values. We can also merge entire songs together by linearly interpolating the latent vectors for two bar progressions, producing interesting mixes that are surprisingly fun to listen to. The original songs can sometimes still be identified in the mixtures, and the resulting music sounds harmonic. We again uploaded several audio samples to YouTube (see *Medleys*, *Interpolations* and *Mixtures*).

6. CONCLUSION

We introduce MIDI-VAE, a simple but effective model for performing style transfer between musical compositions. We show the effectiveness of our method on several different datasets and provide audio examples. Unlike most existing models, MIDI-VAE incorporates both the dynamics (velocity and note durations) and instrumentation of music. In the future we plan to integrate our method into a hierarchical model in order to capture style features over longer time scales and allow the generation of larger pieces of music. To facilitate future research on style transfer for symbolic music, and sequence tasks in general, we make our code and data publicly available.⁵

⁵<https://github.com/brunnergino/MIDI-VAE>

7. REFERENCES

- [1] Midi association, the official midi specifications. <https://www.midi.org/specifications>. Accessed: 01-06-2018.
- [2] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.
- [3] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.
- [4] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation—a survey. *arXiv preprint arXiv:1709.01620*, 2017.
- [5] Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Jonas Wiesendanger. JamBot: Music theory aware chord based generation of polyphonic music with LSTMs. In *29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2017.
- [6] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar; A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734, 2014.
- [7] Hang Chu, Raquel Urtasun, and Sanja Fidler. Song from PI: A musically plausible network for pop music generation. *CoRR*, abs/1611.03477, 2016.
- [8] Ching-Hua Chuan and Dorien Herremans. Modeling temporal tonal relations in polyphonic music through deep networks with a novel image-based representation. 2018.
- [9] David Cope. Experiments in music intelligence (EMI). In *Proceedings of the 1987 International Computer Music Conference, ICMC 1987, Champaign/Urbana, Illinois, USA, August 23-26, 1987*, 1987.
- [10] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.
- [11] Douglas Eck and Juergen Schmidhuber. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 103, 2002.
- [12] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1068–1077, 2017.
- [13] Jose D. Fernández and Francisco J. Vico. AI methods in algorithmic composition: A comprehensive survey. *J. Artif. Intell. Res.*, 48:513–582, 2013.
- [14] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2414–2423. IEEE, 2016.
- [15] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1362–1371, 2017.
- [16] Dorien Herremans, Ching-Hua Chuan, and Elaine Chew. A functional taxonomy of music generation systems. *ACM Comput. Surv.*, 50(5):69:1–69:30, 2017.
- [17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.
- [18] Daniel D. Johnson. Generating polyphonic music using tied parallel networks. In *Computational Intelligence in Music, Sound, Art and Design - 6th International Conference, EvoMUSART 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings*, pages 128–143, 2017.
- [19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [20] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.
- [21] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 385–395, 2017.
- [22] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

- [23] Iman Malik and Carl Henrik Ek. Neural translation of musical style. *CoRR*, abs/1708.03535, 2017.
- [24] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron C. Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *CoRR*, abs/1612.07837, 2016.
- [25] Olof Mogren. C-RNN-GAN: continuous recurrent neural networks with adversarial training. *CoRR*, abs/1611.09904, 2016.
- [26] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network. *CoRR*, abs/1805.07848, 2018.
- [27] Michael C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connect. Sci.*, 6(2-3):247–280, 1994.
- [28] Colin Raffel and Daniel PW Ellis. Intuitive analysis, creation and manipulation of midi data with pretty_midi. In *15th International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, pages 84–93, 2014.
- [29] Adam Roberts, Jesse Engel, and Douglas Eck. Hierarchical variational autoencoders for music. In *NIPS Workshop on Machine Learning for Creativity and Design*, 2017.
- [30] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844, 2017.
- [31] Peter M Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43, 1989.
- [32] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, page 125, 2016.
- [33] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6309–6318, 2017.
- [34] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 324–331, 2017.
- [35] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 2852–2858, 2017.
- [36] Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. Adversarially regularized autoencoders for generating discrete structures. *CoRR*, abs/1706.04223, 2017.
- [37] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2242–2251, 2017.

UNDERSTANDING A DEEP MACHINE LISTENING MODEL THROUGH FEATURE INVERSION

Saumitra Mishra, Bob L. Sturm, Simon Dixon

Centre for Digital Music, School of Electronic Engineering and Computer Science

Queen Mary University of London, United Kingdom

{saumitra.mishra, b.sturm, s.e.dixon}@qmul.ac.uk

ABSTRACT

Methods for interpreting machine learning models can help one understand their global and/or local behaviours, and thereby improve them. In this work, we apply a global analysis method to a machine listening model, which essentially inverts the features generated in a model back into an interpretable form like a sonogram. We demonstrate this method for a state-of-the-art singing voice detection model. We train up-convolutional neural networks to invert the feature generated at each layer of the model. The results suggest that the deepest fully connected layer of the model does not preserve temporal and harmonic structures, but that the inverted features from the deepest convolutional layer do. Moreover, a qualitative analysis of a large number of inputs suggests that the deepest layer in the model learns a decision function as the information it preserves depends on the class label associated with an input.

1. INTRODUCTION

Deep neural networks (DNNs) are state-of-the-art in numerous machine learning applications. This success is due to their high expressive power and strong generalisation capability [10]. DNNs acquire these capabilities automatically through training over large amounts of data and tuning of millions of parameters. Despite their success, DNNs remain “black-boxes” as we know very little about the process by which they form their predictions.

Recent research has highlighted problems associated with DNNs. For example, researchers have demonstrated that attacking these models with carefully generated inputs, called “adversarial examples”, changes their predictions [11, 15, 44]. Such behaviour may be dangerous to a system (e.g., autonomous vehicle) if its decision making depends on DNN predictions [32]. Moreover, like shallow machine learning models, a DNN may exploit confounders in a dataset and behave correctly for the wrong reasons. Such behaviour limits the performance of a model in the

real world where such confounders are absent. Thus, there is an urgent need to bring interpretability to these black-box models, i.e. to understand the behaviour of a DNN [4].

Researchers have attempted to analytically [33, 48] and empirically explain the behaviour of DNNs. In this work, we focus on understanding these models empirically using post-hoc visualisation methods [27, 31]. We can broadly classify such methods into two categories: (1) methods that explain model predictions (local analysis); and (2) methods that explain a model (global analysis). Local analysis uses variants of *sensitivity analysis* (e.g., gradient-based sensitivity analysis) to generate attribution maps that highlight the input dimensions [39, 40, 43] or input regions (groups of contiguous dimensions) [35, 47] in favour of (or against) a prediction. Such analysis is useful but may result in inconsistent [16] and uninterpretable (noisy) explanations [41]. Although some local explanation methods can generate cleaner visualisations, they depend on the type of non-linearity [42] or network architecture [38, 47] and thus are not generalisable.

In another direction, the global analysis of DNNs aims for an insight that generalises across input instances. For example, irrespective of the class label associated with an input image, the shallow layers of image content recognition models show sensitivity to low-level structures, e.g., edges and gradients [47]. There exist several methods for global analysis. For example, *activation maximisation* aims to synthesise examples in the input space (e.g., pixels) that maximally activate a specific neuron [9, 29, 40, 46] or layer [28] in a model. Similarly, *feature inversion* aims to highlight the input content (features) preserved by any layer in a DNN model by inverting the corresponding feature [6, 23].

In this work, we apply feature inversion to a machine listening model that classifies an input audio frame (or excerpt) into predefined classes. Previous work in the analysis of deep machine listening models has focused both on local and global analysis. The methods to generate local explanations for model predictions use bin-level [1] or region-level [26] attribution maps. On the other hand, Dieleman et al. [3] globally analyse a music autotagging model by visualising filters in the first convolutional layer. Similarly, in [36] the authors globally analyse a scaled-down version of their deep onset detector by visualising the most activated feature maps and their corresponding filter kernels. Our method differs from these global analy-



© Saumitra Mishra, Bob L. Sturm, Simon Dixon. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Saumitra Mishra, Bob L. Sturm, Simon Dixon. “Understanding a Deep Machine Listening Model through Feature Inversion”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

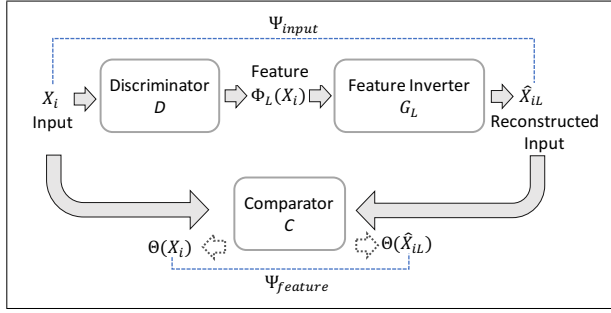


Figure 1: Functional block diagram of our feature inversion method. The method inverts a feature $\Phi_L(\mathbf{x}_i)$ from a layer L by training a feature inverter G_L that jointly minimises the input space loss Ψ_{input} and feature space loss $\Psi_{feature}$. Φ_L and Θ are the representation functions of a discriminator D and comparator C , respectively.

sis methods as it neither limits the analysis only to shallow layers nor puts any restriction on the depth of a model.

We demonstrate our method for a state-of-the-art singing voice detection (SVD) model that classifies an input mel spectrogram excerpt into two classes: ‘vocal’ and ‘non-vocal’ [37]. We first train up-convolutional neural networks [7], one per layer of the SVD model, to invert the features generated by it. We then quantify the performance of the inversion models (we call them “feature inverters”) by calculating the normalised reconstruction error (NRE) that [23] defines as the normalised Euclidean distance between an input and its inverted representation. The results demonstrate that NRE is largest for a feature inverter that inverts the deepest layer (the last fully connected layer) in the SVD model and decreases for feature inverters that invert features from shallow layers. Finally, we qualitatively analyse the inverted features for both classes (vocal and non-vocal) to understand the preserved input content at each layer of the SVD model. Similarly, we analyse the inverted features for inputs selected from different datasets to test whether the conclusions from one dataset generalise to the other. The experimental code and results are available online.¹

2. FEATURE INVERSION

Feature inversion aims to map the feature generated at any layer of a DNN back to a plausible input. Each layer in a DNN maps an input feature to an output feature and in the process ignores the input content that does not seem relevant to the classification task. Thus, the inversion of a feature from any layer of a DNN will highlight the input content preserved by that layer.

2.1 Prior Work

Mahendran et al. [23] and Dosovitskiy et al. [6] apply feature inversion to analyse the global behaviour of convolutional neural networks (CNNs). Mahendran et al. [23, 24]

invert the features from AlexNet [18] (a CNN for image recognition). Their work demonstrates that the inverted features from the deepest convolutional layer in AlexNet are visually similar (preserve the spatial layout and colour) to the input image. They also demonstrate that although the reconstructions from fully connected layers are visually poor, they still depict the presence of high-level features (e.g., the facial features of an animal). Their work also highlights the invariances captured by the AlexNet layers. For example, the inverted representations from the deepest layer in the model (a fully connected layer) depict an object at different locations, orientations and scales.

The method introduced by Mahendran et al. [23] generates an inverted representation $\mathbf{x}_{iL}^* \in \mathbb{R}^n$ from an L^{th} layer feature by iteratively minimising the feature space loss between an input image $\mathbf{x}_i \in \mathbb{R}^n$ and an intermediate representation $\mathbf{x}_{iL}' \in \mathbb{R}^n$. The method starts with a randomly sampled \mathbf{x}_{iL}' and in each iteration updates it by calculating the gradient of the loss function at \mathbf{x}_{iL}' . Formally, given a CNN with representation function $\Phi_L : \mathbb{R}^n \rightarrow \mathbb{R}^d$ that maps an n -dimensional input to a d -dimensional feature $\Phi_L(\mathbf{x}_i)$ at a layer L , the method inverts $\Phi_L(\mathbf{x}_i)$ by solving

$$\mathbf{x}_{iL}^* = \arg \min_{\mathbf{x}_{iL}'} \|\Phi_L(\mathbf{x}_{iL}') - \Phi_L(\mathbf{x}_i)\|^2 + \alpha f(\mathbf{x}_{iL}') \quad (1)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a regularisation function (a natural image prior) that limits the search to realistic images and α is a scaling constant. Regularisation is needed since an unrestricted search may output fooling examples [30] that cause high activations to a neuron (or a layer), but do not possess features found in natural images. The method by Mahendran et al. [23, 24] has two major limitations: (1) hand-crafting a prior is challenging as for some inputs (e.g., images, audio) defining the constituents of a real input is difficult; and (2) the method needs to solve Eq. 1 for every new feature it needs to invert.

The feature inversion method proposed by Dosovitskiy et al. [6] tackles both the above issues and demonstrates visually improved reconstructions even for the fully connected layers of AlexNet. The method trains another neural network, an up-convolutional neural network (feature inverter) [7], to invert the features of a DNN. The method trains a feature inverter by minimising the input space loss Ψ_{input} , defined as the squared Euclidean distance between an input image and its inverted representation. Although this method learns a natural image prior implicitly during training and is expensive only at the training time, the inverted representations are blurry for all the layers. The reason behind this is the way a feature inverter inverts a feature. A forward pass through AlexNet (or any DNN) maps several inputs to the same feature. Thus, to invert a feature, a feature inverter generates an input that is an average of all the inputs that AlexNet maps to the given feature. This averaging effect results in blurry reconstructions.

2.2 Our Method

Fig. 1 provides an overview of our method. We use the approach of Dosovitskiy et al. [6], but modify its loss

¹ <https://github.com/saum25/ISMIR-2018>

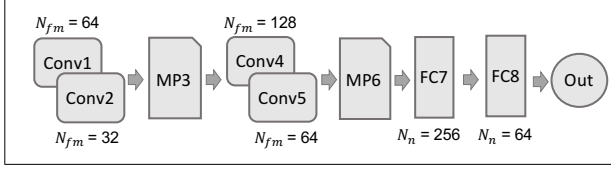


Figure 2: The architecture of the singing voice detection model proposed by Schlüter et al. [37]. N_{fm} denotes the number of feature maps in the output of a convolutional layer. N_n denotes the number of neurons in a fully connected layer. Conv: convolutional layer, MP: max-pooling layer, FC: fully connected layer, Out: output layer.

function to reduce the effect of input averaging. Recent works [5, 14] demonstrate that minimising loss in the perceptual space helps to reduce the over-smoothness problem for image generation models. We extend this idea to machine listening. Thus, in addition to the input space loss Ψ_{input} , our method also calculates the feature space loss $\Psi_{feature}$. We define total loss Ψ as:

$$\Psi = \lambda_{input} \Psi_{input} + \lambda_{feature} \Psi_{feature} \quad (2)$$

where λ_{input} and $\lambda_{feature}$ weight the losses of the input space and feature space. Thus, our method trains a feature inverter G_L to invert a feature $\Phi_L(\mathbf{x}_i)$ by jointly minimising the input space and feature space losses. To evaluate $\Psi_{feature}$, we use the approach from [5] where the authors use a comparator C to map an input \mathbf{x}_i and its inverted representation $\hat{\mathbf{x}}_{iL}$ to the feature space. A comparator is a pre-trained discriminative model that may or may not be of the same depth as the discriminator D (the model whose features we are inverting). We can even use D as a comparator by extracting feature vectors at a layer of D (e.g., Dosovitskiy et al. [5] use the deepest convolutional layer of AlexNet as a comparator for inverting AlexNet).

Formally, given an input excerpt $\mathbf{x}_i \in \mathbb{R}^n$ and a representation function $\Phi_L : \mathbb{R}^n \rightarrow \mathbb{R}^d$ that maps \mathbf{x}_i to a d -dimensional feature $\Phi_L(\mathbf{x}_i)$ at a layer L of a discriminator D , our method trains a feature inverter G_L that maps $\Phi_L(\mathbf{x}_i)$ to an inverted representation $\hat{\mathbf{x}}_{iL} \in \mathbb{R}^n$. In order to do that, the method calculates Ψ_{input} and $\Psi_{feature}$. Given a comparator C with a representation function $\Theta : \mathbb{R}^n \rightarrow \mathbb{R}^{d'}$, we define $\Psi_{feature}$ as the squared Euclidean distance between $\Theta(\mathbf{x}_i)$ and $\Theta(\hat{\mathbf{x}}_{iL})$, where $\hat{\mathbf{x}}_{iL}$ is an inverted representation for an input \mathbf{x}_i at layer L and d' is the dimensionality of the feature space for C . Similarly, we define Ψ_{input} as the squared Euclidean distance between \mathbf{x}_i and $\hat{\mathbf{x}}_{iL}$. The method trains an up-convolutional neural network $G_L(\Phi_L(\mathbf{x}_i); \mathbf{w})$ with parameters \mathbf{w} by the optimisation

$$\begin{aligned} \mathbf{w}^* = \arg \min_{\mathbf{w}} & \sum_i (\|\mathbf{x}_i - G_L\{\Phi_L(\mathbf{x}_i); \mathbf{w}\}\|^2 \\ & + \|\Theta(\mathbf{x}_i) - \Theta(G_L\{\Phi_L(\mathbf{x}_i); \mathbf{w}\})\|^2) + \beta \|\mathbf{w}\|^2 \end{aligned} \quad (3)$$

where $\beta > 0$ is the regularisation constant. Once we train G_L , we can invert any feature $\Phi_L(\mathbf{x}_i)$ by a forward pass

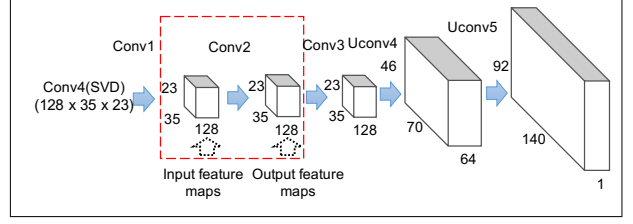


Figure 3: Feature inverter architecture for the Conv4 layer of the SVD model. The highlighted components represent the ‘Conv2’ convolutional layer and its input and output feature maps. Uconv: up-convolutional layer, Conv: convolutional layer.

through G_L :

$$\hat{\mathbf{x}}_{iL} = G_L(\Phi_L(\mathbf{x}_i); \mathbf{w}^*) \quad (4)$$

3. INVERTING THE FEATURES OF A DEEP SINGING VOICE DETECTOR

We now demonstrate our feature inversion method from Section 2 for a state-of-the-art SVD model [37]. We first introduce the SVD model and then explain the architectures and training details of our feature inverters. Finally, we evaluate the performance of the feature inverters on two SVD datasets.

3.1 The Deep Singing Voice Detection Model

Singing voice detection is an audio segmentation problem where the task is to classify an input audio frame (excerpt) into one of the two categories: singing voice with or without instrumental music (‘vocal’) or instrumental music without singing voice (‘non-vocal’). There exist several methods for singing voice detection. Some use hand-crafted features to train shallow classifiers [20, 22, 34], while others jointly optimise the feature extraction and classification steps using deep learning [19, 21, 37].

Schlüter et al. [37] train an SVD model using a CNN and seven data augmentation techniques. Their model achieves state-of-the-art performance on public benchmark datasets.² Fig. 2 depicts the architecture of their 8-layered SVD model. Each convolutional layer performs convolution using 3×3 filters with 1×1 stride and no zero padding. The two max-pooling layers perform pooling with 3×3 stride and no zero padding. The input to the model is a mel spectrogram of about 1.6sec (115 frames). The model was trained on the Jamendo dataset [34]. Jamendo is a dataset of pop music songs and it consists of non-overlapping training, validation and test subsets with 61, 16 and 16 audio files, respectively. The model uses the auxiliary data (57 frames on each sides of the centre frame) as context to classify the centre frame in an input.

3.2 Feature Inverter Architectures

We train up-convolutional neural networks to invert the features generated by the above SVD model. We design

² <https://github.com/f0k/ismir2015>

Layer	Input Shape	Units	Output Shape
FC1	256×1	256	256×1
Reshape	256×1	-	$16 \times 4 \times 4$
Uconv2	$16 \times 4 \times 4$	64	$64 \times 8 \times 8$
Conv3	$64 \times 8 \times 8$	64	$64 \times 8 \times 8$
Uconv4	$64 \times 8 \times 8$	32	$32 \times 16 \times 16$
Conv5	$32 \times 16 \times 16$	32	$32 \times 16 \times 16$
Uconv6	$32 \times 16 \times 16$	16	$16 \times 32 \times 32$
Conv7	$16 \times 32 \times 32$	16	$16 \times 32 \times 32$
Uconv8	$16 \times 32 \times 32$	8	$8 \times 64 \times 64$
Conv9	$8 \times 64 \times 64$	8	$8 \times 64 \times 64$
Uconv10	$8 \times 64 \times 64$	1	$1 \times 128 \times 128$

Table 1: Feature inverter architecture to invert the FC7 layer of the SVD model. Input and output shape dimensions are ordered as the number of channels \times time \times frequency. Uconv: up-convolutional layer, Conv: convolutional layer, FC: fully connected layer. Units refer to the number of neurons in a fully connected layer or the number of filters in a convolutional layer.

Inv-idx	Inv-inp	Inv-depth	Inv-nconv
FC8	64×1	11	4
FC7	256×1	10	4
MP6	$64 \times 11 \times 7$	5	1
Conv5	$64 \times 33 \times 21$	5	3
Conv4	$128 \times 35 \times 23$	5	3
MP3	$32 \times 37 \times 25$	6	4
Conv2	$32 \times 111 \times 76$	4	4
Conv1	$64 \times 113 \times 78$	2	2

Table 2: Architectural overview of the feature inverters for all the layers in the SVD model. Inv-idx: SVD layer a feature inverter inverts, Inv-inp: input to a feature inverter (number of channels \times time \times frequency), Inv-depth: number of layers in a feature inverter, Inv-nconv: number of convolutional layers in a feature inverter. Conv: convolutional layer, FC: fully connected layer and MP: max-pooling layer.

two categories of architectures, one to invert the fully connected (FC) layers and the other to invert the convolutional (Conv) and max-pooling (MP) layers of the SVD model.³ The architecture of inversion models in [6] inspires the design of our feature inverters, but we adapt the architectures to suit the SVD model. A majority of feature inverters need to perform the upsampling (unpooling) operation that is an approximate inverse of the max-pooling operation done in the SVD model. In order to perform unpooling and convolution in a single step, we use up-convolutional layers (Uconv) with 4×4 filters and 2×2 stride. This configuration of Uconv layers upsamples an input feature map by 2 [7]. The number of such layers depends on the dimensionality of the layer we are inverting. For example, the

³ “inverting a layer” is another way to refer to the inversion of the features generated by a layer.

feature inverter to invert the 256-dimensional FC7 layer uses 5 Uconv layers (Table 1), while the feature inverter to invert the Conv4 layer uses two Uconv layers (Fig. 3). The feature inverters for the Conv1 and Conv2 layers in the SVD model do not use Uconv layers as for them the model generates features without using the max-pooling layer.

We increase the capacity of the feature inverters by adding convolutional layers; either after every up-convolutional layer (for inverting an FC layer) or before the first up-convolutional layer (for inverting a Conv or MP layer). We empirically decide the number of convolutional layers for each feature inverter. The convolutional layers perform convolution using 3×3 filters with 1×1 stride and improve the visual appearance of the reconstructions [8]. Table 2 provides details about the depth and the number of Conv layers in each feature inverter. All the layers use exponential linear unit (ELU) non-linearity given by $y(x) = x$ if $x > 0$, otherwise $e^x - 1$ [2]. The network uses batch normalisation layers [13] to make sure the input to each layer follows a standard normal distribution. Except for the Conv1 and Conv2 layers, each feature inverter generates an inverted representation with a larger spatial size and later trims it to match the input excerpt size (115×80). The feature inverters for the Conv1 and Conv2 layers generate an inverted representation of the same shape as input by symmetrically padding the missing dimensions.

3.3 Training of the Feature Inverters

We train one feature inverter per layer of the SVD model. We train a feature inverter using mel spectrogram excerpts of about 1.6 sec that we extract from the Jamendo training dataset. We show one such sample in Fig. 4. We generate excerpts with a hop size of 10 frames (140 ms). Thus, we train each feature inverter using a data set of about 100k features. We do not use any data augmentation techniques. In order to prevent overfitting, we run the optimisation to a fixed number of weight updates (30 epochs) and select a feature inverter giving the lowest loss on the validation subset. We use the Conv5 layer of the SVD model as the comparator, i.e., we encode the mel spectrogram and the inverted representation using Conv 5. We initialise the feature inverter weights using the He normal initialisation method [12]. In each iteration, for a mini-batch of 32 randomly selected excerpts, the training objective jointly minimises the feature and input space losses and updates the change in parameters using ADAM [17]. We set the scaling factors λ_{input} and $\lambda_{feature} = 1$ (Eq. 2). We start training with an initial learning rate of 0.001 and decay it by 0.5 when the training loss does not change for 2 consecutive epochs. The training procedure performs regularisation using L_2 weight decay and sets $\beta = 1e - 4$ (Eq. 3).

3.4 Quantitative Evaluation of the Feature Inverters

We train eight feature inverters using the Jamendo training dataset and the architectures and training methodology discussed above. We evaluate the performance of each feature inverter on an evaluation set of 128 mel spectrogram excerpts. We build the evaluation set by randomly selecting

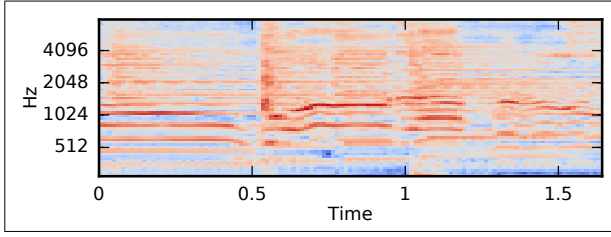


Figure 4: Sample mel spectrogram excerpt from the Jamendo dataset. This excerpt belongs to the vocal class.

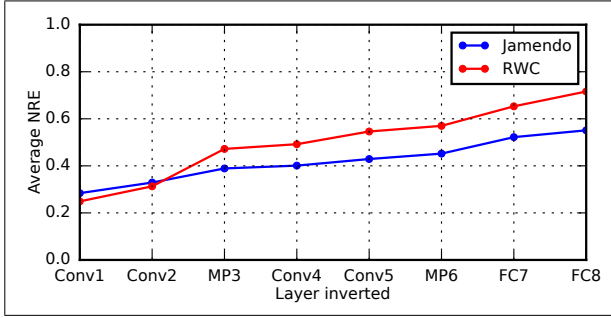


Figure 5: Performance evaluation of the feature inverters. The plot depicts the change in average normalised reconstruction error (NRE) as a feature inverter inverts different layers in the SVD model.

8 excerpts from each of the 16 audio files in the Jamendo test dataset. We quantify the performance of feature inverters by calculating the average normalised reconstruction error (NRE) for each feature inverter on the evaluation dataset. [23] defines NRE as:

$$NRE = \|\mathbf{x}_i - \hat{\mathbf{x}}_{iL}\| / N_c \quad (5)$$

where N_c is a normalising constant computed from the average pairwise Euclidean distance between excerpts in the evaluation set.

We also evaluate the feature inverters on the RWC dataset [25] to understand whether the results of the quantitative evaluation on Jamendo extend to the RWC dataset. The RWC dataset for singing voice detection is a public benchmark dataset that contains a collection of 100 pop music songs, but unlike Jamendo, there is no partitioning into separate subsets. Thus, to evaluate our models we first build an RWC test dataset by randomly selecting 20 audio files from a set of 100 and use them to build an evaluation dataset of 160 randomly selected excerpts (8 excerpts per audio files). Moreover, in order to evaluate the feature inverters on a larger evaluation dataset, we randomly sample 10 different evaluation sets, calculate the average NRE for each and later take an average. Thus, effectively we evaluate our feature inverters on an evaluation dataset of size 1280 (Jamendo) and 1600 (RWC) excerpts.

Fig. 5 shows the results of the evaluation. For both datasets, the reconstruction error is largest for the deepest layer in the SVD model (FC8) and decreases for representations inverted from shallower layers. This is predictable as the dimensionality of the features in shallow

layers is larger than in deep layers, making it easier to invert them. For instance, the dimensionality reduction of features from MP6 to FC7 is about 19 times, compressing a 4928-dimensional feature to 256 dimensions. Similarly, we see a large increase in the average NRE between the feature inverters for the Conv2 and MP3 layers. This likely occurs due to max-pooling operation that compresses feature dimensionality by 9 times between the two layers.

The results also depict that the feature inverters have larger reconstruction error on the RWC dataset at all but two layers. This is expected since both the discriminator (the SVD model) and the feature inverters are trained on the Jamendo dataset. One possible explanation for the comparable average NRE of the Conv1 and Conv2 layers is that these shallow layers of the model are learning generalisable features [47]. This becomes less so at deeper layers, where features are likely tuned to specific traits of the training data.

We also compare the performance of the feature inversion method we use in this work (we call it ' M_{joint} ') with a baseline method (we call it ' M_{input} ') that trains feature inverters using image loss only. We train and test the feature inverters of M_{input} on the Jamendo dataset. We find that across all the layers of the SVD model, the average NRE of the feature inverters using M_{input} is either similar or slightly lower than for those using M_{joint} . Such a behaviour is predictable as M_{input} aims to only minimise the input reconstruction loss, while M_{joint} aims to jointly optimise both the loss functions, which may or may not result in lower NRE [5]. The benefit of using M_{joint} comes from the generation of inverted representations that are perceptually closer to an input, a property that is challenging for M_{input} to achieve.

4. QUALITATIVE ANALYSIS OF THE INVERTED FEATURES

Fig. 6 shows visualisations for each layer of the SVD model. We generate these visualisations by selecting four inputs, two from each dataset (Jamendo and RWC), in which one belongs to each of the two classes (vocal and non-vocal). We then use the feature inverters to invert the features extracted by the SVD model from each input. The results provide some insights into the model behaviour. For example, reconstructions from the FC8 layer suggest that FC8 does not retain the harmonic structures present in the inputs. Moreover, it appears that this layer preserves either the high frequency or the low-frequency content of an input. Similarly, FC8 does not preserve any temporal information (musical onset locations) present in the inputs. Interestingly, for a large number of cases (in addition to these four inputs), we found a clear demarcation between the vocal and the non-vocal class visualisations from this layer. We find that for the vocal class, energy appears in higher frequencies while for the non-vocal class energy appears in lower frequencies. These visualisations suggest that the SVD model learns a class-decision function in this layer.

Similarly, reconstructions from the FC7 layer suggest that the layer preserves some harmonic content and ap-

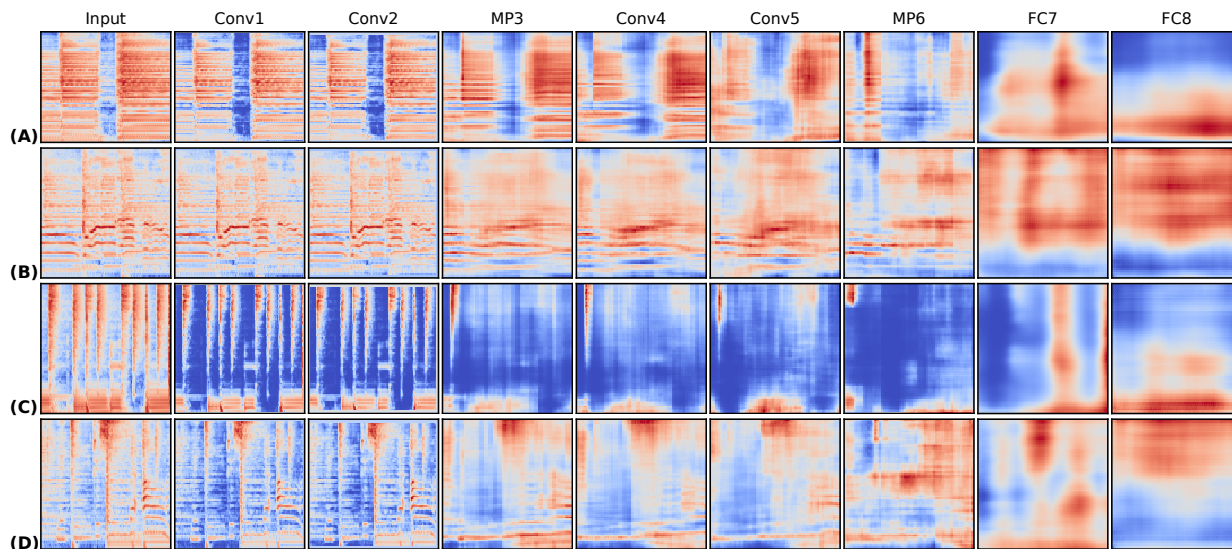


Figure 6: Feature inversion from successive layers of the SVD model. Each row corresponds to one input excerpt: (A), (B) are respectively non-vocal and vocal excerpts from “03 - Say me Good Bye.mp3” in the Jamendo test dataset. Similarly, (C) and (D) are respectively non-vocal and vocal excerpts from “RWC- MDB-P-2001-M04/5 Audio Track.aiff” in the RWC test dataset. Columns contain mel spectrograms of (from left to right): the input signal then inverted representations from successive SVD model layers (as labelled). The visualisations highlight how the model ignores the input content as it forms higher-level representations. Inversions of shallow layers resemble the input, but the reconstruction quality reduces for deeper layers. Conv: convolutional layer, MP: max-pooling layer, FC: fully connected layer.

proximate onset locations of the inputs. But, there are some deviations from this behaviour. For instance, in Fig. 6 row B, the harmonic structures are less evident. Similarly, for the input in row C, the feature inverter at FC7 is unable to reconstruct all the harmonic and temporal content present in the input. This may be due to the fact that we do not train the feature inverters on RWC, thus the reconstruction error is higher for this input, resulting in poor reconstruction.

We find that reconstructions from the deepest convolutional layer of the model contain more information than those from the two fully connected layers. For both inputs from Jamendo (Fig 6A-B), the model preserves much of the input content (e.g., the reconstructions capture the harmonic structure and approximately align the temporal boundaries with the input). This confirms the quantitative results of model inversion for Conv5 and FC7 layers, where we show that the average NRE is about 18% less for Conv5. The visualisations for the RWC excerpts (Fig. 6C-D) report similar results. Finally, reconstructions from all the other layers follow a similar pattern. Moving toward shallower layers, they become visually similar to the input, increasingly showing the presence of finer harmonics and temporal structures. Moreover, the inversions from Conv1 and Conv2 are very close to the respective inputs. This suggests that the filters of the first 2 convolutional layers act as a bijective map, e.g., performing an invertible frequency transform. Moreover, the visualisations from deeper layers in the model are more blurry than from shallow layers. This suggests that deeper layers capture more invariances from data than shallow layers.

5. CONCLUSION AND FUTURE WORK

In this work, we applied a model analysis method called feature inversion to a state-of-the-art singing voice detection model. Feature inversion helped to understand the global behaviour of the SVD model by visualising the information preserved by any layer in the model. We trained up-convolutional neural networks to invert the features of the model. We quantitatively analysed the feature inverters for each layer in the model to understand the change in input reconstruction error across different layers. We found that the average NRE changes by about 15% for Jamendo between the MP6 and FC7 layers due to high dimensionality reduction. Moreover, we qualitatively visualised the inverted representations to understand the input content preserved by any layer in the model. We found that the deepest fully connected layer does not retain any of the temporal or harmonic structures present in an input. We also found that for a large number of inputs this layer seems to learn a decision function that depends on the class associated with an input. Qualitative analysis of other layers revealed that the FC7 layer preserves some harmonic and temporal information of an input while the reconstructions from the Conv5 layer are visually similar to the input.

In our future work, we plan to improve the loss function by adding adversarial loss [5] that helps to generate realistic inverted representations that are close to one of the classes. This facilitates sonification of inverted representations, giving more insights into the model behaviour. Moreover, we plan to extend the analysis by applying feature inversion to different architectures of the SVD model and also to different machine listening tasks.

6. ACKNOWLEDGEMENTS

We would like to thank Jan Schlüter for discussions and sharing his implementation of the SVD model. We also thank the anonymous reviewers for their valuable comments and suggestions.

7. REFERENCES

- [1] K. Choi, G. Fazekas, and M. B. Sandler. Explaining Deep Convolutional Neural Networks on Music Classification. *arXiv e-prints*, arXiv:1607.02444, 2016.
- [2] DA. Clevert, T. Unterthiner, and S. Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *ArXiv e-prints*, arXiv:1511.07289, 2015.
- [3] S. Dieleman and B. Schrauwen. End-to-End Learning for Music Audio. In *Proc. ICASSP*, 2014.
- [4] F. Doshi-Velez and B. Kim. Towards a Rigorous Science of Interpretable Machine Learning. *arXiv e-prints*, arXiv:1702.08608, 2017.
- [5] A. Dosovitskiy and T. Brox. Generating Images with Perceptual Similarity Metrics Based on Deep Networks. In *Proc. NIPS*, 2016.
- [6] A. Dosovitskiy and T. Brox. Inverting Visual Representations with Convolutional Networks. In *Proc. CVPR*, 2016.
- [7] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to Generate Chairs with Convolutional Neural Networks. In *Proc. CVPR*, 2015.
- [8] A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, and T. Brox. Learning to Generate Chairs, Tables and Cars with Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
- [9] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualising Higher-Layer Features of a Deep Network. Technical Report 1341, University of Montreal, June 2009.
- [10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
- [11] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. In *Proc. ICLR*, 2015.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proc. ICCV*, 2015.
- [13] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proc. ICML*, 2015.
- [14] J. Snell and K. Ridgeway and R. Liao and B.D. Roads and M. C. Mozer and R. S. Zemel. Learning to Generate Images with Perceptual Similarity Metrics. In *Proc. ICIP*, 2017.
- [15] C. Kereliuk, Bob L. Sturm, and J. Larsen. Deep Learning, Audio Adversaries, and Music Content Analysis. In *Proc. WASPAA*, 2015.
- [16] PJ. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim. The (Un)reliability of Saliency Methods. In *Proc. NIPS Workshop*, 2017.
- [17] D. Kingma and B. Jimmy. Adam: A Method for Stochastic Optimization. In *Proc. ICLR*, 2015.
- [18] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *Proc. NIPS*, 2012.
- [19] S. Leglaive, R. Hennequin, and R. Badeau. Singing Voice Detection with Deep Recurrent Neural Networks. In *Proc. ICASSP*, 2015.
- [20] B. Lehner, R. Sonnleitner, and G. Widmer. Towards Light-Weight, Real-Time-Capable Singing Voice Detection. In *Proc. ISMIR*, 2013.
- [21] B. Lehner, G. Widmer, and S. Bock. A Low-Latency, Real-Time-Capable, Singing Voice Detection Method with LSTM Recurrent Neural Networks. In *Proc. EUSIPCO*, 2015.
- [22] B. Lehner, G. Widmer, and R. Sonnleitner. On the Reduction of False Positives in Singing Voice Detection. In *Proc. ICASSP*, 2014.
- [23] A. Mahendran and A. Vedaldi. Understanding Deep Image Representations by Inverting Them. In *Proc. CVPR*, 2015.
- [24] A. Mahendran and A. Vedaldi. Visualizing Deep Convolutional Neural Networks Using Natural Pre-images. *International Journal of Computer Vision*, pages 1–23, 2016.
- [25] M. Mauch, H. Fujihara, K. Yoshii, and M. Goto. Timbre and Melody Features for the Recognition of Vocal Activity and Instrumental Solos in Polyphonic Music. In *Proc. ISMIR*, 2011.
- [26] S. Mishra, B. L. Sturm, and S. Dixon. Local Interpretable Model-Agnostic Explanations for Music Content Analysis. In *Proc. ISMIR*, 2017.
- [27] G. Montavon, W. Samek, and KR. Müller. Methods for Interpreting and Understanding Deep Neural Networks. *Digital Signal Processing*, 73(Supplement C):1–15, 2018.
- [28] A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going Deeper into Neural Networks . “<https://research.googleblog.com/2015/06/>

- inceptionism-going-deeper-into-neural.html", 2015.
- [29] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the Preferred Inputs for Neurons in Neural Networks Via Deep Generator Networks. In *Proc. NIPS*, 2016.
 - [30] A. Nguyen, J. Yosinski, and J. Clune. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In *Proc. CVPR*, 2015.
 - [31] C. Olah, A. Mordvintsev, and L. Schubert. Feature Visualization. *Distill*, 2017.
 - [32] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical Black-Box Attacks against Machine Learning. In *Proc. ACM ASIACCS*, 2017.
 - [33] R. Vidal and J. Bruna and R. Giryes and S. Soatto. Mathematics of Deep Learning. *ArXiv e-prints*, arXiv:1712.04741, 2017.
 - [34] M. Ramona, G. Richard, and B. David. Vocal Detection in Music Using Support Vector Machines. In *Proc. ICASSP*, pages 1885–1888, 2008.
 - [35] M. T. Ribeiro, S. Singh, and C. Guestrin. Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proc. KDD*, 2016.
 - [36] J. Schlüter and S. Böck. Improved Musical Onset Detection with Convolutional Neural Networks. In *Proc. ICASSP*, 2014.
 - [37] J. Schlüter and T. Grill. Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks. In *Proc. ISMIR*, 2015.
 - [38] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual Explanations from Deep Networks Via Gradient-Based Localization. In *Proc. ICCV*, 2017.
 - [39] A. Shrikumar, P. Greenside, and A. Kundaje. Learning Important Features Through Propagating Activation Differences. In *Proc. ICML*, pages 3145–3153, 2017.
 - [40] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Proc. ICLR*, 2014.
 - [41] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. Smoothgrad: Removing Noise by Adding Noise. In *Proc. ICML Workshop on Visualisation for Deep Learning*, 2017.
 - [42] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for Simplicity: The All Convolutional Net. In *Proc. ICLR Workshop*, 2015.
 - [43] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic Attribution for Deep Networks. In *Proc. ICML*, 2017.
 - [44] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing Properties of Neural Networks. In *Proc. ICLR*, 2014.
 - [45] A. Weller. Challenges for Transparency. In *Proc. ICML Workshop on Human Interpretability in Machine Learning*, 2017.
 - [46] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding Neural Networks Through Deep Visualization. In *Proc. ICML Deep Learning Workshop*, June 2015.
 - [47] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *Proc. ECCV*, 2014.
 - [48] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding Deep Learning Requires Rethinking Generalization. In *Proc. ICLR*, 2017.

COMPARING RNN PARAMETERS FOR MELODIC SIMILARITY

Tian Cheng, Satoru Fukayama, Masataka Goto

National Institute of Advanced Industrial Science and Technology (AIST), Japan
{tian.cheng, s.fukayama, m.goto}@aist.go.jp

ABSTRACT

Melodic similarity is an important task in the Music Information Retrieval (MIR) domain, with promising applications including query by example, music recommendation and visualisation. Most current approaches compute the similarity between two melodic sequences by comparing their local features (distance between pitches, intervals, etc.) or by comparing the sequences after aligning them. In order to find a better feature representing global characteristics of a melody, we propose to represent the melodic sequence of each musical piece by the parameters of a generative Recurrent Neural Network (RNN) trained on its sequence. Because the trained RNN can generate the identical melodic sequence of each piece, we can expect that the RNN parameters contain the temporal information within the melody. In our experiment, we first train an RNN on all melodic sequences, and then use it as an initialisation to train an individual RNN on each melodic sequence. The similarity between two melodies is computed by using the distance between their individual RNN parameters. Experimental results showed that the proposed RNN-based similarity outperformed the baseline similarity obtained by directly comparing melodic sequences.

1. INTRODUCTION

Melodic similarity is a task to analyse the similarity between melodies, which has been used for music retrieval, recommendation, visualisation and so on. To compute the similarity, a melody is always represented by a sequence of monophonic, musical fragments/events (MIDI event, pitch, etc.). Current approaches usually compare two melodic sequences using the string edit distance [8, 9, 17], geometric measures [19] and N-Gram based measures [5, 27]. Alignment-based methods are applied when two melodic sequences are of different lengths [15, 23], or when events of two sequences are not corresponding to each other one by one [2]. Not only melodic sequence but also melody slopes on continuous melody contours are aligned for comparing melodic similarity [28]. Readers can refer to [25] for state-of-the-art melodic similar-

ity methods. The existing methods focus on local features extracted from melodic sequences, such as distances between pitches or between subsets of melodic sequence (N-Gram). In addition alignment is needed when two melodic sequences are not comparable directly.

In order to deal with these drawbacks, we propose to train a generative Recurrent Neural Network (RNN) on a melodic sequence, and use the RNN parameters to represent the melodic sequence. The proposed feature (RNN parameters) projects a melodic sequence to a point in the parameter space, having two characteristics described as follows. Firstly, the feature is independent to the length of the input melodic sequence because every sequence is represented by its RNN parameters of the same dimension. Secondly, because the RNN can generate an identical sequence, we can expect that the RNN parameters contain the global, temporal information of the melody.

In our experiment, we first train an RNN on all melodic sequences from 80 popular songs as an initialisation. With the initialisation, RNNs are trained on individual melodic sequences. All the networks are trained in tensorflow. We compute the similarity between two melodic sequences by the Cosine similarity of their RNN parameters. The results show that the similarity based on RNN parameters outperforms the baseline similarity obtained by comparing the melodic sequences directly. To the best of our knowledge, this is the first study that uses parameters of generative RNNs for the purpose of computing melodic similarity.

2. RELATED WORK

In this section, we introduce related work on RNN-based melody generation models, and briefly introduce researches on word and sentence embedding for understanding semantic meanings in natural language processing.

2.1 RNN-based melody generation models

We discuss several state-of-the-art RNN-based melody generation models. The RNN-based generative models are usually applied with Long Short Term Memory (LSTM) units in order to model a long time dependence, such as Melody RNN in Magenta [1] and folk-rnn [22]. Magenta [1] uses 2-layer RNNs with 64 or 128 LSTM units per layer, while folk-rnn [22] uses a deeper network (RNN with 3 hidden layers of 512 LSTM units for each layer).

The RNNs generate melody by predicting the next melodic event based on its previous N events:

$$[x_{t-N}, \dots, x_{t-1}] \rightarrow x_t,$$



© Tian Cheng, Satoru Fukayama, Masataka Goto. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Tian Cheng, Satoru Fukayama, Masataka Goto. "Comparing RNN parameters for melodic similarity", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

Models	Representation	Architecture
Magenta [1]	MIDI event	2-layer RNN (LSTM)
Folk-rnn [22]	abc notation	3-layer RNN (LSTM)
Hierarchical RNN [26]	bar profile, beat profile and note	3 RNNs (2-layer LSTM) for bar, beat and note

Table 1: Brief summary of RNN-based melody generation models.

where x_t denotes the melodic event in time t . The melodic event can be represented in many forms, for example MIDI events [1], abc notation [22] and so on, as shown in Table 1. With quantised time steps (in sixteenth notes, for example), a melody can be represented as a sequence of pitches¹ or MIDI events (pitch onset, offset, and no event) [1].

Rhythm information can also be modelled for melody generation. One simple way is to concatenate beat information with the melodic event for each frame to feed into the network [1]. There are also several hierarchical RNNs proposed with rhythm information. In [4], each note is represented by its pitch and duration, and 2 RNNs (rhythm and melody RNNs) are trained for duration and pitch, respectively. The rhythm network receives the current pitch and duration as inputs, and outputs the duration of the next note. The melody network receives the current pitch and generated upcoming duration as inputs to generate the pitch of the next note. [26] trains 3 RNNs for bar, beat, and note, respectively. The first RNN generated bar profiles. Generated bar profiles are fed into the second network to generate beats, and then bar and beat profiles are fed into the third network to generate notes.

Studies of generative RNN models always list generated examples [1, 22] as results, or conduct a listening test for evaluation [26]. We believe that the generative RNN actually learns something ‘musical’ and can be used for music analysis. In this paper we extend the utility of the generative RNN to represent a melody and evaluate it in a melodic similarity task.

2.2 Word embedding and sentence embedding

In natural language processing, word embedding and sentence embedding work on representing semantic meanings of words and sentences. There are two successful word embedding models introduced in [13, 14]: word representations are learnt in order to predict surrounding words or to predict the current word by its content. In these ways, the meaning of a word is related to its context. With the embedded words, a representative vector for a sentence (a sequence of words) can be learned at the same time of parsing the sentence [21] or can be trained in a weakly supervised way on the click-through data by making sentence vectors with similar meanings close to each other [18]. Inspired by word embedding, [11] learns to represent a paragraph by predicting words in the paragraph using previous words and a paragraph vector. The same paragraph vector

is shared when predicting words in the paragraph and then is used to represent the paragraph.

We believe that word embedding may correspond to chord embedding [3, 12] in understanding music; and sentence embedding may correspond to representing a sequence of chords (also an interesting topic to investigate). In general, the musical meaning (of a sequence of pitches or chords) is less intuitive than the textual meaning (of a word or a sentence). Thus, it is more difficult to learn a good representation for a musical sequence. In this paper we work on representing a melody (a sequence of pitches). We train an RNN model to predict the current pitch by its previous pitches in a melody and represent the melody by the RNN parameters. To the best of our knowledge, this is the first work to use network parameters directly as a representation.

3. TRAINING RNNs

For each melodic sequence, we train a generative RNN on it. The parameters of the trained RNN will be used as a feature to represent the melody. We first train an initialisation on all melodic sequences, and then train on individually melodic sequences with the initialisation.

3.1 Data

We conduct the experiment on the RWC Music Database (Popular Music) [7]. There is a subjective similarity study [10] undertaken on 80 songs (RWC-MDB-P-2001 No.1-80) of the RWC Music Database. In this study 27 participants are asked to vote the similarity (on melody, rhythm, vocals and instruments, respectively) for 200 pairs of clips after listening to them. Each clip lasts for 30 seconds (starting from the first chorus starting time). For these pairs of clips, the similarity votes range from 0 to 27.² The larger the vote is, the more similar the clips are. The melodic similarity matrix is shown in Figure 1, indicating the similarity scores of 200 pairs of clips. The matrix is symmetric because if a is similar to b , it means that b is similar to a as well. There are 400 non-zero values in the matrix (twice of 200 because of the symmetry).

We use the same 30-second clip as in the subjective study [10] from each song for training RNNs. We denote the clip from piece ‘RWC-MDB-P-2001 No.X’ as clip X , $X \in [1, 80]$. The melodic similarity results of this study [10] are used as the ground truth for evaluation.

3.2 Arranging the training data

We train RNNs using the melody annotation of the RWC Music Database (Popular Music) from the publicly available AIST Annotation [6]. A melody in the annotation is represented as a fundamental frequency sequence in 10 ms frames as shown in Figure 2(a). We call the frames with frequencies ‘melody frames’, and the frames without frequencies ‘silent frames’. We convert the frequencies (f)

¹ <https://brangerbriz.com/blog/using-machine-learning-to-create-new-melodies/>

² The dataset [10] has been publicly available on the web page of the RWC Music Database at <http://staff.aist.go.jp/m.goto/RWC-MDB/AIST-Annotation/SSimRWC/>.

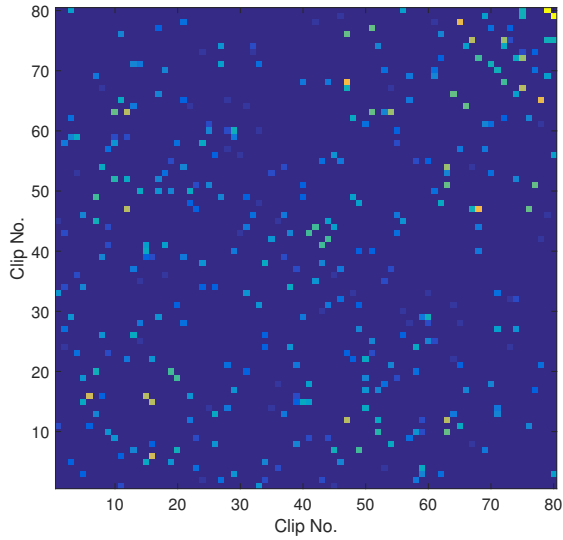


Figure 1: The melodic similarity of 200 pairs of clips.

into pitches (p – indicated by MIDI indices) for melody frames:

$$p = 69 + 12 \log_2 \frac{f}{440}. \quad (1)$$

The histogram of the pitches in the training set is shown in Figure 3. We focus on pitches in 3 octaves ranging from 43 to 78. Frames with pitches beyond this range are considered as silent frames.

3.2.1 Frame hop size

The original frames are arranged in a hop size of 10 ms. We use a hop size of 50 ms (shown in Figure 2(b)) because RNNs tend to repeat the previous frames with a small frame hop size.

3.2.2 Skip silent frames

Because of the high ratio of the silent frames (shown in Figure 2(b)), there will be many invalid training samples with a sequence of silent frames to predict a silent frame if we use all frames in the training data. Therefore, we simply skip all the silent frames to discard those invalid training samples, resulting in a pitch sequence with only melody frames (shown in Figure 5(b)).

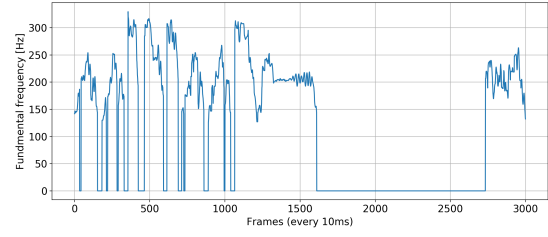
We aim to look back for 2 seconds to predict the next frame. With a frame hop size of 50 ms, there are 40 frames in the input sequence: $[x_{t-N}, \dots, x_{t-1}] \rightarrow x_t, N = 40$.

3.2.3 Zero-padding at the beginning

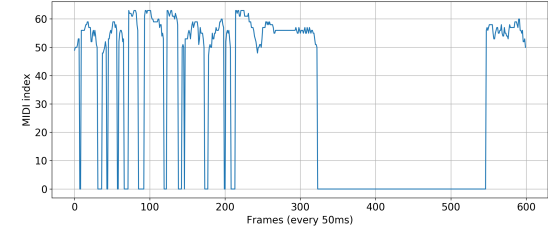
We find if the first training sample is $[x_0, \dots, x_{39}] \rightarrow x_{40}$, then the generation of the first 40 frames are not modelled in the RNN. In order to generate the whole sequence, we concatenate a sequence of 40 silent frames in the front of each clip, with the first training sample of $[x_S, \dots, x_S] \rightarrow x_0$ (x_S is the silent frame padding in the front of the clip).

3.3 Network architecture

We apply a network architecture similar to Megenta [1], but with GRU cells instead of LSTM cells to reduce the



(a) A sequence of fundamental frequencies.



(b) A sequence of pitches

Figure 2: Melodic sequences with different frame hop sizes. Frames with values of 0 are silent frames.

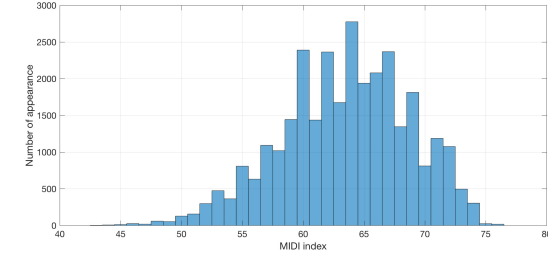


Figure 3: The histogram of the pitches in the dataset.

parameter dimensions. The RNN contains 2 hidden layers with 64 GRU cells per layer. The output layer is a fully-connected layer with a softmax activation function. The inputs are one-hot encoded vectors with a dimension of 37 (36 pitches and a silent state). We hope the RNN can fit the individual pitch sequences as much as possible. In this case, overfitting is intended and not a problem any longer; hence no drop out is applied.

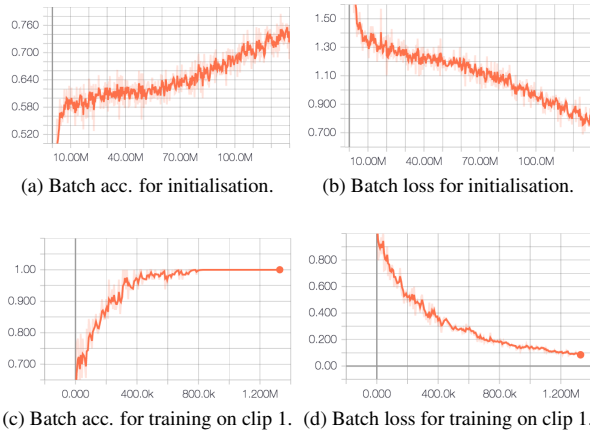
The network is trained by minimising the cross entropy loss using Adam optimisation with learning rate of 0.001 (other parameters of Adam are with default values in tensorflow).

3.4 Initialisation and training on individual clips

In order to gain a consistent training, we use a fixed initialisation. The initialisation is trained on the training samples from all 80 clips for 100 epochs. Then with this initialisation, we train an individual RNN on each melodic sequence for 500 iterations.³ After data arrangement of Section 3.2, there are around 200-600 training samples for

³ An iteration means RNN parameters are updated once on a batch of training samples. In contrast, an epoch means a full training on all training samples. We use the iteration number to stop training because in this way RNN parameters are updated for the same times, hence more comparable. However, when to stop training still needs further investigation.

	Initialisation	Individual RNNs
No. of RNNs	1 RNN	80 RNNs
Training data	80 clips	each clip
Batch size	512	64
Early stop	100 epochs	500 iterations

Table 2: RNN training settings.**Figure 4:** Batch accuracies and losses of training for initialisation and training on clip 1 with the initialisation.

every clip. We use a large batch size of 512 for initialisation training because of a big number of training samples, and a smaller batch size of 64 for training for each individual sequence. Training settings are shown in Table 2.

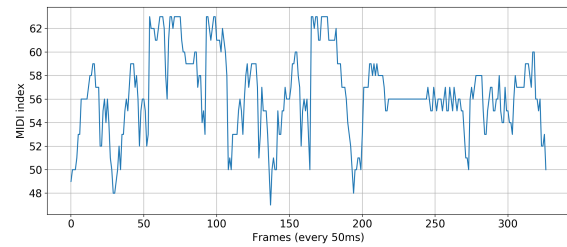
Training for initialisation and training on clip 1 are shown in Figure 4. After training for initialisation, the batch accuracy reaches 0.7 (Figure 4(a)) and the batch loss decreases to around 0.8 (Figure 4(b)). After training on clip 1 with the initialisation, the batch accuracy further increases from 0.7 to 1 (Figure 4(c)); and the batch loss reduces from 0.8 to around 0.1 (Figure 4(d)). With the RNN trained on clip 1, we can generate an identical melodic sequence, as shown in Figure 5.

3.5 Cosine similarity between RNN parameters

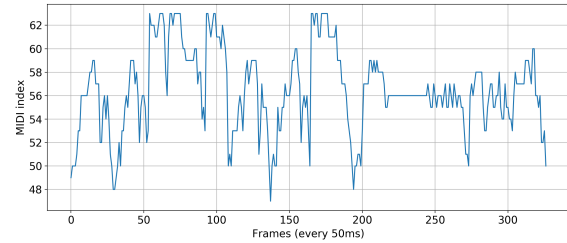
The parameter dimensions of an RNN are shown in Table 3. The total number of parameters is 46,757.

We reshape matrices to vectors, and concatenate the vectors. The concatenated parameters of the initialisation RNN and RNNs trained on clip 3 and clip 80 are shown in Figure 6. The differences in parameters of different RNNs are subtle. The similarity between two clips is indicated by the Cosine similarity between their concatenated RNN parameters. The larger the Cosine similarity is, the more similar the clips are.

In the data arrangement stage (see Section 3.2), the melody of a clip (30 seconds) is represented as a sequence of pitches of 600 frames (including silent frames), as shown in Figure 2(b). We use the Cosine similarity between two pitch sequences as the baseline similarity.



(a) Generated pitch sequence.



(b) Original pitch sequence of clip 1.

Figure 5: An identical pitch sequence generated by the trained RNN.

Matrix	Dimension
cell_0/gru_cell/gates/kernel	(101, 128)
cell_0/gru_cell/gates/bias	(128)
cell_0/gru_cell/candidate/kernel	(101, 64)
cell_0/gru_cell/candidate/bias	(64)
cell_1/gru_cell/gates/kernel	(128, 128)
cell_1/gru_cell/gates/bias	(128)
cell_1/gru_cell/candidate/kernel	(128, 64)
cell_1/gru_cell/candidate/bias	(64)
fully_connected/weights	(64, 37)
fully_connected/biases	(37)
all parameters	46,757

Table 3: Parameter dimensions.

4. RESULTS ANALYSIS

4.1 Evaluation metric and results

In the subjective similarity study, each clip is compared to 4-6 other clips, usually 5 clips [10]. For example, clip 3 is compared to clips as shown in Table 4(a). We measure the similarity of two clips by computing the Cosine similarity between their RNN parameters. We compare the rank of votes to the rank of similarities for evaluation. For example, as shown in Table 4(a), 8 people vote the melody of clip 80 is similar to that of clip 3, and 7 people vote the similarity between clip 29 and clip 3. Based on these votes we assume clip 80 is more similar to clip 3 than clip 29. Thus, the Cosine similarity between clip 80 and clip 3 should be larger than that between clip 29 and clip 3 $C(80, 3) > C(29, 3)$. We first convert the similarity and votes into ranks (as shown in Table 4(b)), and then use the pair-wise evaluation metric—Kendall's tau (τ)—to compare the ranks. For clip 3, the τ is 0.2 based on similarities between RNN parameters, better than $\tau = -0.2$ based on

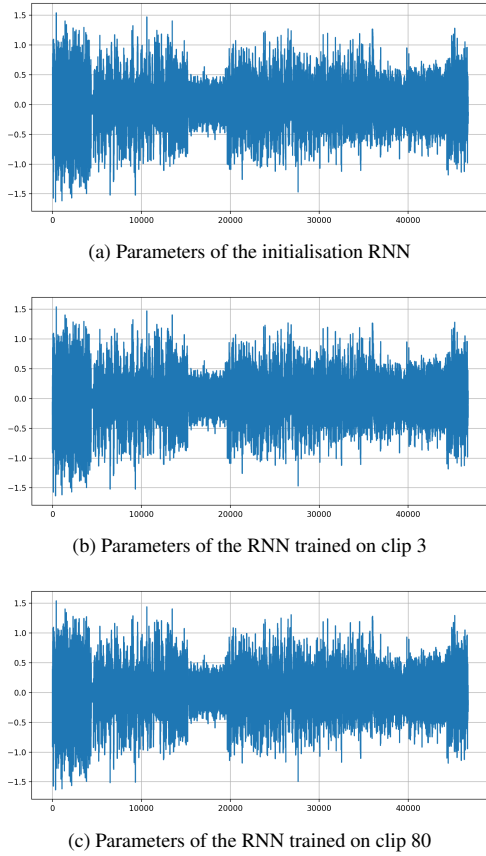


Figure 6: Parameters of different RNNs with subtle differences.

similarities between pitch sequences.

The results for 200 pairs of clips are shown in Table 5. The average τ s are 0.125 and 0.073 based on Cosine similarities between RNN parameters and between pitch sequences, respectively.⁴ In the preliminary test, we found that there is no improvement in performance by using a dimension-reducing technique, such as Principle Component Analysis (PCA), before computing Cosine similarity, or by using distances between eigenvectors (weighted by eigenvalues) of parameter matrices.

4.2 Visualisation

4.2.1 Similarity v.s. vote

We assume if there are more votes on X than on Y when comparing to A, then the X should be more similar to A than Y. However, this may be too strict when votes are close (8 on X and 7 on Y, for example). In order to show whether there is a trend that the similarity value is larger for pairs of clips with a higher vote in general, we show Cosine similarity v.s. vote plots for RNN parameters and baseline pitch sequences in Figure 7.

We know the RNN parameters of different clips are very similar to each other, as shown in Figure 6. Therefore, the

⁴ Using the Euclidean distance provides similar results as using the Cosine similarity: 0.120 and 0.074 for RNN parameters and pitch sequences, respectively.

No.	80	29	59	62	5
$Votes$	8	7	6	4	3
C_{RNN}	0.9975	0.9973	0.99717	0.9976	0.99725
C_{pitch}	0.6175	0.7146	0.6256	0.7097	0.6584

(a) Cosine similarities between parameters of clips compared to clip 3.

No.	80	29	59	62	5	τ
R_{Votes}	1	2	3	4	5	
R_{RNN}	2	3	5	1	4	0.2
R_{pitch}	5	1	4	2	3	-0.2

(b) Ranks of Cosine similarities.

Table 4: Evaluation for clip 3. C_{RNN} and C_{pitch} are the Cosine similarities between parameters and between pitch sequences, respectively.

Similarity	τ
C_{RNN}	0.125
C_{pitch}	0.073

Table 5: Results.

Cosine similarities between RNN parameters are in a small range from 0.995 to 0.999 (Figure 7(a)). The Cosine similarities between melodic sequences are in a larger range from 0.4 to 0.9 (Figure 7(b)). However, neither RNN parameters nor melodic sequences provide a clear trend of the similarity increasing with number of votes.

4.2.2 t-SNE

To visualise the 80 songs in a low-dimensional space, we first reduce the dimension of the features to 5 by PCA, then further reduce it to 2 by t-SNE, with the implementation of [20]. The visualisation based on RNN parameters and pitch sequences is shown in Figure 8. For a clearer visualisation, we only indicate pairs of clips with higher votes (above 9 votes out of 27, as listed in Table 6) by connecting those pairs with lines.

Because the t-SNE visualisation is not a linear projection from the similarity to the distance on the 2-dimensional space, we do not compare the vote against the distance between two clips in t-SNE visualisation, but focus on the grouping of clips. We observe some interesting grouping of clips in Figure 8(a): the triangle at the top left for (75, 79, 80), and two lines at bottom right connecting (15, 16) and (6, 16). In Figure 8(b), no such grouping of clips can be obviously observed.

5. DISCUSSIONS AND CONCLUSIONS

From the t-SNE visualisation, we observe some interesting grouping of clips based on RNN parameters (Figure 8(a)). However, visualisation based on the Cosine similarity between RNN parameters does not show a clear relation between the similarity and the vote (Figure 7(a)). It may

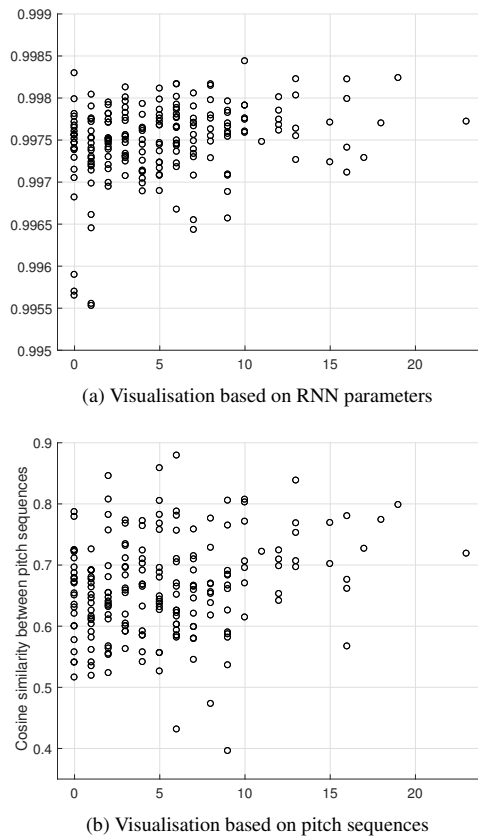


Figure 7: Similarity v.s. vote plot based on different features.

indicate that a direct comparison between RNN parameters is too simple to infer the information in such a large dimension. Figure 6 also illustrates the difficulties with the proposed approach, too many parameters with subtle differences. We would like to dig deeper to understand which parameters are most significant for computing melodic similarity.

Perception studies show that changes in relative scale or relative duration do not have a major impact on melodic similarity [24]. The similarity measure should be invariant to music transformations, such as transposition in pitch and tempo changes [16, 23]. The proposed generative RNN can model the input pitch sequence, but cannot deal with the

No.	Pair	Vote	No.	Pair	Vote	No.	Pair	Vote
1	(79, 80)	23	11	(10, 63)	13	21	(10, 52)	11
2	(47, 68)	19	12	(47, 76)	13	22	(7, 20)	10
3	(65, 78)	18	13	(51, 63)	13	23	(7, 45)	10
4	(6, 16)	17	14	(51, 77)	13	24	(29, 60)	10
5	(12, 47)	16	15	(64, 66)	13	25	(47, 67)	10
6	(12, 63)	16	16	(7, 49)	12	26	(70, 71)	10
7	(15, 16)	16	17	(19, 20)	12	27	(75, 79)	10
8	(67, 75)	16	18	(41, 43)	12	28	(75, 80)	10
9	(54, 63)	15	19	(42, 44)	12			
10	(72, 75)	15	20	(68, 72)	12			

Table 6: A list of pairs of songs with similarity votes above 9 votes out of 27.

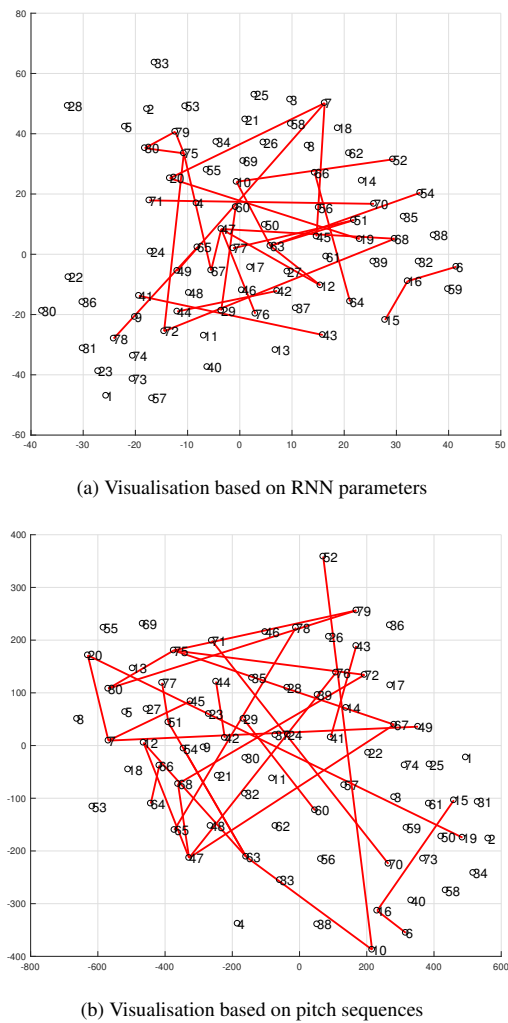


Figure 8: t-SNE visualisation based on different features.

similarity under music transformations. In the future, we would like to tackle this problem by training RNNs with coordinate differences instead of absolute coordinates as inputs, such as intervals and durations instead of pitches and onsets [16].

We work on the melodic similarity based on the performance-based representation of melodies, which seems to complicate the task. We hope we can achieve more success on symbolic melody representation by using score-based representation on a simpler dataset.

In this paper, we propose to represent a melodic sequence by the parameters of its corresponding generative RNN, and test the utility of the melodic feature (RNN parameters) in the melodic similarity task. The proposed feature contains temporal information within the melodic sequence, and independent of the length of the sequence. We extend the utility of generative RNNs to use the network for music similarity analysis rather than music generation. We expect that the proposed feature (generative RNN parameters) can be used in other tasks, such as musicological analysis and music cognition.

6. ACKNOWLEDGEMENT

This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan.

7. REFERENCES

- [1] Magenta: Melody RNN. https://github.com/tensorflow/magenta/tree/master/magenta/models/melody_rnn. Accessed: 2018-03-30.
- [2] D. Bountouridis, D. G. Brown, F. Wiering, and R. C. Veltkamp. Melodic Similarity and Applications Using Biologically-Inspired Techniques. *Applied Sciences*, 7(12), 2017.
- [3] G. Brunner, Y. Wang, R. Wattenhofer, and J. Wiesen-danger. JamBot: Music Theory Aware Chord Based Generation of Polyphonic Music with LSTMs. In *Proceedings of the 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2017.
- [4] F. Colombo, S. P. Muscinelli, A. Seeholzer, J. Brea, and W. Gerstner. Algorithmic Composition of Melodies with Deep Recurrent Neural Networks. *Computing Research Repository (CoRR)*, abs/1606.07251, 2016.
- [5] J. S. Downie. *Evaluating a Simple Approach to Musical Information retrieval: Conceiving Melodic N-grams as Text*. PhD thesis, University of Western Ontario, 1999.
- [6] M. Goto. AIST Annotation for the RWC Music Database. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, pages 359–360, 2006.
- [7] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC Music Database: Popular, Classical, and Jazz Music Databases. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*, pages 287–288, 2002.
- [8] M. Grachten, J.-L. Arcos, and R. L. de Mantaras. Melodic Similarity: Looking for a Good Abstraction Level. In *Proceedings of the 5th International Society of Music Information Retrieval (ISMIR)*, 2004.
- [9] P. Hanna, P. Ferraro, and M. Robine. On Optimizing the Editing Algorithms for Evaluating Similarity Between Monophonic Musical Sequences. *Journal of New Music Research*, 36(4):267–279, 2007.
- [10] S. Kawabuchi, C. Miyajima, N. Kitaoka, and K. Takeda. Subjective Similarity of Music: Data Collection for Individuality Analysis. In *Proceedings of Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–5, 2012.
- [11] Q. Le and T. Mikolov. Distributed Representations of Sentences and Documents. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2014.
- [12] S. Madjiheurem, L. Qu, and C. Walder. Chord2Vec: Learning Musical Chord Embeddings. In *Proceedings of the Constructive Machine Learning Workshop at 30th Conference on Neural Information Processing Systems (NIPS)*, 2016.
- [13] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of International Conference on Learning Representations (ICLR) Workshop*, 2013.
- [14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of Advances in Neural Information Processing Systems 26 (NIPS)*, pages 3111–3119, 2013.
- [15] D. Müllensiefen and K. Frieler. Optimizing Measures of Melodic Similarity for the Exploration of a Large Folk Song Database. In *Proceedings of the 5th International Society of Music Information Retrieval (ISMIR)*, pages 1–7, 2004.
- [16] D. Müllensiefen and K. Frieler. Evaluating Different Approaches to Measuring the Similarity of Melodies. In et al. V. Batagelj, editor, *Data Science and Classification*, pages 299–306. Springer, Berlin, 2006.
- [17] K. S. Orpen and D. Huron. Measurement of Similarity in Music: A Quantitative Approach for Non-parametric Representations. *Computers in Music Research*, 4:1 – 44, 1992.
- [18] H. Palangi, P. li, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward. Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707, 2016.
- [19] M. W. Park and E. C. Lee. Similarity Measurement Method between Two Songs by Using the Conditional Euclidean Distance. *Wseas Transaction On Information Science And Applications*, 10(12), 2013.
- [20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [21] R. Socher, C. Y. Lin, A. Y. Ng, and C. D. Manning. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of International Conference on Machine Learning (ICML)*, 2011.

- [22] B. L. Sturm, J. F. Santos, and I. Korshunova. Folk Music Style Modelling by Recurrent Neural Networks with Long Short Term Memory Units. In *Extended abstracts for the Late-Breaking Demo Session of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.
- [23] J. Urbano, J. Lloréns, J. Morato, and S. Sánchez-Cuadrado. MIREX 2012 Symbolic Melodic Similarity: Hybrid Sequence Alignment with Geometric Representations. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2012.
- [24] M. R. Velankar, H. V. Sahasrabuddhe, and P. A. Kulkarini. Modeling Melody Similarity Using Music Synthesis and Perception. *Procedia Computer Science*, 45:728 – 735, 2015.
- [25] V. Velardo, M. Vallati, and S. Jan. Symbolic Melodic Similarity: State of the Art and Future Challenges. *Computer Music Journal*, 40(2):70–83, 2016.
- [26] J. Wu, C. Hu, Y. Wang, X. Hu, and J. Zhu. A Hierarchical Recurrent Neural Network for Symbolic Melody Generation. *Computing Research Repository (CoRR)*, abs/1712.05274, 2017.
- [27] S. Yazawa, Y. Hasegawa, K. Kanamori, and M. Hamanaka. Melodic Similarity Based on Extension Implication-Realization Model. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2013.
- [28] Y. Zhu, M. Kankanhalli, and Q. Tian. Similarity Matching of Continuous Melody Contours for Humming Querying of Melody Databases. In *Proceedings of IEEE Workshop on Multimedia Signal Processing*, 2002.

VISUALIZATION OF AUDIO DATA USING STACKED GRAPHS

Mathieu Lagrange*, Mathias Rossignol, Grégoire Lafay*

*LS2N, CNRS, École Centrale de Nantes
mathieu.lagrange@cnrs.fr

ABSTRACT

In this paper, we study the benefit of considering stacked graphs to display audio data. Thanks to a careful use of layering of the spectral information, the resulting display is both concise and intuitive. Compared to the spectrogram display, it allows the reader to focus more on the temporal aspect of the time/frequency decomposition while keeping an abstract view of the spectral information.

The use of such a display is validated using two perceptual experiments that demonstrate the potential of the approach. The first considers the proposed display to perform an identification task of the musical instrument and the second considers the proposed display to evaluate the technical level of a musical performer. Both experiments show the potential of the display and potential applications scenarios in musical training are discussed.

1. INTRODUCTION

The visual display of quantitative information [13] is at the core of the growth of human knowledge as it allows human beings to go beyond the limitation of natural languages in terms of precision and scale.

Defining what is the essence of a good visual display of quantitative data is non trivial and usually domain specific. That said, in most scientific fields, such displays serve two majors goals: 1) the routine interaction of the researcher with the data or the physical phenomenon and 2) the need of the researcher to motivate its claim to its peers. Both tasks require the display to fulfill the simplicity rule both in terms of production and design. First, the display shall be computed and adapted according to the need of the researcher very efficiently in order to allow an effective exploration of the data. Second, the display shall be able to convey at the first glance an important qualitative aspect about the data.

This paper is about the visualization of audio data, and audio data is originally made to be listened to. Therefore, we shall keep in mind that "all visual projections of sounds are arbitrary and fictitious" [11]. That said, even if recorded versions of sounds can now be played back

at convenience, it is still useful to represent them graphically as listening depends on time. On contrary, the visual display allows the reader to grasp a global view of the waveform at a glance. Also, the eye is less subject to stimulation fatigue and the visual display is very powerful to convey evidence as we are still fully into the print culture that since the Gutenberg invention gives an "uncritical acceptance [to] visual metaphors and models" [8].

We propose in this paper a display of audio data that is intuitive and gives information about the main dimensions of sound in a compact manner using stacked graphs [3]. The display can be computed easily and efficiently¹. In order to put this display into context, an overview of the routinely used type of displays is given, respectively from the perspective of the musician composer in Section 2 and the physicist in Section 3. We shall argue that the proposed display fully described in Section 4 can be thought of as the physicist's counterpart to a notational system introduced by Schafer [11].

The display is then evaluated and compared to the commonly used waveform and spectrogram displays with two perceptual experiments. In the first experiment, the subjects have to distinguish between tones of different musical instruments by listening to the sounds or by considering the visual displays under evaluation. The protocol and the results for this experiment are presented in Section 5. In the second experiment, the subjects are asked to distinguish between saxophone performances of different level of instrumental expertise by listening to the sound and by considering the displays under evaluation. The protocol and the results for this experiment are presented in Section 6.

2. ABOUT NOTATION

From the phonetic alphabet for speech to the musical score for music, notation consists in putting together on a one or two-dimensional space symbols describing specific sound events. In a manner probably inherited from writing, time sequencing is usually depicted from left to right in the Western musical culture. Specific to the musical score is the use of the vertical axis to depict the pitch. A musical tone is therefore solely described in terms of time of appearance, duration, pitch and sometimes intensity. As such, the score is largely prescriptive and gives a tremendous amount of freedom to the musical performer in terms



© Mathieu Lagrange*, Mathias Rossignol, Grégoire Lafay* . Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Mathieu Lagrange*, Mathias Rossignol, Grégoire Lafay* . "Visualization of audio data using stacked graphs", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

¹ A reference implementation as well as all the data discussed in this paper is available at <https://bitbucket.org/mlagrange/paperaudiostackgraph>

	Attack	Body	Decay
Duration	moderate	non-existent	slow
Frequency	steady low		
Fluctuations	transient	steady-state	
Dynamics	loud to soft		
Duration	← 3 seconds →		

Figure 1: Annotation of a church bell from Schafer [11].

of interpretation.

In an intent to provide a more descriptive notation of musical objects, Schaeffer [10] designed a "solfège des object musicaux" that extensively apprehend the description of any kind of sound object. Perhaps because of its complexity this notation is hardly used today. In an effort to simplify this notation, Schafer proposed a notational system that can be considered for describing any kind of sound, be it a unique event or any kind of compound. The main rationale is to split the temporal axis from left to right into 3 parts corresponding to the *attack*, *sustain* and *decay*. For each part, its duration, frequency (related to the notion of mass as introduced by Schaeffer), fluctuations (related to the notion of grain as introduced by Schaeffer) and dynamics are displayed from top to bottom. Except for the frequency content that is depicted as a rough spectrogram contour, the other dimensions are described according to a specific alphabet of a few symbols. An example taken from [11] of such annotation is given on Figure 1 for the sound of a church bell.

3. ABOUT MEASURE

When dealing with sound as a physicist, one wants to quantify mechanical properties and display them precisely. As in notation, the main aspect that is commonly looked for is the distribution of energy across frequency and time. The distribution of energy as a function of the modulation rate and the frequency scale of observations are less considered in the signal processing literature [2,4] but are shown to be perceptually important [5,14].

Therefore, in order to display a sound on a two-dimensional plane, one has to resort to a choice or a compromise. Either timing is emphasized and frequency neglected as in the waveform display 2a or frequency is emphasized and timing neglected as in the display of the Fourier spectrum 2b. A compromise is made by considering time and frequency respectively as horizontal and vertical axes of the two-dimensional plane as with the popular spectrogram *magnitude of the short term Fourier transform*, see Figure 2c. In such display, the use of a color code conveys information about energy.

That said, we believe that the spectrogram display still favors frequency over time. Spectral structure can be analyzed precisely, for example harmonicity, modulations, etc. Conversely, temporal dynamics and structure are harder to appreciate, as the way energy fluctuates in each sub bands has to be reconstructed from the color code.

The spectrogram is a display that is thus in our opinion very powerful for close inspection of a sound event that is

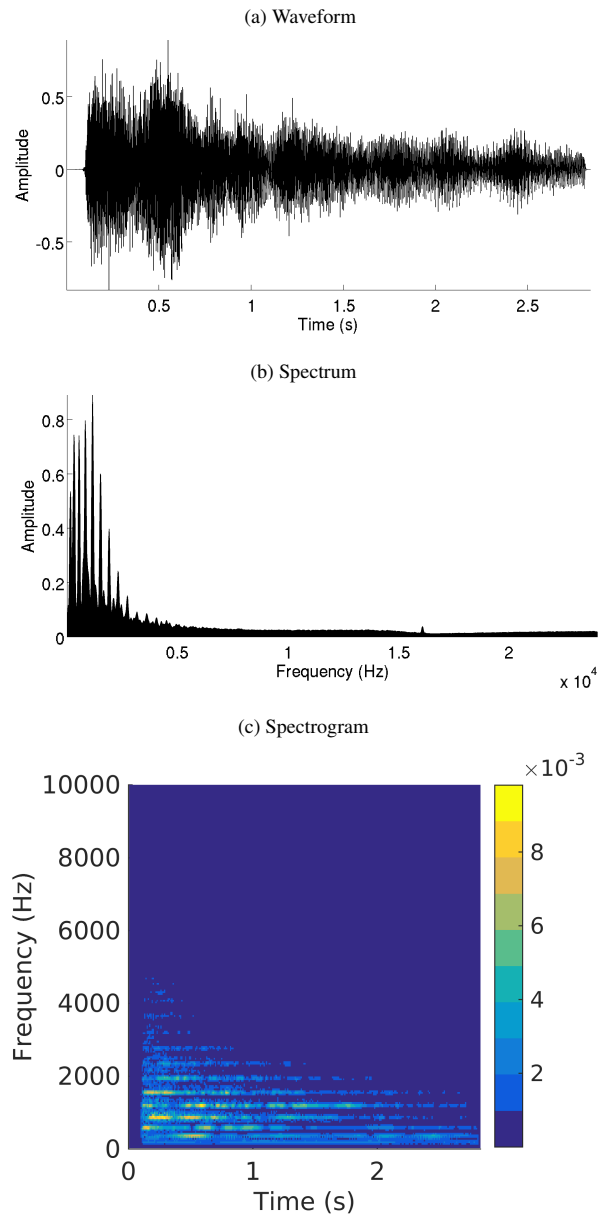


Figure 2: Standard displays of the sound of a church bell.

active over a short period of time. Indeed, enlarging the time resolution quickly blurs the frequency resolution and may lead to a completely non informative display.

4. VISUALIZING SPECTRAL CONTENT USING STACKED GRAPH

With those limitations in mind, we propose in this paper to take a compromise that conversely favors time over frequency. In such display, the plane is therefore organized with time and energy as the horizontal and vertical axes respectively. The frequency is displayed as stacked layers displaying the level of energy across frequency sub bands of growing frequency range. Those layers can have colors assigned.

We seek a display that depicts information that is per-

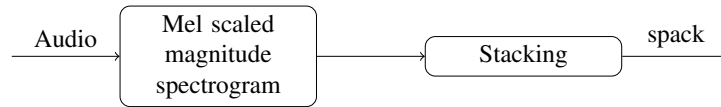


Figure 3: Processing chain of the spack display.

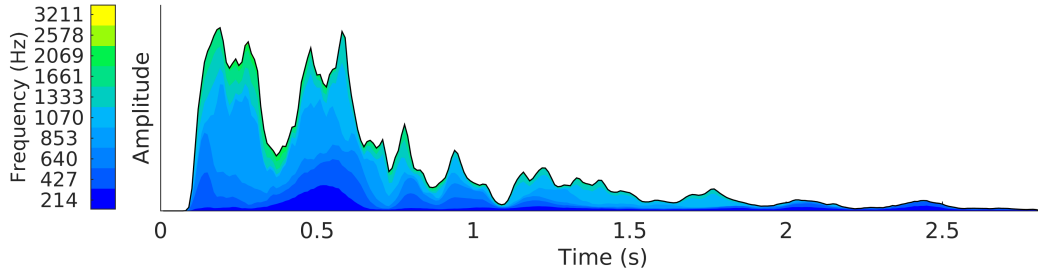


Figure 4: Spectral stack display (spack) of the sound of a church bell. The color code conveys nicely the modulation within each frequency band and the overall disappearance of the high frequency range.

ceptually meaningful. Therefore, we consider spectral data projected on a Mel-scale [12].

In order to improve legibility, colors are assigned to frequency layers according to their ranges with a color code ranging from blue (low frequency) to yellow (high frequency). The blue color is often associated with large phenomena, with the following adjectives: celestial, calm, deep, whereas the yellow color is often associated with transient phenomena that are highly energetic. Kandinsky in [7] states that "Blue is comparable to low pitched organ sounds. Yellow becomes high pitched and can not be very deep". The color code is then chosen to be a linear gradient from blue (low frequency range) through green (middle frequency range) to yellow (high frequency range). In this paper, the gradient follows the LCH color model specified by the Commission Internationale de l'Éclairage (CIE) so that the perceived brightness appears to change uniformly across the gradient while maintaining the color saturation.

We argue that this display, termed spectral stack (spack), convey useful information about the sound. In particular, it conveys nicely, aside of fine details, the important dimensions retained by Schafer, see Figure 1.

To compute the spack display, a mel-scaled magnitude spectrogram is computed from the audio, see Figure 3. To each mel spectral band is assigned a given color code from dark blue (low frequency) to yellow (high frequency). At each time frame, the spack display is a stacking of the magnitude values of each mel frequency band, see Figure 4.

5. TASK 1: IDENTIFYING THE MUSICAL INSTRUMENT

The identification of the musical instrument used to play a tone rely largely on 2 factors, the spectral envelope and the attack [1, 6]. The spack display shall be able to conveniently display those factors. Indeed, the spectral envelope, *i.e.* the distribution of the energy across frequency is encoded using the stacking axis and color code. The attack is

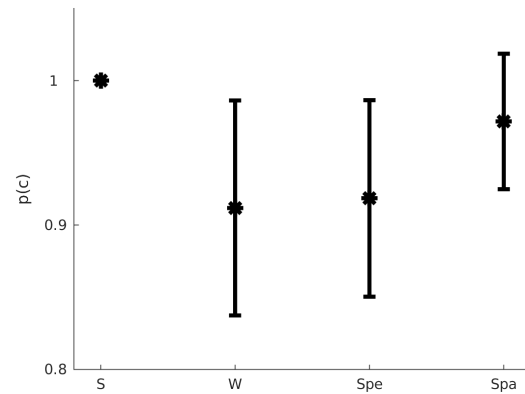


Figure 5: Classification performance of the different displays on Task 1 (identifying the musical instrument): sound (S) waveform (W), spectrogram (Spe) and spack (Spa). The star shows the average performance and the length of the vertical line is twice the standard deviation.

also well displayed as the spack focuses on the display of energy through time.

5.1 Protocol

Several tones played by four musical instruments: piano, violin, trumpet, and flute are considered as stimuli. Each instrument is played *mezzo forte* at 5 different pitches: C, D, E, F and G. For each sound, three visual representations are evaluated: waveform (W), spectrogram (Spe) and spack (Spa). For reference, the sound (S) is also considered².

The test is a forced-choice categorization task. The sounds are displayed by gray dots on a 2 dimensional plane displayed on a computer screen. The dots can be moved

² The sounds and the visual displays are available on the companion website

freely within this plane and colored using 4 different colors, each corresponding to a given instrument. The correspondence is given to the subjects at the beginning of the experiment by the instructor: piano (black), violin (red), trumpet (magenta), and flute (green). If the sound modality is tested, the sound is played when the dot is clicked. If a visual modality is tested, the corresponding display is shown when the dot is clicked using the mouse.

Eight subjects, studying at the Engineering school "Ecole Centrale de Nantes", aged from 24 to 26 years, performed the test. Each subject reported normal hearing. They performed the test at the same time in a quiet environment using headphones. The sound level was set to a comfortable level before the experiment. A short introduction was given by the instructor for each display with a focus on the meaning of the axes and the color code. The subjects performed the evaluation using the sound modality first. The order of the three remaining modalities are ordered randomly among subjects to reduce the impact of precedence. The test is over when the subjects have assigned a color to each dot, this for all the evaluated modalities.

5.2 Results

Classification performance is evaluated as the number of couple of sounds played by the same instrument that have been assigned the same color divided by the number of couples. As can be seen on Figure 5, the task is trivial when listening to the sound, as the subjects achieve perfect classification. On overall, the classification is quite good for each of the graphical displays with a higher average performance for the spack display. Subjects verbally reported ease of use for the spack display.

6. TASK 2: ASSESSING THE LEVEL OF A SAXOPHONE PERFORMANCE

The control of the breath while playing the saxophone is crucial and can be monitored to assess the technical level of a saxophone player [9]. For example, playing a single tone with sharp attack and constant amplitude during the steady state is non trivial and requires years of practice.

Professional players typically practice such exercises on a daily basis as warm-ups and perform them with a trainer to get criticisms in order to improve their skills. Using graphical displays of their performance could be useful for them to spot during or after the performance. In order to be efficient, such display shall be intuitive with a few degrees of freedom in order to be easy to understand.

The validation of the spack display for such pedagogical needs is out of the scope of this paper. Nonetheless, we designed here a task that can demonstrate how several meaningful characteristics of the saxophone performance can be identified only by considering the graphical displays under evaluation.

In this kind of training, it could be useful for the trainer to have some kind of display of its performance. As the crucial part is to be able to control the air flow while playing in order to keep a stable amplitude and timbre, we hy-

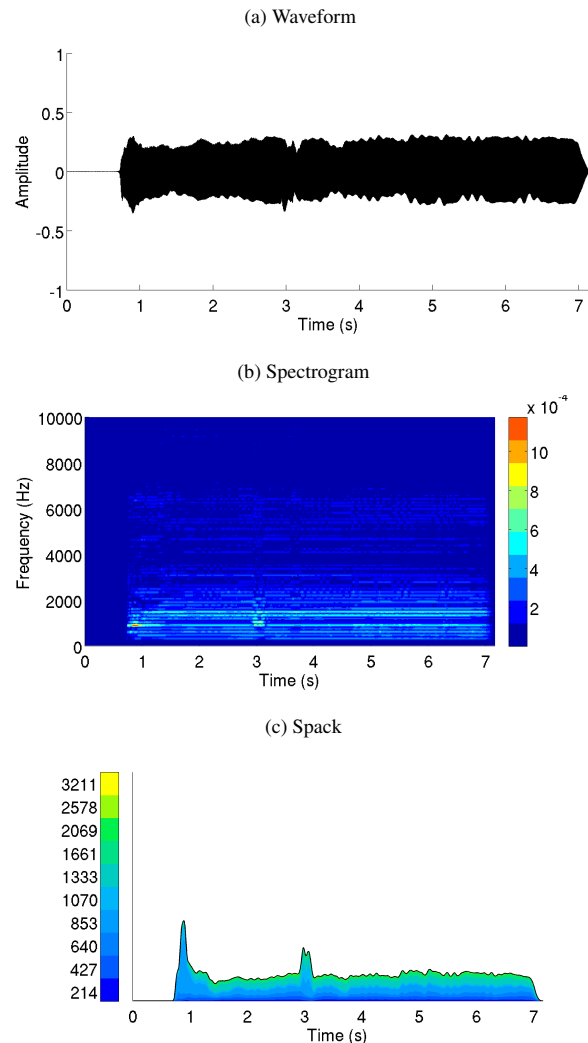


Figure 6: Graphical displays of *forte* B tone. Several performance issues can be observed: lack of airflow control at the attack, change of pitch and loudness at 3 seconds and lack of steady airflow during the whole performance.

pothesize that the spack display may be a good candidate for such a task.

6.1 Protocol

The stimuli considered in this experiment are recorded performances of four saxophone players with a technical level assumed to be high or low (2 low, 2 high). Each player played several tones at pitch B and G. They were asked to play each note in three different ways: *piano*, *forte* and *crescendo decrescendo*³.

The test follows a XXY structure, where three performances are shown to the subject, one is at a given level (high or low) and the other two of the other level (low or high). The subject is then asked, based solely on the modality at hand, to select the one that is different from the two others. 24 triplets are randomly selected from the

³ The sounds and the visual displays are available on the companion website

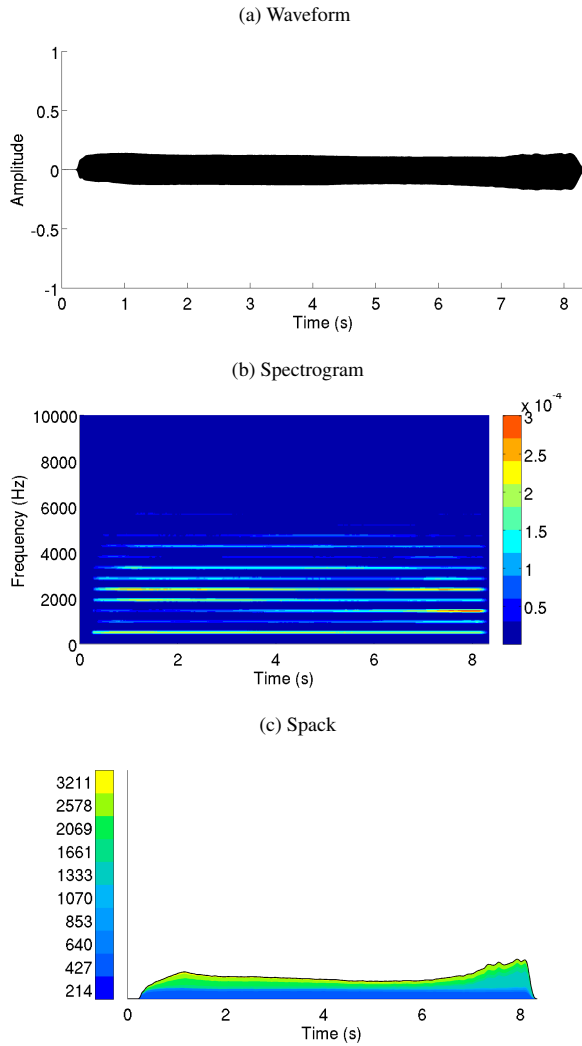


Figure 7: Graphical displays of another *forte* B tone. Several performance issues can be observed, for example: lack of sharpness at the attack, change of timbre and loudness at 5 seconds.

valid combinations of the above described stimuli.

16 subjects, studying at the Engineering school "Ecole Centrale de Nantes", aged from 24 to 28 years, performed the test in two sessions, 9 for the first session, and 7 for the second session. Each subject reported normal hearing. For each session, they performed the test at the same time in a quiet environment using headphones. The sound level was set to a comfortable level before the experiment. A short introduction was given by the instructor for each display with a focus on the meaning of the axis and the color code. The subjects performed the evaluation using the sound modality first. The order of the three remaining modalities are ordered randomly among subjects to reduce the impact of precedence. The test is over when the subjects have examined the 24 triplets for the 4 evaluated modalities.

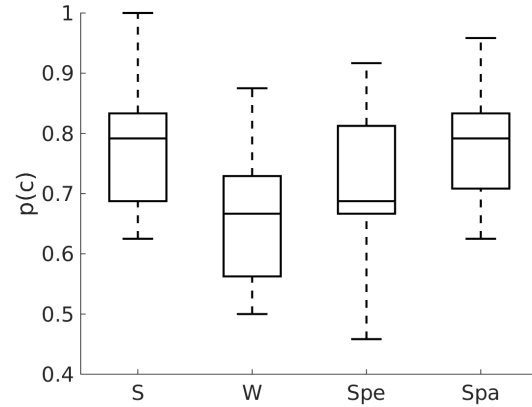


Figure 8: Boxplot display of the differentiation performance of the different displays on Task 2 (detecting the level of the saxophone player): sound (S), waveform (W), spectrogram (Spe) and spack (Spa).

Table 1: Results of the repeated measure ANOVA evaluating the effect of the type of display on the performance.

	sum sq.	df	mean sq.	F	p-value
Type	0.13	3	0.045	5.3	0.003
Error	0.38	45	0.008		

6.2 Results

For each modality, the number of correct selection is averaged among the 24 triplets and then averaged among subjects. As can be seen on Figure 8, the task is more complex than task 1, as the score achieved using the sound modality is lower than task 1. This might be due to the fact that the task is less explicit than task 1. For the visual displays, the same ranking as task 1 is observed with a larger difference between each modality.

A repeated measure ANOVA is used to test the potential significance of the type of display on the differentiation performance. A mauchly test reveals that the default of sphericity is not significant, thus no correction of the degrees of freedom of the Fisher test is needed. Table 1 presents the results of the Fisher test showing that the effect of the representation is significant $p = 0.003$. In addition, a multiple comparison test shows that the only significant differences are between Waveform and Spack $p = 0.03$ and Waveform and Sound $p = 0.003$. No significant difference is found between the remaining modalities: the Sound, the Spectrogram and the Spack displays.

Thus, if considering the graphical displays solely, only the Spack displays significantly improves upon the Waveform display. As can be seen on Figure 8, The spectrogram display have the largest dispersion of correct answer rate, *i.e.* the ratio of correct responses over the number of possible responses, termed $p(c)$ in the following. Considering the distribution of $p(c)$ for the spectrogram display shown on Figure 9, two modes can be observed contrary to the one of the spack display. Even though each sub-

jects have been given the same introduction to each of the graphical displays, their familiarity with the standard displays may vary since some subjects had previous training in signal processing courses. This may explain the higher mode in the distribution of the spectrogram display. Even if this observation shall be considered with care due to the rather low number of subjects, this can lead us to conjecture about the influence of the familiarity of the subjects with the spectrogram display on the reported performance. The spack display does not exhibit the same distribution profile and prior familiarity cannot be assumed as the display was equally new to all subjects.

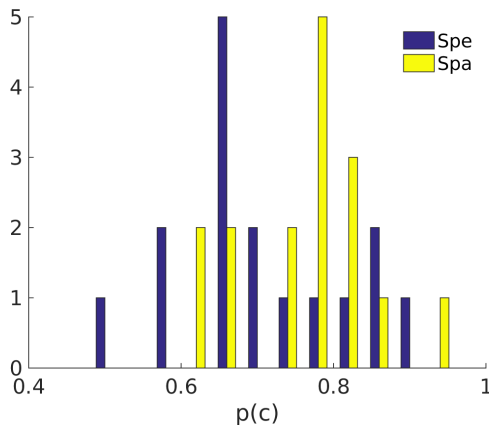


Figure 9: Histogram of the classification performance for the spectrogram (Spe) and the spack (Spa) displays. Only the spectrogram display exhibit two modes, suggesting different levels of expertise of the subjects.

7. CONCLUSIONS

In this paper, we proposed a display based on the stacking of the envelopes of logarithmically spaced band pass filters. We have shown qualitatively that this kind of display may have some potential as it conveys nicely the distribution of the energy across time and frequency in a way that is an alternative to the one taken when considering the spectrogram.

When considering two evaluation tasks: 1) identifying the type of instrument played, and 2) identifying at which skill level a saxophone tone is played, the spack display compares favorably to more conventional displays, such as the waveform and spectrogram displays. Subjects reported ease of understanding and quick access to important aspects of the sounds.

Future work will focus on the design of validation tasks for the spack display using a wider range of audio data, namely speech and environmental data.

As the spack display is both compact and intuitive, it can be considered as an inspection tool while practicing a musical instrument in order to monitor the control of the nuance and the timbre while playing. Evaluation of the spack display in such a training use case would thus be of interest.

8. ACKNOWLEDGMENTS

The authors would like to acknowledge support for this project from ANR project Houle (grant ANR-11-JS03-005-01) and ANR project Cense (grant ANR-16-CE22-0012).

9. REFERENCES

- [1] Trevor R Agus, Clara Suied, Simon J Thorpe, and Daniel Pressnitzer. Fast recognition of musical sounds based on timbre. *The Journal of the Acoustical Society of America*, 131(5):4124–4133, 2012.
- [2] Joachim Anden and Stephane Mallat. Multiscale Scattering for Audio Classification. In *ISMIR*, 2011.
- [3] L Byron and M Wattenberg. Stacked Graphs-Geometry & Aesthetics. *IEEE Trans. Vis. Comput. Graph.*, 2008.
- [4] Taishih Chi, Powen Ru, and Shihab Shamma. Multiresolution spectrotemporal analysis of complex sounds. *The Journal of the Acoustical Society of America*, 118(2):887, 2005.
- [5] Torsten Dau, Birger Kollmeier, and Armin Kohlrausch. Modeling auditory processing of amplitude modulation. i. detection and masking with narrow-band carriers. *The Journal of the Acoustical Society of America*, 102(5):2892–2905, 1997.
- [6] John M Grey. Multidimensional perceptual scaling of musical timbres. *the Journal of the Acoustical Society of America*, 61(5):1270–1277, 1977.
- [7] W. Kandinsky. *Concerning the spiritual in art*. Dover publications, 1954.
- [8] M McLuhan. *The Gutenberg Galaxy*. University of Toronto Press, 1963.
- [9] Matthias Robine and Mathieu Lagrange. Evaluation of the technical level of saxophone performers by considering the evolution of spectral parameters of the sound. In *ISMIR*, pages 79–84, 2006.
- [10] P Schaeffer. *Traité des objets musicaux*. Éditions Du Seuil, 1966.
- [11] RM Schafer. *The soundscape: Our sonic environment and the tuning of the world*. Destiny books, Rochester, Vermont, 1977.
- [12] SS Stevens, J. Volkman, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 185(8), 1937.
- [13] E.R. Tufte. *The Visual Display of Quantitative Information*, volume 7. Graphics press Cheshire, CT, 1983.
- [14] Xiaowei Yang, Kuansan Wang, and Shihab A Shamma. Auditory representations of acoustic signals. *IEEE transactions on information theory*, 38(2):824–839, 1992.

TWO WEB APPLICATIONS FOR EXPLORING MELODIC PATTERNS IN JAZZ SOLOS

Klaus Frieler^{1*} Frank Höger¹ Martin Pfeiderer¹ Simon Dixon²

¹ Institute for Musicology, University of Music “Franz Liszt” Weimar, Germany

² Center for Digital Music, Queen Mary University of London, UK

*Please direct correspondence to: klaus.frieler@hfm-weimar.de

ABSTRACT

This paper presents two novel user interfaces for investigating the pattern content in monophonic jazz solos and exemplifies how these interfaces could be used for research on jazz improvisation. In jazz improvisation, patterns are of particular interest for the analysis of improvisation styles, the oral transmission of musical language, the practice of improvisation, and the psychology of creative processes. The ongoing project “Dig That Lick” is devoted to addressing these questions with the help of a large database of jazz solo transcriptions generated by automated melody extraction algorithms. To expose these transcriptions to jazz researchers, two prototypes of user interfaces were designed that work currently with the 456 manually transcribed jazz solos of the Weimar Jazz Database. The first one is a Shiny application that allows exploring a set of 653 of the most common patterns by eminent players. The second one is a web interface for a general two-staged pattern search in the Weimar Jazz Database featuring regular expressions. These applications aim on the one hand at an expert audience of jazz researchers to facilitate generating and testing hypotheses about patterns in jazz improvisation, and on the other hand at a wider audience of jazz teachers, students, and fans.

1. INTRODUCTION

Music Information Retrieval offers exciting options for musicological research, particularly for methodologies which are hard (or impossible) to carry out manually, e. g., large corpus studies and measuring acoustical properties. One such field of application is the mining of patterns. Patterns – and repetitions in general – play an important role in nearly all music styles [10] and are thus of interest for many sub-fields of musicology. In particular, they form a crucial component of jazz improvisation [1, 13, 14]. The concept of ‘pattern’ can be defined in different ways and appears under different names in the literature. The more

formal definition as ‘repeated sub-sequences’ (over a suitable sequence space) contrasts with more specific usages in jazz theory and practice, where patterns are often called ‘formulas’, ‘licks’, ‘stock-phrases’, and ‘riffs’. The main differences between these terms lie in their supposed origin, their function, and their musical characteristics.

A formula is mostly understood as a rather short pattern, which is well-rehearsed by an improviser. A formula is generally not musically autonomous, i. e., it can be rhythimized differently or embedded in other formulas to make longer phrases (e. g., John Coltrane’s solo on “Giant Steps” [9]). The term ‘lick’ (or stock-phrase) usually refers to a melodic unit with a distinctive recognizable character. In some cases, licks trace their origin to an individual performer or even to a single solo. For example, Charlie Parker created many licks that were used by other jazz musicians [14]. In most cases, however, licks cannot be attributed to a single originator. Thus, they form specific music vocabularies of smaller and wider scope. A riff can be regarded as a lick which is constantly repeated as an accompaniment and has thus a different musical function than a normal lick [12]. Other special cases are short quotations of popular tunes which are often used to humorous effect or for the cultural practice of inter-textuality (‘intermusicality’), or ‘signifyin’ [11].

Given the significance of patterns and licks in jazz, several research questions arise. Some concern historical issues, e. g., the oral tradition of licks and the development of a typical jazz language; some are of a more systematic nature, e. g., the psychology of the creative process, where patterns and formulas can be regarded as necessary to accomplish the highly virtuoso feat of modern jazz improvisation. Some of these research questions are:

- To what extent are patterns and licks used to shape an improvisation?
- When and by whom are patterns and licks created and how are they transmitted between players over time (*pattern archeology*)?
- Does pattern usage change with time and styles?
- Is there an influence of jazz education on pattern usage (e. g., by published pattern collections)?
- How are patterns used to build phrases, e. g., to construct a typical bebop line?



© Klaus Frieler, Frank Höger, Martin Pfeiderer, Simon Dixon. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Klaus Frieler, Frank Höger, Martin Pfeiderer, Simon Dixon. “Two web applications for exploring melodic patterns in jazz solos”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

- Which role do external musical influences such as quotes and signifying references play in jazz improvisation?

In this paper, two web tools which could help to address these questions are introduced. First, the research background and some similar tools are discussed (Sect. 2), before we describe the tools in detail in Sect. 3, including two use case examples. The web applications are still prototypes under active development, but they are already helpful to make some interesting observations which will be reported in Sect. 4. Thoughts on future prospects of these tools conclude the paper (Sect. 5).

2. BACKGROUND

The project “Dig That Lick: Analysing Large-Scale Data for Melodic Patterns in Jazz Performances” (DTL) is a two-year project within the fourth “T-AP Digging Into Data Challenge”.¹ It sets out to investigate some of the aforementioned research questions using an interdisciplinary approach combining musicology, computer science, MIR, and jazz research. The project aims, first, at developing tools for pattern mining on symbolic as well as audio data, and, second, at understanding psychological and social aspects of patterns and licks in jazz. The development of appropriate tools consists of three main pillars:

1. Automatic transcription of jazz solo improvisations from audio informed by discographic metadata.
2. Pattern mining and search in the melody transcriptions.
3. Development of suitable user interfaces.

Since the project is still in its initial phase, we will focus in this paper on the second and third issues by describing two prototypes of user interfaces for pattern mining in the Weimar Jazz Database [18] which was created by the Jazzomat Research Project [19]².

In recent years, several scientific or commercial web-based melody search engines with interfaces for different databases of different provenance and quality have been implemented, all of which are scored-based. A web search showed that many of these projects are now defunct or discontinued. To name a few: C-Brahms (defunct), Midomi (discontinued, but seemingly functional), Hymnar (active, only hymns), Mutopia, Music N-gram Viewer (discontinued, but functional), Best Classical Tunes (outdated, but functional), and Melody Search (discontinued, functionality unclear due to Flash player issues). The large number of abandoned sites suggests that melody search is not very popular with a general audience. Some more recent sites, though, aim at musicological experts, e. g., the Troubadour Melodies Database, Global Chant, and Cantus Manuscript Database, which provide simple but efficient search interfaces to specialized corpora.

An older but still functional melody search engine is *Themefinder*³, which interfaces with some large databases of folk and classical music in ***kern* format. The search works well and is fast, though it does not offer metadata filters, regular expressions, or a multi-staged search.

Musipedia⁴ is branded as a “Wikipedia for Music” and is based on a user-generated database of melodies. Search queries are given in Lilypond format, but a piano-like user interface to enter queries with the help of the mouse is also provided. The search is based on similarity matching [22] and, hence, always fuzzy; it is not possible to enforce only exact matches. The result set always comprises full melodies without indicating the matching location for the query. The underlying corpus is not clearly specified, but it seems that some well-known databases such as the Essen Folk Song Collection [21] are incorporated.

Furthermore, Gulati [7] developed a system for melodic pattern discovery in Indian Art Music based on automatically extracted pitch contours and a large set of specialized methods. A demo for browsing patterns in a large audio corpus and a visualization of pattern networks including audio snippets can be found on the accompanying web site⁵.

The investigation of patterns in jazz has rather different requirements compared to those melody search engines. Particularly, the hybrid format of the Weimar Jazz Database, which combines transcriptions with audio, as well as the greater length of jazz solos (as compared to, for instance, incipits, and folk songs) demands fine grained and controlled access to pattern instances. Furthermore, to assist users during exploration, providing scores and audio snippets along with more abstract representations is important in order to connect to established methodological standards in jazz research and practice.

3. TWO PATTERN MINING APPLICATIONS

3.1 The Pattern History Explorer

The main goal of the Pattern History Explorer⁶, an interactive Shiny web application [2], is to enable the exploration of interval patterns in jazz solos by providing information from many different angles. It provides an overview of interval patterns frequently used by a selected subset of performers and traces their usage in the Weimar Jazz Database, allowing for the discovery of cross-artist and cross-temporal relationships.

Currently, 653 interval patterns with 11,630 instances are included. The pattern corpus was created by mining for interval patterns in solos of eminent performers using the partition mode of the *melpat* module from the *MeloSpy-Suite* [6]. Subsequently, all instances of these patterns were searched for in the Weimar Jazz Database, and the results were included in the application. Since the interval distributions of jazz solos are dominated by step-wise motion,

³ <http://www.themefinder.org/>

⁴ <https://www.musipedia.org/>

⁵ <http://compmusic.upf.edu/node/304>

⁶ https://jazzomat.hfm-weimar.de/pattern_history/

¹ <http://dig-that-lick.eecs.qmul.ac.uk/>

² <https://jazzomat.hfm-weimar.de>

a certain number of patterns can be expected by chance alone, even more so by assuming Markov processes of first or higher order [4]. Therefore, the following restriction to a minimum number of instances and sources was imposed to ensure significance of the patterns: interval patterns were limited to be of no fewer than six elements occurring in at least three different solos of at least one musician. According to previous investigations [4], this length seems to be a critical point for pattern distributions. The number of instances of each pattern depends partly on the amount of musical material available. For Bob Berg the criterion was relaxed to patterns of at least seven elements occurring at least twice in two different solos, since from a former study [5] it was already known that many interesting and highly peculiar patterns occur only twice in the subcorpus of Berg's solos in the Weimar Jazz Database. Another exception was Charlie Parker, for whom the source patterns were extracted not from the Weimar Jazz Database but from the Omnibook, a collection of 56 transcriptions of his solos. In order to find only Parker's more eminent patterns, a criterion of at least six elements occurring in at least ten different solos was applied.

In general, the user of the Pattern History Explorer selects a certain interval pattern from the overall set of 653 patterns. Several options are available in order to filter the pattern set or to change the ordering of the patterns according to several criteria (e. g., filtering by performer, length, intrinsic characteristics such as Huron contour [8] or tonality type, or content). For the selected pattern, various kinds of information can be accessed in the following tabs:

- **Listen & See.** A sortable list of all instances of the pattern in the Weimar Jazz Database is displayed. It includes metadata such as name of the performer, title of the solo, year of recording, metrical and start position. In one column the tonal context is displayed as a combination of chord context and extended chordal pitch class values (CDPCX, cf. [6]). Most importantly, score snippets and audio links allow for visual and aural inspection. Here, and in the following tabs, cross-links to the Pattern Search web application (see below) is provided to allow more refined searches.
- **Instances.** Further information about the instances can be found here. A rhythmic encoding based on absolute inter-onset interval (IOI) class (very short, short, medium, long, very long), the starting pitch, the chord context, the CDPCX value and a binary vector indicating the position of metrical accents in the pattern are displayed. Additionally, some of the metadata from the Listen & See tab are repeated. The final column indicates whether the instance is contained in one single phrase. This table is also sortable.
- **Stats.** Musical characteristics and statistical information of the pattern are compiled. Due to space limitations we cover just some of these here, and refer the interested reader to the online documentation

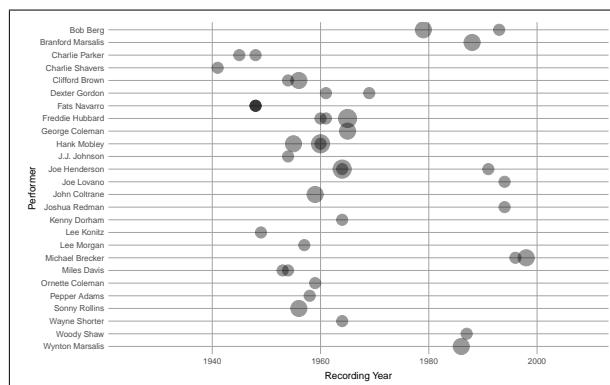


Figure 1. History plot of the pattern [-2, -2, -1, -2, -2, -1, -1]. The first instance can be found in a solo by Charlie Shavers on “Limehouse Blues” from 1947. A cluster can be identified in the years around 1960. The circle radius represents the frequencies of patterns in each solo in that year.

for full details. One important feature is *log excess probability*, which is defined as $\log_{10} \frac{p_o}{p_e}$, where p_o is the observed frequency of a certain pattern within the database and p_e is its expected frequency according to a Markov model of zeroth order taken from the global interval distribution. Not surprisingly, log excess probability is strongly correlated with pattern length ($r = .82$, $p < .001$). Another interesting feature is the number of different starting pitches, as this indicates whether an interval pattern is indeed a pitch pattern. A pitch pattern with a fixed set of pitches might be based on a physiological motor pattern, i. e., tied to a specific fingering on the instrument. Other interesting information provided here is which soloists favor this pattern and who played it first. A list of instances by performer can be found at the bottom of the page.

- **Timeline.** This page contains a visualization of the distribution of pattern instances over time and performer. See Fig. 1 for a sample plot for the pattern [-2, -2, -1, -2, -2, -1, -1] (a descending diatonic line with a chromatic ending). The first instance can be found in a solo by Charlie Shavers from 1947 and it is favored by Hank Mobley with six instances in three different solos, mostly starting on the supertonic (second scale step) over a ii^7-V^7 transition. This tonal interpretation is preferred in many other instances. Other players who use this pattern are Joe Henderson and Freddie Hubbard with five instances each. As the plot shows, this pattern is spread widely over jazz history since its first occurrence.

Besides the tabs related to patterns, the tab ‘Info’ provides general information on the Pattern History Explorer and the ‘Help’ tab contains detailed documentation. The tab ‘General Stats’ collects plots pertaining to statistics of all included patterns and instances (e. g., Fig. 3).

3.2 Pattern Search

While pre-computing a set of patterns is helpful in regard to the exploratory approach of the Pattern History Explorer, searching for instances of arbitrary patterns of any length and frequency of occurrence within a database requires a different approach. Although it is possible to search the Weimar Jazz Database with its accompanying software MeloSpySuite and MeloSpyGUI, our web-based pattern search interface⁷ provides most of the functionality of the `melpat` search module while also extending it with audio and score snippets (both as isolated patterns and within their melodic context) for visual and aural inspection.

To execute a basic search, the user has to enter a pattern as a space or comma separated list of elements and choose a corresponding transformation, that is, a mathematical mapping of the basic melodic representation, also known as viewpoints [3]. Currently, ten pitch-related transformations for primary search are offered (e. g., MIDI pitch, semitone intervals, CDPCX). An additional 18 transformations, such as duration, IOI classes and various structural markers, are supplied for the optional secondary search. Additionally, the search space can be constrained by seven metadata categories, like performer, style, or recording year. Search patterns can be regular expressions (in a specific hybrid syntax depending on the selected transformation)⁸ which allows searches for variants in a single run. The secondary search can be used to refine the result space, e. g., by filtering out certain rhythmic or metrical configurations or by confining instances to a single phrase. Since the last constraint is used frequently, there is also a shortcut checkbox that fills in the correct secondary search pattern (which is based on phrase boundary markers). The user also has the option to request inclusion of up to 20 tones before and after the actual pattern instance in both score and audio files. In order to generate these, the corresponding checkboxes have to be selected, which is the default. There might be cases though, where the result set is very large (e. g., searching for very short intervals), and one wants to avoid generating all audio and score files as it would take a considerable amount of time. If both checkboxes are disabled, instances will appear in the result table after a few seconds which allows for a first examination of the results. The generation of audio and score files is also suppressed (with a warning being shown) when the result set exceeds the (current) limit of 100 instances. Finally, by clicking the ‘Display whole phrase’ button, the score of the whole phrase containing the pattern is displayed.

The underlying search algorithm is built upon the basic Python regular expression module, which is fed with virtual Unicode strings constructed from the different melodic representations (transformations) with different alphabets. Scores are generated with the help of Lilypond, while audio snippets are directly extracted from the solo audio files. On average, a full search including audio and score gener-

ation takes several minutes, depending strongly on the size of the result set. If the results can be found in the cache, they are returned in a few seconds.

Finally, a documentation page⁹ is included as well as a search history which gives a (browser-local) overview of all distinct search requests which have been submitted by the user. For each specific search, a comment can be added and by clicking on the ‘Restore Search’ link the result of the corresponding search is displayed.

3.3 Example 1: A typical Parker lick

Assume that we want to find all occurrences of interval pattern [-1, -2, -1, -9, 3, 3, -1, -2] which was often played by Charlie Parker within various recordings [14]. For this, we enter it into the primary pattern field and select ‘Semitone intervals’ as the primary transformation. By executing the search we get eight instances, six by Charlie Parker, and one each by Dexter Gordon and Sonny Stitt, who are both known to be strongly influenced by Charlie Parker [15].

3.4 Example 2: Hunting for Coltrane’s “Giant Steps” pattern

In his improvisations on “Giant Steps” recorded in 1959, seminal tenor saxophonist John Coltrane repeatedly uses a four-tone pattern that consists of the root, the supertonic, the third and the fifth of the underlying chord [9, 20]. It would be interesting to know whether other jazz musicians have used this simple pattern, and if Coltrane used it on other recordings as well. Additionally, it is interesting to know whether the pattern is only played over major chords or also with minor chords. While the Pattern History Explorer currently covers only patterns with seven tones (six intervals) or more, the Pattern Search application offers several options to search for arbitrary patterns using various transformations and filters.

The pattern can be expressed, for example, with the transformation ‘Chordal Pitch Class’ (CPC), which maps pitches to pitch classes starting with the root of the underlying chord. Since the third of a chord can be either major (CPC = 4) or minor (CPC = 3), we use regular expression syntax to reflect this and the search pattern would be

0, 2, "[, 3, 4, "], 7

This translates as “the root (0) followed by the supertonic (2) followed by either the minor (3) or the major third (4) followed by the fifth of the chord (7)”¹⁰. Note that special symbols of regular expressions, here the square brackets as character set indicators, must be quoted, because of the hybrid syntax, where chordal pitch classes are expressed as integers (not characters).

Hitting the search button yields 323 instances. Because the limit of 100 instances is exceeded, no score or audio file are generated and only the result list together with a

⁷ https://dig-that-lick.hfm-weimar.de/pattern_search

⁸ https://jazzomat.hfm-weimar.de/commandline_tools/melpat/melpat.html#search-pattern-syntax

⁹ https://dig-that-lick.hfm-weimar.de/pattern_search/documentation

¹⁰ Alternatively, the same search could be carried out using the pattern 1235 over the extended chordal diatonic pitch class (CDPCX) representation.

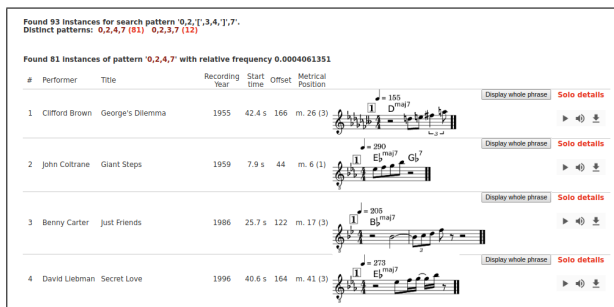


Figure 2. The first four search results for a pattern with chordal pitch class transformation using regular expression 0, 2, "[", 3, 4, "]", 7 and secondary search pattern 1, 0, ".", 0 over primary and secondary metrical accents.

warning is displayed. To prune the result set, one can tick the ‘Within Single Phrase’ box, which results in 297 instances. Characteristic of Coltrane’s use of the pattern is that it very often starts on a strong beat (first or third beat in 4/4 time). To express this constraint with a secondary search, we use the transformation ‘Primary and secondary metrical accent’, which takes on the values ‘1’ for an event on a beat position and ‘0’ otherwise, and the search pattern

1, ". {3} "

with the operation ‘Match’. The dot stands for any symbol (here only ‘0’ or ‘1’), while the number 3 in braces is a quantifier meaning “exactly three repetitions”. Again, the quotes are necessary here because of the hybrid syntax. This leads to all CPC patterns that start on a strong beat, whereby the last three tones can lie on arbitrary metrical positions. This results in 93 instances (by 37 players in 58 different solos), which are displayed as a complete list (Fig. 2). There are 12 instances with a minor third and 83 instances with major. Interestingly, John Coltrane never uses the minor version. 26 instances originate from him, from which 18 can be found in two recordings of “Giant Steps”. Michael Brecker, who is said to be heavily influenced by John Coltrane, accounts for nine instances, all of which are also major.

Visual inspection of the results shows, however, that some of the instances are either not over one single chord or do not follow an ascending step-wise motion. Similarly, even though most of the instances feature a plain motion in eighth notes, there are a few instances with rather different rhythms which still satisfy our metrical constraint. To filter these cases, one could use other transformations, e.g., searching for duration patterns. Thus, here and in other situations, a tertiary or even quaternary search stage would be needed, which is not available yet but planned for a future version of the Pattern Search application.

4. SOME OBSERVATIONS AND A HYPOTHESIS

In order to demonstrate the usefulness of the two systems in the context of jazz research, we like to report some first observations.

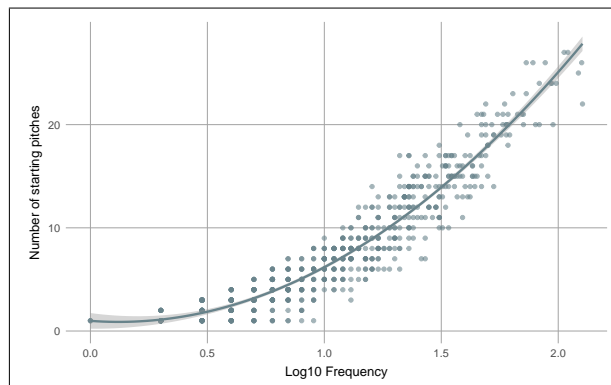


Figure 3. Number of different starting pitches versus \log_{10} of pattern frequency. The correlation is $r = -.92$. The fit is a quadratic polynomial.

Observation 1 *Pattern usage varies with performers and appeared with bebop.*

Looking at the distributions of patterns in the Pattern History Explorer with respect to performer, it seems clear that different jazz improvisers have different levels of pattern usage. For example, players from earlier styles (e.g., New Orleans, Swing) have far fewer long interval patterns in their repertoire than later players. This might be partly due to the fact that these performers are seldom playing long lines – a practice that only became widespread with the advent of bebop, and which probably made the usage of patterns a necessity.

Observation 2 *The more frequent an interval pattern, the more tonally flexible it is.*

As shown in Fig. 3, the number of distinct starting pitches of a certain pattern generally increases with its frequency of occurrence. The relationship is approximately logarithmic in the frequency $N_p \propto \log f$ with a strong correlation of $r = .92$ ($p < .001$). In other words, the more frequent (and shorter) a pattern, the more tonally flexible it is. This seems to reflect typical rehearsal routines, where shorter patterns are more likely to be practiced in all keys whereas long and very long patterns are designed to fit in only one or two tonal contexts, e.g., to chord changes of specific songs. This has to be tested on a larger database, taking into account that not all keys, chords, and chord combinations are equally likely to occur in jazz.

Observation 3 *Patterns are mostly simple and reflect common rehearsal routines.*

Diatonic patterns, i.e., step-wise motions, are by far the most frequent pattern type followed by chromatic patterns. Together they account for about 78% of all patterns included in the Pattern History Explorer. This implies that the main share of patterns is musically rather simple, i.e., built from diatonic scales, chromatic runs, and arpeggios. One can conjecture that this is a result of practice traditions in which scales and arpeggios are rehearsed for technical

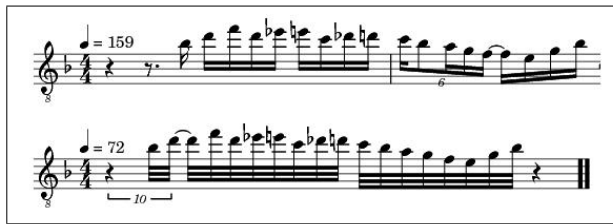


Figure 4. The interval pattern [4, 3, -3, 1, 1, -4, 1, 1, -2, -2, -1, -2, -2, -1, 3, 3] of length 16 as found in two solos by Charlie Parker.

fluency, but might end up ‘to lie in the fingers’, i. e., as motor patterns.

Hypothesis 1 *Jazz solos are hierarchically composed of adaptive chunks.*

A general problem of pattern mining in jazz solos and other melody corpora is, however, to distinguish truly meaningful (i. e., intended) patterns from randomly occurring patterns [7]. This is closely connected to the problem of finding adequate models for the underlying (random) processes, for which some evidence can be extracted already from the data.

Some very long patterns can be found, e. g., a pattern of 16 tones by Charlie Parker with two instances in two different solos which are also tonally and rhythmically very similar (Fig. 4). The *a priori* probability under any Markov model for this is practically zero. This clearly shows that this pattern was preconceived and then reproduced as a single chunk. In general, it seems very doubtful that jazz solos could be successfully modeled with Markov chains (of any order) on the level of single tones or intervals [16, 17].

This is also corroborated by the existence of (non-trivial) trill-like patterns (see Fig. 5 for an example), which are amongst the longest patterns that can be found. These patterns are somewhat trivial, as they are “oscillations” of a repeated shorter pattern, but they are also examples of “super-patterns”, i. e., long patterns containing shorter sub-patterns. This hints at Markov models not working on the note event level but hierarchically on a chunk level.

To sum up, these first observations suggest that jazz solos do not follow Markov models of any order on a tone-level, but are rather created by hierarchical processes with interspersed “islands of high probability”, where patterns are reproduced as complete chunks while being adapted rhythmically and tonally to the current context. This strategy could be dubbed “mutatis mutandis”, which seems to be basic not only for improvisation but music creation in general.

5. CONCLUSION & OUTLOOK

Both applications presented in this paper are already usable interfaces for the Weimar Jazz Database and serve as prototypes for applications to explore large databases, which are going to be automatically extracted from large collections of jazz recordings. Both tools can be primarily



Figure 5. The interval pattern [-2, -2, -1, 5, -2, -2, -1, 5, -2, -2, -1, 5, -2, -2, -1] of length 19 as found in two solos by Bob Berg on the album ‘Enter the Spirit’ from 1993.

viewed as bespoke interfaces for the specific needs of jazz researchers, but they could also be of interest to jazz teachers, students and fans, as well as for training courses in computational music analysis. Compared to the possibilities of the *melpat* module of the *MeloSpySuite*, they provide superior presentation of results, particularly in their provision of audio and score excerpts.

The development of both applications is still ongoing. The Pattern History Explorer will be augmented by more patterns in the future. For the Pattern Search application, removing the limit of 100 instances for full searches and speed improvements are already under construction. Another significant extension would be the implementation of arbitrarily many search stages, since the current status of only two stages is often too restrictive. The incorporation of other databases, such as the Essen Folk Song Collection, would also be feasible without major modifications.

The current system is based on the Python regular expression module and requires all melodies to be loaded into memory at once. This is sufficiently fast for the Weimar Jazz Database with 200,000 events. For searching larger data sets, however, a more advanced retrieval technology is needed, e. g., distributed NoSQL databases and sophisticated search algorithms. Moreover, the generation of score files could be optimized by switching from Lilypond to VexFlow¹¹, not only to speed up score rendering but also to allow for score customization.

Some features in the applications are only possible due to the high-quality manual transcriptions in the Weimar Jazz Database with its comprehensive set of annotations. In the scenario of automatically extracted transcriptions, several curtailments can be expected, e. g., transformations that depend on such annotations might not be usable. However, it is always possible to use pitch and interval transformations and to extract audio snippets for aural control, providing also feedback for the transcription algorithm.

Finally, it is planned to conduct user studies to gather feedback for further improvements of the interface.

6. ACKNOWLEDGEMENT

The “Dig that Lick” project is supported by the Deutsche Forschungsgemeinschaft (DFG, PF669/9-1) and the UK Economic and Social Research Council (ES/R004005/1).

¹¹ <http://www.vexflow.com/>

7. REFERENCES

- [1] Paul F. Berliner. *Thinking in jazz. The infinite art of improvisation*. University of Chicago Press, Chicago, 1994.
- [2] Winston Chang, Joe Cheng, J. J. Allaire, Yihui Xie, and Jonathan McPherson. *shiny: Web Application Framework for R*. 2017.
- [3] Darrell Conklin and Christina Anagnostopoulou. Representation and discovery of multiple viewpoint patterns. In *Proceedings of the 2001 International Computer Music Conference*, San Francisco, 2001. ICMA.
- [4] Klaus Frieler. Pattern usage in monophonic jazz solos, 2014. Talk given at the International Jazzomat Research Workshop, September, 26–27, 2014.
- [5] Klaus Frieler. Bob Berg’s Solo on “Angles”. In Martin Pfeleiderer, Klaus Frieler, Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart, editors, *Inside the Jazzomat. New Perspectives for Jazz Research.*, pages 41–84. Schott-Campus, Mainz, 2017.
- [6] Klaus Frieler. Computational melody analysis. In Martin Pfeleiderer, Klaus Frieler, Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart, editors, *Inside the Jazzomat. New Perspectives for Jazz Research.*, pages 41–84. Schott-Campus, Mainz, 2017.
- [7] Sankalp Gulati. *Computational Approaches for Melodic Description in Indian Art Music Corpora*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2016.
- [8] David Huron. The melodic arch in western folksongs. *Computing in Musicology*, 10:3–23, 1996.
- [9] A. C. Lehmann and S. Goldhahn. Duration of playing bursts and redundancy of melodic jazz improvisation in John Coltranes “Giant Steps”. *Musicae Scientiae*, 20(3):345–360, September 2016.
- [10] Elizabeth Hellmuth Margulis. *On repeat: how music plays the mind*. Oxford University Press, New York, NY, 2014.
- [11] Ingrid Monson. *Saying something. Jazz improvisation and interaction*. University of Chicago Press, Chicago, 1996.
- [12] Ingrid Monson. Riffs, Repetition, and Theories of Globalization. *Ethnomusicology*, 43:31–65, 1999.
- [13] Martin Norgaard. How jazz musicians improvise. The central role of auditory and motor patterns. *Music Perception: An Interdisciplinary Journal*, 31(3):271–287, 2014.
- [14] Thomas Owens. *Charlie Parker. Techniques of improvisation*. PhD thesis, University of California, Los Angeles, 1974.
- [15] Thomas Owens. *Bebop. The music and its players*. Oxford University Press, New York, 1995.
- [16] Francois Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, 2003.
- [17] François Pachet and Pierre Roy. Markov constraints: steerable generation of Markov sequences. *Constraints*, 16(2):148–172, April 2011.
- [18] Martin Pfeleiderer. The Weimar Jazz Database. In Martin Pfeleiderer, Klaus Frieler, Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart, editors, *Inside the Jazzomat. New Perspectives for Jazz Research*. Schott-Campus, Mainz, 2017.
- [19] Martin Pfeleiderer, Klaus Frieler, Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart, editors. *Inside the Jazzomat: New Perspectives for Jazz Research*. Schott Music GmbH & Co. KG, Mainz, 1 edition, 2017. OCLC: 1015349144.
- [20] Lewis Porter. *John Coltrane, his life and music*. University of Michigan Press, Ann Arbor, 1998.
- [21] H. Schaffrath. The Essen folksong collection. In D. Huron, editor, *Database containing 6,255 folksong transcriptions in the Kern format and a 34-page research guide*. CCARH, Menlo Park, CA, 1995.
- [22] Frans Wiering, Rainer Typke, and Remco C. Veltkamp. Transportation Distances and their Application in Music-Notation Retrieval. In *Music Query: Methods, Strategies, and User Studies*, volume 13 of *Computing in Musicology*, pages 113–128. 2004.

LEARNING TO LISTEN, READ, AND FOLLOW: SCORE FOLLOWING AS A REINFORCEMENT LEARNING GAME

Matthias Dorfer*

Florian Henkel*

Gerhard Widmer*[†]

*Institute of Computational Perception, Johannes Kepler University Linz, Austria

[†]The Austrian Research Institute for Artificial Intelligence (OFAI), Austria

matthias.dorfer@jku.at

ABSTRACT

Score following is the process of tracking a musical performance (audio) with respect to a known symbolic representation (a score). We start this paper by formulating score following as a multimodal Markov Decision Process, the mathematical foundation for sequential decision making. Given this formal definition, we address the score following task with state-of-the-art deep reinforcement learning (RL) algorithms such as synchronous advantage actor critic (A2C). In particular, we design multimodal RL agents that simultaneously learn to listen to music, read the scores from images of sheet music, and follow the audio along in the sheet, in an end-to-end fashion. All this behavior is learned entirely from scratch, based on a weak and potentially delayed reward signal that indicates to the agent how close it is to the correct position in the score. Besides discussing the theoretical advantages of this learning paradigm, we show in experiments that it is in fact superior compared to previously proposed methods for score following in raw sheet music images.

1. INTRODUCTION

This paper addresses the problem of score following in sheet music images. The task of an automatic score following system is to follow a musical performance with respect to a known symbolical representation, the score (cf. Figure 1). In contrast to audio-score alignment in general [20], all of this takes place in an on-line fashion. Score following itself has a long history in Music Information Retrieval (MIR) and forms the basis for many subsequent applications such as automatic page turning [2], automatic accompaniment [6, 23] or the synchronization of visualizations to the live music during concerts [1, 22].

Traditional approaches to the task depend on a symbolic, computer-readable representation of the score, such as MusicXML or MIDI (see e.g. [1, 6, 10, 14, 16, 17, 21–23]). This representation is created either manually (e.g. via the

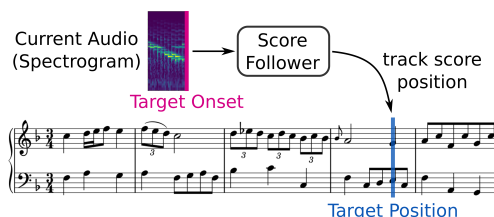


Figure 1. Sketch of score following in sheet music. Given the incoming audio, the score follower has to track the corresponding position in the score (image).

time-consuming process of (re-)setting the score in a music notation program), or automatically via optical music recognition software [3, 4, 12]. However, automatic methods are still unreliable and thus of limited use, especially for more complex music like orchestra pieces [26].

To avoid these complications, [7] proposes a multimodal deep neural network that directly learns to match sheet music and audio in an end-to-end fashion. Given short excerpts of audio and the corresponding sheet music, the network learns to predict which location in the given sheet image best matches the current audio excerpt. In this setup, score following can be formulated as a multimodal localization task. However, one problem with this approach is that successive time steps are treated independently from each other. We will see in our experiments that this causes jumps in the tracking process especially in the presence of repetitive passages. A related approach [8] trains a multimodal neural network to learn a joint embedding space for snippets of sheet music and corresponding short excerpts of audio. The learned embedding allows to compare observations across modalities, e.g., via their cosine distance. This learned cross-modal similarity measure is then used to compute an off-line alignment between audio and sheet music via dynamic time warping.

Our proposal is inspired by these works, but uses a fundamentally different machine learning paradigm. The central idea is to interpret score following as a *multimodal control problem* [9] where the agent has to navigate through the score by adopting its reading speed in reaction to the currently playing performance. To operationalize this notion, we formulate score following as a *Markov Decision Process (MDP)* in Section 3. MDPs are the mathematical foundation for sequential decision making and permit



© Matthias Dorfer, Florian Henkel, Gerhard Widmer. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Matthias Dorfer, Florian Henkel, Gerhard Widmer. “Learning to Listen, Read, and Follow: Score Following as a Reinforcement Learning Game”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

us to address the problem with state-of-the-art *Deep Reinforcement Learning* (RL) algorithms (Section 4). Based on the MDP formulation, we design agents that consider both the score and the currently playing music to achieve an overall goal, that is to track the correct position in the score for as long as possible. This kind of interaction is very similar to controlling an agent in a video game, which is why we term our MDP the *score following game*; it is in fact inspired by the seminal paper by Mnih et al. [19] which made a major contribution to the revival of deep RL by achieving impressive results in a large variety of Atari games. In experiments with monophonic as well as polyphonic music (Section 5), we will show that the RL approach is indeed superior to previously proposed score following methods [7]. The code for the score following game is available at https://github.com/CPJKU/score_following_game.

2. DESCRIPTION OF DATA

To set the stage, we first need to describe the kind of data needed for training and evaluating the multimodal RL score following agents. We assume here that we are given a collection of piano pieces represented as pairs of audio recordings and sheet music images. In order to train our models and to later quantify the score following error, we first need to establish correspondences between individual pixel locations of the note heads in a sheet and their respective counterparts (note onset events) in the respective audio recordings. This has to be done either in a manual annotation process or by relying on synthetic training data which is generated from digital sheet music formats such as *Muscore* or *Lilypond*. As this kind of data representation is identical to the one used in [7, 8] we refer to these works for a detailed description of the entire alignment process.

3. SCORE FOLLOWING AS A MARKOV DECISION PROCESS

Reinforcement learning can be seen as a computational approach to learning from interaction to achieve a certain predefined goal. In this section, we formulate the task of score following as a *Markov Decision Process* (MDP), the mathematical foundation for reinforcement learning or, more generally, for the problem of sequential decision making¹. Figure 2 provides an overview of the components involved in the score following MDP.

The *score following agent* (or *learner*) is the active component that interacts with its *environment*, which in our case is the score following game. The interaction takes place in a closed loop where the environment confronts the agent with a new situation (a *state* S_t) and the agent has to respond by making a decision, selecting one out of a predefined set of possible *actions* A_t . After each action taken the agent receives the next state S_{t+1} and a numerical *reward signal* R_{t+1} indicating how well it is doing in achieving the overall goal. Informally, the agent's goal in our case is to track a performance in the score

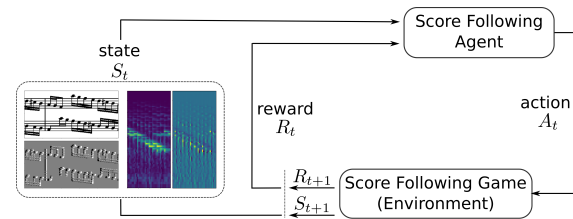


Figure 2. Sketch of the score following MDP. The agent receives the current state of the environment S_t and a scalar reward signal R_t for the action taken in the previous time step. Based on the current state it has to choose an action (e.g. decide whether to increase, keep or decrease its speed in the score) in order to maximize future reward by correctly following the performance in the score.

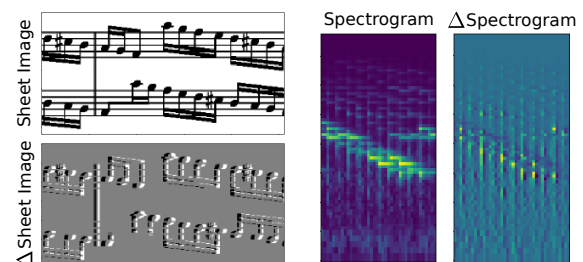


Figure 3. Markov state of the score following MDP: the current sheet sliding window and spectrogram excerpt. To capture the dynamics of the environment we also add the one step differences (Δ) wrt. the previous time step (state).

as accurately and robustly as possible; this criterion will be formalized in terms of an appropriate reward signal in Section 3.3 below. By running the MDP interaction loop we end up with a sequence of states, actions and rewards $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \dots$, which is the kind of experience a RL agent is learning its behavior from. We will elaborate on different variants of the learning process in Section 4. The remainder of this section specifies all components of the score following MDP in detail. In practice, our MDP is implemented as an environment in OpenAI-Gym², an open source toolkit for developing and comparing reinforcement learning algorithms.

3.1 Score Following Markov States

Our agents need to operate on two different inputs at the same time, which together form the state S_t of the MDP: input modality one is a sliding window of the sheet image of the current piece, and modality two is an audio spectrogram excerpt of the most recently played music (~ 2 seconds). Figure 3 shows an example of this input data for a piece by J.S. Bach. Given the audio excerpt as an input the agent's task is to navigate through the global score to constantly receive sheet windows from the environment that match the currently playing music. How this interaction with the score takes place is explained in the next subsection. The important part for now is to note that score fol-

¹ The notation in this paper follows the book by Barto and Sutton [25]

² <https://gym.openai.com/>

lowing embodies dynamics which have to be captured by our state formulation, in order for the process to satisfy the Markov property. Therefore, we extend the state representation by adding the one step differences (Δ) of both the score and the spectrogram. With the Δ images and spectrograms a state contains all the information needed by the agent to determine where and how fast it is moving along in the sheet image.

3.2 Agents, Actions and Policies

The next item in the MDP (Figure 2) is the agent, which is the component interacting with the environment by taking actions as a response to states received. As already mentioned, we interpret score following as a multimodal control problem where the agent decides how fast it would like to progress in the score. In more precise terms, the agent controls its score progression speed v_{pxl} in *pixels per time step* by selecting from a set of actions $A_t \in \{-\Delta v_{pxl}, 0, +\Delta v_{pxl}\}$ after receiving state S_t in each time step. Actions $\pm\Delta v_{pxl}$ increase or decrease the speed by a value of Δv_{pxl} pixels per time step. Action $a_1 = 0$ keeps it unchanged. To give an example: a pixel speed of $v_{pxl} = 14$ would shift the sliding sheet window 14 pixels forward (to the right) in the global unrolled score.

Finally, we introduce the concept of a *policy* $\pi_\Theta(a|s)$ to define an agent's behavior. π is a conditional probability distribution over actions conditioned on the current state. Given a state s , it computes an action selection probability $\pi_\Theta(a|s)$ for each of the candidate actions $a \in A_t$. The probabilities are then used for sampling one of the possible actions. In Section 4 we explain how to use deep neural networks as function approximators for policy π_Θ by optimizing the parameters Θ of a policy network.

3.3 Goal Definition: Reward Signal and State Values

In order to learn a useful action selection policy, the agent needs *feedback*. This means that we need to define how to report back to the agent how well it does in accomplishing the task and, more importantly, what the task actually is.

The one component in an MDP that defines the overall goal is the reward signal $R_t \in \mathbb{R}$. It is provided by the environment in form of a scalar, each time the agent performs an action. The *sole objective of a RL agent is to maximize the cumulative reward over time*. Note, that achieving this objective requires foresight and planning, as actions leading to high instantaneous reward might lead to unfavorable situations in the future. To quantify this longterm success, RL introduces the *return* G which is defined as the discounted cumulative future reward: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$. The discount rate γ (with $0.0 < \gamma \leq 1.0$, in our case 0.9) is a hyper-parameter assigning less weight to future rewards if smaller than 1.0.

Figure 4 summarizes the reward computation in our score following MDP. Given annotated training data as described in Section 2, the environment knows, for each onset time in the audio, the true target position x in the score. From this, and the current position \hat{x} of the agent, we compute the current tracking error as $d_x = \hat{x} - x$, and define

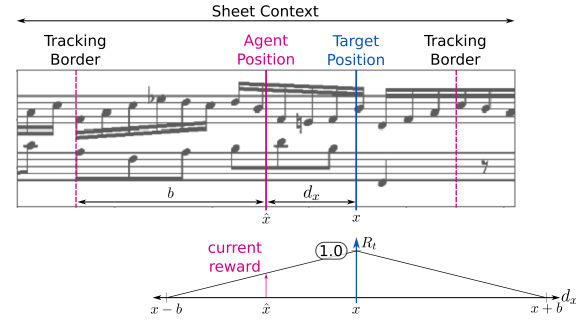


Figure 4. Reward definition in the score following MDP. The reward R_t decays linearly (range $[0, 1]$) depending on the agent's distance d_x to the current true score position x .

the reward signal r within a predefined *tracking window* $[x - b, x + b]$ around target position x as: $r = 1.0 - |d_x|/b$. Thus, the reward per time step reaches its maximum of 1.0 when the agent's position is identical to the target position, and decays linearly towards 0.0 as the tracking error reaches the maximum permitted value b given by the window size. Whenever the absolute tracking error exceeds b (the agent drops out of the window), we reset the score following game (back to start of score, first audio frame). As an RL agent's sole objective is to maximize cumulative future reward, it will learn to match the correct position in the score and to not lose its target by dropping out of the window. We define the target onset, corresponding to the target position in the score, as the *rightmost frame* in the spectrogram excerpt. This allows to run the agents on-line, introducing only the delay required to compute the most recent spectrogram frame. In practice, we linearly interpolate the score positions for spectrogram frames between two subsequent onsets in order to produce a continuous and stronger learning signal for training.

As with policy π , we will use function approximation to predict the future cumulative reward for a given state s , estimating how good the current state actually is. This estimated future reward is termed the *value* $V(s)$ of state s . We will see in the next section how state-of-the-art RL algorithms use these value estimates to stabilize the variance-prone process of policy learning.

4. LEARNING TO FOLLOW

Given the formal definition of score following as an MDP we now describe how to address it with reinforcement learning. Note that there is a large variety of RL algorithms. We focus on *policy gradient methods*, in particular the class of *actor-critic methods*, due to their reported success in solving control problems [9]. The learners utilized are *REINFORCE with Baseline* [27] and *Synchronous Advantage Actor Critic (A2C)* [18, 28], where the latter is considered a state-of-the-art approach. As describing the methods in full detail is beyond the scope of this paper, we provide an intuition on how the methods work and refer the reader to the respective papers.

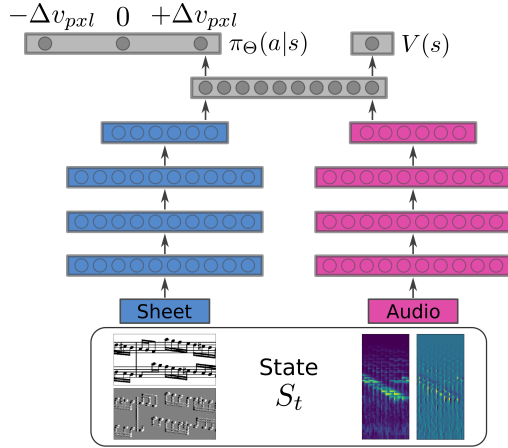


Figure 5. Multimodal network architecture used for our score following agents. Given state s the policy network predicts the action selection probability $\pi_{\Theta}(a|s)$ for the allowed action $A_t \in \{-\Delta v_{pxl}, 0, +\Delta v_{pxl}\}$. The value network, sharing parameters with the policy network, provides a state-value estimate $V(s)$ for the current state.

4.1 Policy and State-Value Approximation via DNNs

In Section 3, we introduced *policy* π_{Θ} , determining the behavior of an agent, and *value function* $V(s)$, predicting how good a certain state s is with respect to cumulative future reward. Actor-critic methods make use of both concepts. The actor is represented by policy π_{Θ} and is responsible for selecting the appropriate action in each state. The critic is represented by the value function $V(s)$ and helps the agent to judge how good the selected actions actually are. In the context of deep RL both functions are approximated via a Deep Neural Network (DNN), termed policy and value network. We denote the parameters of the policy network with Θ in the following.

Figure 5 shows a sketch of such a network architecture. As the authors in [7], we use a multimodal convolutional neural network operating on both sheet music and audio at the same time. The input to the network is exactly the Markov state of the MDP introduced in Section 3.1. The left part of the network processes sheet images, the right part spectrogram excerpts (including Δ images). After low-level representation learning, the two modalities are merged by concatenation and further processed using dense layers. This architecture implies that policy and value network share the parameters of the lower layers, which is a common choice in RL [18]. Finally, there are two output layers: the first represents our policy and predicts the action selection probability $\pi_{\Theta}(a|s)$. It contains three output neurons (one for each possible action) converted into a valid probability distribution via soft-max activation. The second output layer consists of one linear output neuron predicting the value $V(s)$ of the current state. Table 1 lists the exact architectures used for our experiments. We use exponential linear units for all but the two output layers [5].

Table 1. Network architecture. DO: Dropout, Conv(3, stride-1)-16: 3×3 convolution, 16 feature maps and stride 1.

Audio (Spectrogram) 78 × 40	Sheet-Image 80 × 256
Conv(3, stride-1)-32	Conv(5, stride-(1, 2))-32
Conv(3, stride-1)-32	Conv(3, stride-1)-32
Conv(3, stride-2)-64	Conv(3, stride-2)-64
Conv(3, stride-1)-64 + DO(0.2)	Conv(3, stride-1)-64 + DO(0.2)
Conv(3, stride-2)-64	Conv(3, stride-2)-64
Conv(3, stride-2)-96	Conv(3, stride-2)-64 + DO(0.2)
Conv(3, stride-1)-96	Conv(3, stride-2)-96
Conv(1, stride-1)-96 + DO(0.2)	Conv(1, stride-1)-96 + DO(0.2)
Dense(512)	Dense(512)
Concatenation + Dense(512)	
Dense(256) + DO(0.2)	Dense(512) + DO(0.2)
Dense(3) - Softmax	Dense(1) - Linear

4.2 Learning a Policy via Actor-Critic

One of the first algorithms proposed for optimizing a policy was REINFORCE [27], a Monte-Carlo algorithm that learns by generating entire episodes $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots$ of states, actions and rewards by following policy π_{Θ} while interacting with the environment. Given this sequence it updates the parameters Θ of the policy network according to the following update rule by replaying the episode time step by time step:

$$\Theta \leftarrow \Theta + \alpha G_t \nabla_{\Theta} \ln \pi_{\Theta}(A_t | S_t, \Theta) \quad (1)$$

α is the step size or learning rate and G_t is the true discounted cumulative future reward (the return) received from time step t onwards. Gradient ∇_{Θ} is the direction in parameter space in which to go if we want to maximize the selection probability of the respective action. This means whenever the agent did well, achieving a high return G_t , we take larger steps in parameter space towards selecting the responsible actions. By changing the parameters of the policy network, we of course also change our policy (behavior) and we will select beneficial actions more frequently in the future when confronted with similar states.

REINFORCE and policy optimization are known to have high variance in the gradient estimate [11]. This results in slow learning and poor convergence properties. To address this problem, **REINFORCE with Baseline** (REINFORCE_{bl}) adapts the update rule of Equation (1) by subtracting the estimated state value $V(s)$ (see Section 3.3) from the actual return G_t received:

$$\Theta \leftarrow \Theta + \alpha (G_t - V(s)) \nabla_{\Theta} \ln \pi_{\Theta}(A_t | S_t, \Theta) \quad (2)$$

This simple adaptation helps to reduce variance and improves convergence. The value network itself is learned by minimizing the mean squared error between the actually received return and the predicted value estimate of the network, $(G_t - V(s))^2$. REINFORCE_{bl} will be the first learning algorithm considered in our experiments.

Actor-critic methods are an extension of the baseline concept, allowing agents to learn in an online fashion while interacting with the environment. This avoids the need for creating entire episodes prior to learning. In particular, our actor-critic agent will only look into the future a fixed number of t_{max} time steps (in our case, 15). This implies that we do not have the actual return G_t available for updating

the value function. The solution is to *bootstrap* the value function (i.e., update the value estimate with estimated values), which is the core characteristic of actor-critic methods. The authors in [18] propose the **Synchronous Advantage Actor Critic (A2C)** and show that running multiple actors (in our case 16) in parallel on different instances of the same kind of environment, further helps to stabilize training. We will see in our experiments that this also holds for the score following task. For a detailed description of the learning process we refer to the original paper [18].

5. EXPERIMENTAL RESULTS

In this section we experimentally evaluate our RL approach to score following and compare it to a previously introduced method [7] that solves the same task. In addition to quantitative analysis we also provide a video of our agents interacting with the score following environment.³

5.1 Experimental Setup

Two different datasets will be used in our experiments. The *Nottingham Dataset* comprises 296 monophonic melodies of folk music (training: 187, validation: 63, testing: 46); it was already used in [7] to evaluate score following in sheet music images. The second dataset contains 479 classical pieces by various composers such as Beethoven, Mozart and Bach, collected from the freely available *Mutopia Project*⁴ (training: 360, validation: 19, testing: 100). It covers polyphonic music and is a substantially harder challenge to a score follower. In both cases the sheet music is typeset with Lilypond and the audios are synthesized from MIDI using an acoustic piano sound font. This automatic rendering process provides the precise audio – sheet music alignments required for training (see Section 2). For audio processing we set the computation rate to 20 FPS and compute log-frequency spectrograms at a sample rate of 22.05kHz. The FFT is computed with a window size of 2048 samples and post-processed with a logarithmic filterbank allowing only frequencies from 60Hz to 6kHz (78 frequency bins).

The spectrogram context visible to the agents is set to 40 frames (2 sec. of audio) and the sliding window sheet images cover 160×512 pixels and are further downsampled by a factor of two before being presented to the network. As optimizer we use the *Adam* update rule [15] with an initial learning rate of 10^{-4} and running average coefficients of 0.5 and 0.999. We then train the models until there is no improvement in the number of tracked onsets on the validation set for 50 epochs and reduce the learning rate by factor 10 three times. The tempo change action Δv_{pxl} is 0.5 for Nottingham and 1.0 for the polyphonic pieces.

5.2 Evaluation Measures and Baselines

Recall from Section 3.3 and Figure 4 that from the agent’s position \hat{x} and the ground truth position x , we compute the tracking error d_x . This error is the basis for our evaluation

³ score following video: <https://youtu.be/COPNciY510g>

⁴ <http://www.mutopiaproject.org/>

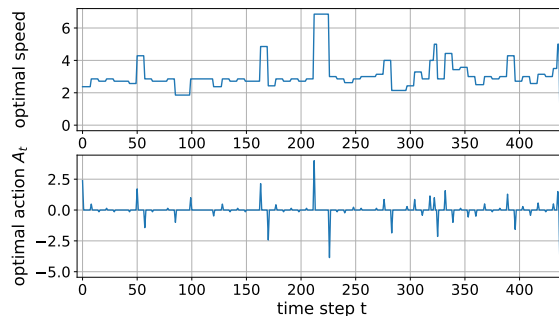


Figure 6. Optimal tempo curve and corresponding optimal actions A_t for a continuous agent (piece: J. S. Bach, BWV994). The A_t would be the target values for training an agent with supervised, feed-forward regression.

measures. However, compared to training, we only consider time steps in our evaluation where there is actually an onset present in the audio. While interpolating intermediate time steps is helpful for creating a stronger learning signal (Section 3.3), it is not musically meaningful. Specifically, we will report the evaluation statistics *mean absolute tracking error* $\overline{|d_x|}$ as well as its standard deviation $std(|d_x|)$ over all test pieces. These two measures quantify the *accuracy* of the score followers. To also measure their *robustness* we compute the ratio R_{on} of overall tracked onsets as well as the ratio of pieces R_{tue} tracked from beginning entirely to the end.

As *baseline method* we consider the approach described in [7], which models score following as a multimodal localization task (denoted by MM-Loc in the following).

As a *second baseline*, we also tried to train an agent to solve the score following MDP in a fully supervised fashion. This is theoretically possible, as we know for each time point the exact corresponding position in the score image, which permits us to derive an optimal tempo curve and, consequently, an optimal sequence of tempo changes for each of the training pieces. Figure 6 shows such an optimal tempo curve along with the respective tempo change actions for a short Bach piece. The latter would serve as targets y in a supervised regression problem $y = f(x)$. The network structure used for this experiment is identical to the one in Figure 5 except for the output layers. Instead of policy π_θ and value V we only keep a single linear output neuron predicting the value of the optimal tempo change in each time step. However, a closer look at Figure 6 already reveals the problem inherent in this approach. The optimal tempo change is close to zero most of the time. For the remaining time steps we observe sparse spikes of varying amplitude. When trying to learn to approximate these optimal tempo changes (with a mean squared error optimization target), we ended up with a network that predicts values very close to zero for all its inputs. We conclude that the relevant tempo change events are too sparse for supervised learning and exclude the method from our tables in the following. Besides these technical difficulties we will also discuss conceptual advantages of addressing score following as an MDP in Section 6.

Method	R_{true}	R_{on}	$ d_x $	$std(d_x)$
Nottingham (monophonic, 46 test pieces)				
MM-Loc [7]	0.43	0.65	3.15	13.15
REINFORCE _{bl}	0.94	0.96	4.21	4.59
A2C	0.96	0.99	2.17	3.53
Mutopia (polyphonic, 100 test pieces)				
MM-Loc [7]	0.61	0.72	62.34	298.14
REINFORCE _{bl}	0.20	0.35	48.61	41.99
A2C	0.74	0.75	19.25	23.23

Table 2. Comparison of score following approaches. Best results are marked in bold. For A2C and REINFORCE_{bl} we report the average over 10 evaluation runs.

5.3 Experimental Results

Table 2 provides a summary of the experimental results. Looking at the Nottingham dataset, we observe large gaps in performance between the different approaches. Both RL based methods manage to follow almost all of the test pieces completely to the end. In addition, the mean tracking error is lower for A2C and shows a substantially lower standard deviation. The high standard deviation for MM-Loc is even more evident in the polyphonic pieces. The reason is that MM-Loc is formulated as a localization task, predicting a location probability distribution over the score image given the current audio. Musical passages can be highly repetitive, which leads to multiple modes in the location probability distribution, each of which is equally probable. As the MM-Loc tracker follows the mode with highest probability it starts to jump between such ambiguous structures, producing a high standard deviation for the tracking error and, in the worst case, loses the target.

Our MDP formulation of score following addresses this issue, as the agent controls its progression speed for navigating through the sheet image. This restricts the agent as it does not allow for large jumps in the score and, in addition, is much closer to how music is actually performed (e.g. from left to right and top to bottom when excluding repetitions). Our results (especially the ones of A2C) reflect this theoretical advantage.

However, in the case of complex polyphonic scores we also observe that the performance of REINFORCE_{bl} degrades completely. The numbers reported are the outcome of more than five days of training. We already mentioned in Section 4 that policy optimization is known to have high variance in the gradient estimate [11], which is exactly what we observe in our experiments. Even though REINFORCE_{bl} managed to learn a useful policy for the Nottingham dataset it also took more than five days to arrive at that. In contrast, A2C learns a successful policy for the Nottingham dataset in less than six hours and outperforms the baseline method on both datasets. For Mutopia it tracks more than 70% of the 100 test pieces entirely to the end without losing the target a single time. This re-

sult comes with an average error of only 20 pixels which is about 5mm in a standard A4 page of Western sheet music – three times more accurate than the baseline with a mean error of 62 pixels.

We also report the results of REINFORCE_{bl} to emphasize the potential of RL in this setting. Recall that the underlying MDP is the same for both REINFORCE_{bl} and A2C. The only part that changes is a more powerful learner. All other components including network architecture, optimization algorithm and environment remain untouched. Considering that deep RL is currently one of the most intensively researched areas in machine learning, we can expect further improvement in the score following task whenever there is an advance in RL itself.

6. DISCUSSION AND CONCLUSION

We have proposed a formulation of score following in sheet music images as a Markov decision process and showed how to address it with state-of-the-art deep reinforcement learning. Experimental results on monophonic and polyphonic piano music show that this is competitive with recently introduced methods [7]. We would like to close with a discussion of some specific aspects that point to interesting future perspectives.

Firstly, we trained all agents using a continuous reward signal computed by interpolating the target (*ground truth*) location between successive onsets and note heads. Reinforcement learners can, of course, also learn from a *delayed* signal (e.g. non-zero rewards only at actual onsets or even bar lines or downbeats). This further implies that we could, for example, take one of our models trained on the synthesized audios, annotate a set of real performance audios at the bar level (which is perfectly feasible), and then fine-tune the models with the very same algorithms, with the sole difference that for time points without annotation the environment simply returns a neutral reward of zero.

Secondly, we have already started to experiment with continuous control agents that directly predict the required tempo changes, rather than relying on a discrete set of action. Continuous control has proven to be very successful in other domains [9] and would allow for a perfect alignment of sheet music and audio (cf. Figure 6).

A final remark concerns RL in general. For many RL benchmarks we are given a simulated environment that the agents interact with. These environments are fixed problems without a natural split into training, validation and testing situations. This is different in our setting, and one of the main challenges is to learn agents, which generalize to unseen pieces and audio conditions. While techniques such as weight-decay, dropout [24] or batch-normalization [13] have become a standard tool for regularization in supervised learning they are not researched in the context of RL. A broad benchmark of these regularizers in the context of RL would be therefore of high relevance.

We think that all of this makes the score following MDP a promising and in our opinion very exciting playground for further research in both music information retrieval and reinforcement learning.

7. ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC Grant Agreement 670035, project CON ESPRESSIONE).

8. REFERENCES

- [1] Andreas Arzt, Harald Frostel, Thassilo Gadermaier, Martin Gasser, Maarten Grachten, and Gerhard Widmer. Artificial intelligence in the concertgebouw. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2424–2430, Buenos Aires, Argentina, 2015.
- [2] Andreas Arzt, Gerhard Widmer, and Simon Dixon. Automatic page turning for musicians via real-time machine listening. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI)*, pages 241–245, Patras, Greece, 2008.
- [3] Stefan Balke, Sanu Pulimootil Achankunju, and Meinard Müller. Matching musical themes based on noisy OCR and OMR input. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 703–707, Brisbane, Australia, 2015.
- [4] Donald Byrd and Jakob Grue Simonsen. Towards a standard testbed for optical music recognition: Definitions, metrics, and page images. *Journal of New Music Research*, 44(3):169–195, 2015.
- [5] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *International Conference on Learning Representations (ICLR)* (arXiv:1511.07289), 2015.
- [6] Arshia Cont. A coupled duration-focused architecture for realtime music to score alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):837–846, 2009.
- [7] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. Towards score following in sheet music images. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 789–795, New York City, United States, 2016.
- [8] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. Learning audio-sheet music correspondences for score identification and offline alignment. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 115–122, Suzhou, China, 2017.
- [9] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.
- [10] Zhiyao Duan and Bryan Pardo. A state space model for on-line polyphonic audio-score alignment. In *Proc. of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011.
- [11] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, 2004.
- [12] Jan Hajič jr and Pavel Pecina. The MUSCIMA++ Dataset for Handwritten Optical Music Recognition. In *14th International Conference on Document Analysis and Recognition (ICDAR)*, pages 39–46, New York, United States, 2017.
- [13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456, 2015.
- [14] Özgür Izmirli and Gyanendra Sharma. Bridging printed music and audio through alignment using a mid-level score representation. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 61–66, Porto, Portugal, 2012.
- [15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)* (arXiv:1412.6980), 2015.
- [16] Frank Kurth, Meinard Müller, Christian Fremerey, Yoon-ha Chang, and Michael Clausen. Automated synchronization of scanned sheet music with audio recordings. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, pages 261–266, Vienna, Austria, 2007.
- [17] Marius Miron, Julio José Carabias-Orti, and Jordi Janer. Audio-to-score alignment at the note level for orchestral recordings. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 125–130, Taipei, Taiwan, 2014.
- [18] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.
- [19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharmarajan, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

- [20] Meinard Müller. *Fundamentals of Music Processing*. Springer Verlag, 2015.
- [21] Eita Nakamura, Philippe Cuvillier, Arshia Cont, Nobutaka Ono, and Shigeki Sagayama. Autoregressive hidden semi-markov model of symbolic music performance for score following. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 392–398, Málaga, Spain, 2015.
- [22] Matthew Prockup, David Grunberg, Alex Hrybyk, and Youngmoo E. Kim. Orchestral performance companion: Using real-time audio to score alignment. *IEEE Multimedia*, 20(2):52–60, 2013.
- [23] Christopher Raphael. Music Plus One and machine learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.
- [24] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [25] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.
- [26] Verena Thomas, Christian Fremerey, Meinard Müller, and Michael Clausen. Linking Sheet Music and Audio - Challenges and New Approaches. In Meinard Müller, Masataka Goto, and Markus Schedl, editors, *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*, pages 1–22. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2012.
- [27] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.
- [28] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger B. Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *CoRR*, abs/1708.05144, 2017.

MATRIX CO-FACTORIZATION FOR COLD-START RECOMMENDATION

Olivier Gouvert¹

Thomas Oberlin¹

Cédric Févotte¹

¹ IRT, Université de Toulouse, CNRS, France

firstname.lastname@irit.fr

ABSTRACT

Song recommendation from listening counts is now a classical problem, addressed by different kinds of collaborative filtering (CF) techniques. Among them, Poisson matrix factorization (PMF) has raised a lot of interest, since it seems well-suited to the implicit data provided by listening counts. Additionally, it has proven to achieve state-of-the-art performance while being scalable to big data. Yet, CF suffers from a critical issue, usually called cold-start problem: the system cannot recommend new songs, i.e., songs which have never been listened to. To alleviate this, one should complement the listening counts with another modality. This paper proposes a multi-modal extension of PMF applied to listening counts and tag labels extracted from the Million Song Dataset. In our model, every song is represented by the same activation pattern in each modality but with possibly different scales. As such, the method is not prone to the cold-start problem, i.e., it can learn from a single modality when the other one is not informative. Our model is symmetric (it equally uses both modalities) and we evaluate it on two tasks: new songs recommendation and tag labeling.

1. INTRODUCTION

New albums and songs are released every day and are instantly available on streaming platforms. An important issue for streaming companies is therefore to develop recommender systems which are able to handle such new songs [13, 20]. More generally, additional information on those songs is needed to enrich the catalog, allowing the user to efficiently explore and find the songs he might like. In this perspective, tag labeling has proven to be very useful. The labels can be attributed by experts or by the user, and algorithms can complement this information with automatic labeling [7].

For both tasks (song recommendation and tag labeling), matrix factorization (MF) techniques [12, 17], and in particular Poisson MF (PMF), reach significant performance. Unfortunately, these techniques suffer from the well-known cold-start problem: such a recommender sys-

tem cannot recommend songs which have never been listened to, and similarly it cannot labeled untagged songs. A joint modeling of both modalities can achieve cold-start recommendation, as soon as at least one modality is observed for every song [8, 22].

In this paper, we propose a new matrix co-factorization model based on PMF, which performs those two tasks jointly. Our model is robust to the cold-start problem for both modalities. It can recommend a song which has never been listened to, based on its associate tags. And symmetrically, it can associate tags on a song based on who listened to it. To do that, we separately model the scale (popularity) of each song according to each modality, while the patterns across the topics are shared.

The state of the art of co-factorization techniques is presented in Section 2, along with some background on PMF. Then, in Section 3 we will present our new model and explain its properties. In Section 4, we provide a majorization-minimization (MM) algorithm for solving our optimization problem and underline its scalability. Finally, in Section 5, we test our model on songs recommendation and tag labeling in various settings.

2. RELATED WORKS

In this paper, we will focus on works based on so-called hybrid techniques [1] and Poisson matrix factorization. Note that recommendation tasks can also be addressed with other techniques such as factorization machines [19].

2.1 Poisson matrix factorization

PMF is a non-negative MF (NMF) technique [14]. Let \mathbf{Y} be a matrix of size $F \times I$, where each column represent an item (song) i according to F features. MF approximates the observed matrix \mathbf{Y} by a low-rank product of two matrices: $\mathbf{Y} \approx \mathbf{W}\mathbf{H}^T$, where $\mathbf{W} \in \mathbb{R}_+^{F \times K}$ represents a dictionary matrix, and $\mathbf{H} \in \mathbb{R}_+^{I \times K}$ represents a matrix of attributes (activations), with $K \ll \min(F, I)$.

When observed data are in the form of counts, i.e., $\mathbf{Y} \in \mathbb{N}^{F \times I}$, a classical hypothesis is to assume that each observation is drawn from a Poisson distribution:

$$y_{fi} \sim \text{Poisson}([\mathbf{W}\mathbf{H}^T]_{fi}). \quad (1)$$

The maximum likelihood (ML) estimator of \mathbf{W} and \mathbf{H} is therefore obtained by minimizing the cost function de-



© Olivier Gouvert, Thomas Oberlin, Cédric Févotte. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Olivier Gouvert, Thomas Oberlin, Cédric Févotte. “Matrix co-factorization for cold-start recommendation”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

defined by:

$$\begin{aligned} C(\mathbf{W}, \mathbf{H}) &= -\log p(\mathbf{Y}|\mathbf{W}, \mathbf{H}) \\ &= D_{KL}(\mathbf{Y} | \mathbf{W}\mathbf{H}^T) + cst \quad (2) \\ \text{s.t. } \mathbf{W} &\geq 0, \mathbf{H} \geq 0, \end{aligned}$$

where cst is a constant w.r.t. \mathbf{W} and \mathbf{H} , and where D_{KL} is the generalized Kullback-Liebler (KL) divergence defined by:

$$D_{KL}(\mathbf{Y}|\mathbf{X}) = \sum_{f,i} \left(y_{fi} \log \frac{y_{fi}}{x_{fi}} - y_{fi} + x_{fi} \right). \quad (3)$$

This low-rank approximation is known as KL non-negative matrix factorization (KL-NMF) [9, 15].

The cost function C is scale invariant, i.e., for any diagonal non-singular matrix $\Lambda \in \mathbb{R}_+^{K \times K}$, we have $C(\mathbf{W}, \mathbf{H}) = C(\mathbf{W}\Lambda^{-1}, \mathbf{H}\Lambda)$. To avoid degenerate solutions, a renormalization such that $\sum_f w_{fk} = F$ is often used, where $w_{fk} = [\mathbf{W}]_{fk}$.

Several extensions based on Bayesian formulations have been proposed in the literature [3, 5, 6, 10, 17]. In [10], the authors developed a hierarchical Poisson factorization (HPF) by introducing new variables: the popularity of the items and the activity of the users. These variables play a significant role in recommendation tasks.

2.2 Co-factorization

A way of circumventing the cold-start problem is to introduce new modalities [8, 11, 16]. Co-factorization frameworks have been developed to jointly factorize two matrices of observations (two modalities): $\mathbf{Y}^A \approx \mathbf{W}^A(\mathbf{H}^A)^T$ and $\mathbf{Y}^B \approx \mathbf{W}^B(\mathbf{H}^B)^T$, with shared information between the activation matrices: $\mathbf{H}^A \approx \mathbf{H}^B$.

2.2.1 Hard co-factorization

Hard co-factorization [8, 21] posits that the link between activations is an equality constraint: $\mathbf{H}^A = \mathbf{H}^B = \mathbf{H}$. This is equivalent to concatenate the observations \mathbf{Y}^A and \mathbf{Y}^B , and the dictionaries \mathbf{W}^A and \mathbf{W}^B :

$$\begin{aligned} D_{KL}(\mathbf{Y}^A|\mathbf{W}^A\mathbf{H}^T) + \gamma D_{KL}(\mathbf{Y}^B|\mathbf{W}^B\mathbf{H}^T) \\ = D_{KL} \left(\begin{pmatrix} \mathbf{Y}^A \\ \gamma \mathbf{Y}^B \end{pmatrix} \middle| \begin{pmatrix} \mathbf{W}^A \\ \gamma \mathbf{W}^B \end{pmatrix} \mathbf{H}^T \right), \quad (4) \end{aligned}$$

where $\gamma \in \mathbb{R}^+$ is a weighting hyperparameter.

As in Section 2.1, scale invariance issues can be solved by a renormalization step such that: $\sum_u w_{uk}^A + \gamma \sum_v w_{vk}^B = U + V$.

2.2.2 Soft co-factorization

Soft co-factorization [21] relaxes the equality constraint on the activations replacing it by a soft penalty controlled by an hyperparameter $\delta \in \mathbb{R}^+$:

$$\begin{aligned} D_{KL}(\mathbf{Y}^A|\mathbf{W}^A(\mathbf{H}^A)^T) + \gamma D_{KL}(\mathbf{Y}^B|\mathbf{W}^B(\mathbf{H}^B)^T) \\ + \delta \text{Pen}(\mathbf{H}^A, \mathbf{H}^B). \quad (5) \end{aligned}$$

A popular choice for this penalty is the ℓ_1 -norm: $\text{Pen}(\mathbf{H}^A, \mathbf{H}^B) = \|\mathbf{H}^A - \mathbf{H}^B\|_1$. It is adapted when both modalities are likely to share the same activations, except at some sparse locations where they can differ significantly.

2.2.3 Offset models

Bayesian formulations of the soft co-factorization problem have also been developed through the introduction of an offset latent variable [11, 22]. The link between activations is therefore given by:

$$h_{ik}^B = h_{ik}^A + \varepsilon_{ik}, \quad (6)$$

where ε is a latent random variable.

In particular in [11], a co-factorization model is developed based on PMF, with $\varepsilon_{ik} \sim \text{Gamma}(\alpha, \beta)$. This choice is motivated by the conjugacy propriety of the gamma distribution with the Poisson distribution. Nevertheless, the model is not symmetric with respect to (w.r.t.) the activations \mathbf{H}^A and \mathbf{H}^B , as $h_{ik}^B > h_{ik}^A$ by construction. Thus, it can solve the cold-start problem only for the modality A and not for B.

3. PROPOSED MODEL

3.1 Notations

In this article, we work with two different modalities. The first modality, denoted by A, corresponds to the listening counts of U users on I songs. The second modality, denoted by B, corresponds to the tags assigned to these I songs, among a set of V tags. \mathbf{W}^A and \mathbf{W}^B thus denote the preferences of users and the atoms of tags across the K patterns, respectively.

3.2 Link between attributes

We propose an equality constraint on normalized activations. We denote by $n_i^A = \sum_k h_{ik}^A$ and $n_i^B = \sum_k h_{ik}^B$, the sum of the rows of the activations. We impose, for each item i :

$$\frac{h_{ik}^A}{n_i^A} = \frac{h_{ik}^B}{n_i^B} = d_{ik}, \quad (7)$$

when $n_i^A > 0$ and $n_i^B > 0$.

- The $I \times K$ matrix \mathbf{D} with entries d_{ik} controls the attributes patterns subject to the constraint $\sum_k d_{ik} = 1$. This information is shared by activations of both modalities. For example, the K patterns can be related to genre information: we expect that experimental rock songs share the same patterns.
- $\mathbf{N}^A = \text{diag}(n_i^A)$ controls the scale of songs across the modality A. It corresponds to the popularity of the song, in the sense that a lot of people listen to it.
- $\mathbf{N}^B = \text{diag}(n_i^B)$ controls the scale of songs across the modality B. It corresponds to the fact that a song can have more or less tag labels.

Two songs can have the same attributes patterns \mathbf{D} but different scales. For example, a song i can be a very popular song, known by a large panel of people: $n_i^A \gg 0$, but lack tag labeling: $n_i^B \approx 0$. On the contrary, another song i can be unpopular (because it is new or not well-received): $n_i^A \approx 0$, but have a lot of tag information (a set of experts may have labeled the song): $n_i^B \gg 0$.

The counterpart of Equation (2) is the following cost function C , which we aim to minimize:

$$\begin{aligned} C(\mathbf{W}^A, \mathbf{W}^B, \mathbf{D}, \mathbf{N}^A, \mathbf{N}^B) & \quad (8) \\ &= D_{KL}(\mathbf{Y}^A | \mathbf{W}^A(\mathbf{N}^A \mathbf{D})^T) \\ &+ \gamma D_{KL}(\mathbf{Y}^B | \mathbf{W}^B(\mathbf{N}^B \mathbf{D})^T) \\ \text{s.t. } \mathbf{W}^A &\geq 0, \mathbf{W}^B \geq 0, \mathbf{D} \geq 0, \\ &\text{diag}(\mathbf{N}^A) \geq 0, \text{diag}(\mathbf{N}^B) \geq 0. \end{aligned}$$

We denote by $\mathbf{Z} = \{\mathbf{W}^A, \mathbf{W}^B, \mathbf{N}^A, \mathbf{N}^B, \mathbf{D}\}$ the set of variables to infer.

3.3 Scale invariance

Let $\Theta = \text{diag}(\theta_i)$ be a diagonal matrix of size $I \times I$ with non-negative entries. We have the following scale invariance:

$$\begin{aligned} C(\mathbf{W}^A, \mathbf{W}^B, \Theta^{-1} \mathbf{D}, \mathbf{N}^A \Theta, \mathbf{N}^B \Theta) \\ = C(\mathbf{W}^A, \mathbf{W}^B, \mathbf{D}, \mathbf{N}^A, \mathbf{N}^B). \end{aligned} \quad (9)$$

This scale invariance allows us to impose the constraint on \mathbf{D} , described in Section 3, by applying a renormalization step (see Section 4.2).

Let $\Lambda = \text{diag}(\lambda_k)$ be a diagonal matrix of size $K \times K$ with non-negative entries, $\bar{\mathbf{W}}^A = \mathbf{W}^A \Lambda^{-1}$, $\bar{\mathbf{W}}^B = \mathbf{W}^B \Lambda^{-1}$ and $\bar{\mathbf{D}} = \mathbf{D} \Lambda$. We also have the following scale invariance:

$$\begin{aligned} C(\bar{\mathbf{W}}^A, \bar{\mathbf{W}}^B, \bar{\mathbf{D}}, \mathbf{N}^A, \mathbf{N}^B) \\ = C(\mathbf{W}^A, \mathbf{W}^B, \mathbf{D}, \mathbf{N}^A, \mathbf{N}^B). \end{aligned} \quad (10)$$

In practice, this invariance is not an issue and we do not apply a renormalization step. However, this kind of invariance plays a role for the scores used in recommendation as discussed in Section 3.4.

3.4 Recommendation tasks

In recommender systems, a classical problem is to propose a ranked list of songs, users or tags. We develop how to construct this list on two tasks: in- and out-prediction.

3.4.1 In-matrix recommendation

In-matrix recommendation is a task of recommendation on users and items which do not suffer from the cold-start problem. For in-matrix recommendation, we propose a ranked list of songs for each user, based on the score defined by:

$$s_{ui}^A = \sum_k w_{uk}^A h_{ik}^A. \quad (11)$$

This score and our cost function C have the same scale invariance described in Eq. 10.

3.4.2 Cold-start (out-matrix) recommendation

Cold-start (or out-matrix) recommendation is a task of recommendation on items which suffer from the cold-start problem (on modality A or B). In this section, we take the example of a cold-start problem on modality A, i.e., the song has no information in the modality A (nobody has listened to this song yet) but has tags associated to it. The following remark would hold for a cold-start problem on modality B.

For cold-start (out-matrix) recommendation the score is defined by:

$$s_{ui}^A = \sum_k w_{uk}^A d_{ik} = \sum_k w_{uk}^A \frac{h_{ik}^A}{\sum_l h_{il}^A}. \quad (12)$$

Contrary to in-prediction, we use \mathbf{D} and not $\mathbf{H}^A = \mathbf{N}^A \mathbf{D}$ since the popularity in the modality A is close to zero for songs with no information, i.e., $n_i^A \approx 0$.

This score and the cost function C do not have the same scale invariance described in Eq. 10. In fact, if we denote $\bar{w}_{uk}^A = \lambda_k w_{uk}^A$ and $\bar{h}_{ik}^A = \lambda_k h_{ik}^A$, we have:

$$\bar{s}_{ui}^A = \sum_k \bar{w}_{uk}^A \frac{\bar{h}_{ik}^A}{\sum_l \bar{h}_{il}^A} = s_{ui}^A \frac{\sum_k h_{ik}^A}{\sum_k \lambda_k h_{ik}^A} = s_{ui}^A c_i, \quad (13)$$

where $c_i = \frac{\sum_k h_{ik}^A}{\sum_k \lambda_k h_{ik}^A}$.

This means that, if we want to rank the different scores s_{ui}^A , we have to do it for a fixed item. Therefore, to properly evaluate the cold-start problem for songs, we will propose a ranked list of users (or tags), for a given item.

For a streaming company, it corresponds to obtaining a ranked list of users which are likely to listen to this new song, or a ranked list of tags which corresponds to the song.

4. OPTIMIZATION

4.1 Auxiliary function

The objective function C has no closed-form minimum and is not convex. We use a MM algorithm [9] to reach a local minimum. The MM algorithms start by designing a majorizing surrogate G of the objective function $C(\mathbf{Z}) \leq G(\mathbf{Z} | \tilde{\mathbf{Z}})$ which is tight at the current value $\tilde{\mathbf{Z}}$, i.e., $C(\tilde{\mathbf{Z}}) = G(\tilde{\mathbf{Z}} | \tilde{\mathbf{Z}})$.

We use Jensen inequality on terms of the form $\log(\sum_i x_i)$. We define:

$$\phi_{uik}^A = \frac{\tilde{w}_{uk}^A \tilde{d}_{ik}}{\sum_k \tilde{w}_{uk}^A \tilde{d}_{ik}}, \quad c_{uik}^A = y_{ui}^A \phi_{uik}^A, \quad (14)$$

$$\phi_{uik}^B = \frac{\tilde{w}_{uk}^B \tilde{d}_{ik}}{\sum_k \tilde{w}_{uk}^B \tilde{d}_{ik}}, \quad c_{uik}^B = y_{ui}^B \phi_{uik}^B. \quad (15)$$

It leads to the following upper-bound:

$$\begin{aligned} G(\mathbf{Z} | \tilde{\mathbf{D}}, \tilde{\mathbf{W}}^A, \tilde{\mathbf{W}}^B) \\ = \sum_{uik} [-c_{uik}^A \log(w_{uk}^A n_i^A d_{ik}) + w_{uk}^A n_i^A d_{ik}] \\ + \gamma \sum_{vik} [-c_{vik}^B \log(w_{vk}^B n_i^B d_{ik}) + w_{vk}^B n_i^B d_{ik}] + cst. \end{aligned} \quad (16)$$

4.2 Updates

The auxiliary function G can be optimized by using a block descent algorithm. At each iteration, we optimize one latent variable, keeping all the others fixed. This technique leads to four update rules described in the following.

- Variables \mathbf{W}^A and \mathbf{W}^B :

$$w_{uk}^A \leftarrow \frac{\sum_i c_{uik}^A}{\sum_i n_i^A d_{ik}}; \quad w_{vk}^B \leftarrow \frac{\sum_i c_{vik}^B}{\sum_i n_i^B d_{ik}} \quad (17)$$

- Variables \mathbf{N}^A and \mathbf{N}^B :

$$n_i^A \leftarrow \frac{\sum_u y_{ui}^A}{\sum_{uk} w_{uk}^A d_{ik}}; \quad n_i^B \leftarrow \frac{\sum_v y_{vi}^B}{\sum_{vk} w_{vk}^B d_{ik}} \quad (18)$$

- Variable \mathbf{D} :

$$d_{ik} \leftarrow \frac{\sum_u c_{uik}^A + \gamma \sum_v c_{vik}^B}{n_i^A \sum_u w_{uk}^A + \gamma n_i^B \sum_v w_{vk}^B} \quad (19)$$

As discussed in Section 3.3, we add a renormalization step at the end of each iteration. The update is as follows:

$$\theta_i = \sum_k d_{ik} / I, \quad (20)$$

$$\mathbf{D} \leftarrow \Theta^{-1} \mathbf{D}; \quad \mathbf{N}^A \leftarrow \mathbf{N}^A \Theta; \quad \mathbf{N}^B \leftarrow \mathbf{N}^B \Theta. \quad (21)$$

4.3 Algorithm

The complete algorithm is summarized in Algorithm 1. Note that the inference only requires browsing the non-zero data $y_{ui}^A > 0$ and $y_{vi}^B > 0$, during the update of the local variables c_{uik}^A and c_{vik}^B . Hence, our algorithm has the same scalability as PMF, making it particularly well-suited for processing huge sparse matrices, as it is the case in recommender systems (see Table 1).

The algorithm is stopped when the relative increment of the cost function C is lower than a chosen parameter τ .

5. EXPERIMENTS

5.1 Experimental Setup

5.1.1 Datasets

We use two datasets extracted from the Million Song Dataset (MSD) [2] and merge them on songs:

- The Taste Profile dataset provides listening counts of 1M users on 380k songs [18]. We select a subset of the users and pre-process the data to remove users and items with few information [16]. We keep only users who listened to at least 20 songs, and songs which have been listened to by at least 20 users.

Algorithm 1: MM Algorithm

Input : $\mathbf{Y}^A, \mathbf{Y}^B, K, \gamma$

Initialize: $\mathbf{W}^A, \mathbf{W}^B, \mathbf{N}^A, \mathbf{N}^B, \mathbf{D}$

repeat

 for each pair (u, i) such that $y_{ui}^A > 0$: Eq. 14

 for each pair (v, i) such that $y_{vi}^B > 0$: Eq. 15

 for each user u and tag v : Eq. 17

 for each item i : Eq. 18-19

 normalization step: Eq. 21

until C converges;

	Taste Profile	Last.fm
# columns (songs)	15,667	15,667
# rows (users or tags)	16,203	620
# non-zeros	792,761	128,652
% non-zeros	0.31%	1.32%

Table 1. Datasets structure after pre-processing.

- The Last.fm dataset provides tag labels for around 500k songs. These tags were extracted from the Last.fm API [4]. Since the tags were collected via user annotation, they are quite noisy. To avoid miss-labeling in the train data, we pre-process it. We keep only the 1000 most used tags in the whole dataset. For each couple song-tag, a confidence rating is given by Last.fm, we keep only couples with confidence higher than 10. Finally, we keep only tags which appears at least in 20 songs. The top 10 of the tags in the dataset after the pre-processing are shown in Table 2.

We binarize the two datasets. Structure of both datasets is described in Table 1.

5.1.2 Evaluation metric: ranking prediction

In each experiment, we will propose a ranked list \mathcal{L} of N items (which can be songs, tags or users) and evaluate its quality w.r.t. a ground-truth relevance. For this, we calculate the discounted cumulative gain (DCG) and its normalized version, the NDCG:

Tags	Occ.	Tags	Occ.
rock	6703	electronic	2413
alternative	4949	female vocalists	2407
indie	4151	indie rock	2171
pop	3853	Love	1875
alternative rock	2854	singer-songwriter	1786

Table 2. Occurrences (Occ.) of the top tags in the dataset after pre-processing.

Experiment	OUT-A		OUT-B		IN-A	
Score	NDCG@20	NDCG@200	NDCG@1*	NDCG@10	NDCG@100	NDCG**
P-coNMF	0.0824 $\pm 1.48e^{-5}$	0.122 $\pm 1.33e^{-5}$	0.416 $\pm 5.85e^{-4}$	0.266 $\pm 1.59e^{-4}$	0.129 $\pm 4.24e^{-6}$	0.286 $\pm 2.82e^{-6}$
H-coNMF	0.0873 $\pm 1.39e^{-5}$	0.131 $\pm 2.21e^{-5}$	0.391 $\pm 1.73e^{-4}$	0.264 $\pm 1.00e^{-4}$	0.122 $\pm 5.72e^{-6}$	0.283 $\pm 2.96e^{-6}$
KL-NMF	0.163 $\pm 5.36e^{-7}$	0.313 $\pm 1.50e^{-7}$

Table 3. Performance of three models: P-coNMF, H-coNMF, KL-NMF, on three different tasks: out-matrix song recommendation (OUT-A), tag labeling (OUT-B), in-matrix recommendation (IN-A). Each algorithm is run 5 times, the mean and the variance of the NDCG metrics are displayed. * NDCG@1 corresponds to the percentage of success on the first predicted tag. ** NDCG is not truncated in this column, it is equivalent to chose $N = I$.

$$\text{DCG@N} = \sum_{n=1}^N \frac{\text{rel}(n)}{\log_2(n+1)}, \quad (22)$$

$$\text{NDCG@N} = \frac{\text{DCG@N}}{\text{IDCG@N}}, \quad (23)$$

where $\text{rel}(n)$ is the ground-truth relevance of the n -th item in the list \mathcal{L} . In the following, $\text{rel}(n) = 1$ if the item is relevant and $\text{rel}(n) = 0$ if not.

The denominator of the DCG penalizes relevant items which are at the end of the ranked list. It accounts for the fact that a user will only browse the beginning of the list, and will not pay attention to items which are ranked at the end. IDCG is the ideal DCG. It corresponds to the DCG score of an oracle which ranks perfectly the list, thus scaling the NDCG between 0 and 1.

5.1.3 Compared methods

For each experiment, we will compare the performance of our model, proportional co-factorization NMF (P-coNMF) with two other methods:

- KL-NMF, presented in Section 2.1. It can only be used for in-matrix prediction as it suffers from the cold-start problem.
- Hard co-factorization (H-coNMF), presented in Section 2.2.1, that use KL-NMF algorithm on concatenated matrix. For out-matrix prediction, we will use a mask that indicates what columns are missing. The objective function is then:

$$C(\mathbf{W}, \mathbf{H}) = D_{KL}(\mathbf{X} \otimes \mathbf{Y} \mid \mathbf{X} \otimes \mathbf{WH}^T), \quad (24)$$

where \otimes is the elementwise multiplication, and \mathbf{X} is the mask. Note that the masked H-coNMF is expected to perform as good as soft coNMF with the ℓ_1 -norm, since it does not enforce common activation for new songs.

For both methods, we chose $K = 100$ latent factors. The hyperparameter is set such that $\gamma = \frac{U}{V}$, which allows to compensate for the size difference between the two datasets ($V \ll U$).

5.2 Cold-start recommendation

In this section, we evaluate our algorithm on cold-start recommendation tasks for both modalities A and B. For this, we artificially replace columns of \mathbf{Y}^A and \mathbf{Y}^B by columns full of zeros, in order to create the train datasets $\mathbf{Y}_{\text{train}}^A$ and $\mathbf{Y}_{\text{train}}^B$. It leads to 10% of songs with only listening counts information, 10% of songs with only tag information and 80% of songs with both informations. The removed columns form the test datasets $\mathbf{Y}_{\text{test}}^A$ and $\mathbf{Y}_{\text{test}}^B$.

For each song among the never-listened-to songs, we want to find a set of users that is likely to listen to it. We train all the algorithms on $\mathbf{Y}_{\text{train}}^A$ and $\mathbf{Y}_{\text{train}}^B$. For each song, we create a ranked list of users based on the score defined in Section 3.4.2. We evaluate its relevance based on the NDCG metrics with ground-relevance defined by: $\text{rel}(u, i) = \mathbb{1}(y_{\text{test}, ui}^A > 0)$, where $\mathbb{1}(x)$ is the indicator function which is equal to 1 when x is true and 0 otherwise.

Similarly, for each song among the untagged songs, we want to find a set of tags that can annotate that song. Then we propose a ranked list of tags and calculate the NDCG score with ground-relevance defined by: $\text{rel}(v, i) = \mathbb{1}(y_{\text{test}, vi}^B > 0)$.

The columns OUT-A and OUT-B of Table 3 present the results of P-coNMF and H-coNMF on the two cold-start problems. For recommending potential listeners (OUT-A), H-coNMF seems to be slightly better than our method. However, P-coNMF outperforms H-coNMF on tag labeling task. P-coNMF presents a success rate of 42% on the first predicted tag. This is an acceptable rate since the tag dataset is noisy: it has not been labeled by experts but by users and presents some incoherences. For example, the tag 'Hip-Hop' can also be written 'hip hop'. More details on tag labeling are provided in Section 5.4. Contrary to H-coNMF, P-coNMF does not need a mask to know which columns are missing. Additionally, the scale variables \mathbf{N}^A and \mathbf{N}^B are able to explain different scalings of the same song in the two datasets. This seems interesting because the amount of listening counts and tags for the same song is often highly different.

	FACTOR #94	FACTOR #29	FACTOR #30
Top tags	Hip-Hop hip hop classic rap Gangsta Rap	new wave post-punk Guilty Pleasures intense Post punk	experimental Experimental Rock Avant-Garde noise weird
Top songs based on H^A	Eminem - "Mockingbird" Eminem - "Without Me" Kid Cudi - "Day 'N' Nite" Kid Cudi - "Up Up & Away" Kid Cudi - "Cudi Zone"	The Cure - "Boys Don't Cry" The Smiths - "There Is A Light [...]" The Smiths - "This Charming Man" The Smiths - "What Difference Does It Make?" Wolfsheim - "Once In A Lifetime"	Animal Collective - "Fireworks" Sigur Ros - "Staralfur" Sonic Youth - "Youth Against Fascism" Grizzly Bear - "Little Brother" TV On The Radio - "Crying"
Top songs based on D	DMX - "Where The Hood At" Lil Jon - "Crunk Juice" 50 Cent - "Straight To The Bank" Eminem - "The Kiss" The Notorious B.I.G. - "Respect"	New Order - "The Perfect Kiss" Talking Heads - "Burning Down The House" Joy Division - "Disorder" Tears For Fears - "Goodnight Song" The Smiths - "Miserable Lie"	The Mars Volta - "Tira Me a Las Aranas" Cocorosie - "Gallows" The Mars Volta - "Concertina" The Mars Volta - "Roulette Dares" TV On The Radio - "Golden Age"

Table 4. Three examples of factors, with, for both of them, the 5 top tags associated to it, the 5 top songs associated to it, with or without the notion of popularity.

5.3 In-matrix song recommendation

We also evaluate our algorithm on in-matrix prediction. The goal is therefore to predict which songs a user is likely to listen. There is no cold-start recommendation here, and KL-NMF can be trained.

We artificially split the listening counts dataset in two. 20% of non-zero values of \mathbf{Y}^A are removed to create the test set $\mathbf{Y}_{\text{test}}^A$. The 80% remaining form the train set $\mathbf{Y}_{\text{train}}^A$ on which the different models are trained. Each method is evaluated with NDCG metric. For each user, a list of songs is proposed based on the score defined in Section 3.4.1, among the songs he never listened to. The ground-truth relevance is defined by $\text{rel}(u, i) = \mathbb{1}(y_{\text{test}, ui}^A > 0)$.

The results are presented in the third column (IN-A) of Table 3. P-coNMF is slightly better than H-coNMF, but we observe that KL-NMF achieves state-of-the-art performance. This is not surprising, since adding information on another modality (tags here) can be viewed as a regularizing term. We lose in precision in in-matrix recommendation task but we solve the cold-start problem. This seems an interesting trade-off.

5.4 Exploratory Analysis

In Table 4, we present for each of the three factors $k \in \{29, 30, 94\}$:

- in the first row, the tags which corresponds to the five highest values of \mathbf{W}^B .
- in the second row, the songs which corresponds to the five highest values of $\mathbf{H}^A = \mathbf{N}^A \mathbf{D}$.
- in the third row, the songs which corresponds to the five highest values of \mathbf{D} .

The top tags associated to each factor are consistent: for example, genre as 'new wave' and 'post-punk' are in the same factor. The model is also robust to the different spellings used by the users ('post-punk' and 'Post punk' for example). Then, we see that the top songs in each factor are related with the top tags. Eminem, 50 Cent and The

Notorious B.I.G. are rap artists. The Cure, The Smiths and Joy Division are the leading figures of the new wave. TV On The Radio, The Mars Volta and Animal Collective are known to be experimental rock bands. Finally, we see that the popularity of songs \mathbf{N}^A has an important influence on the diversity of the top songs in each factor. When this notion is removed (last row of the table), less popular songs and bands appear in the top songs.

6. CONCLUSION

In this paper, we proposed a new Poisson matrix co-factorization, in which the attributes of each modality are assumed proportional. Contrary to hard and ℓ_1 -based soft co-factorization, in this new model each item may have different scaling (or popularity) in each modality. This is of particular interest when tackling cold-start recommendation, in which one scaling is close to zero. The benefits of the algorithm over standard co-factorization have been illustrated for song recommendation, with emphasis placed on cold-start situations.

This raised interesting short-term perspectives, such as the derivation of more involved Bayesian models, and inference or extensions to different, possibly non-binary datasets. Future works should also consider datasets with highly different dimensions or dynamics, by means of a tri-factorization.

7. ACKNOWLEDGMENTS

This work has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program under grant agreement No 681839 (project FACTORY).

8. REFERENCES

- [1] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 191–226. Springer, 2015.

- [2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, 2011.
- [3] John Canny. GaP: A factor model for discrete data. In *Proc. International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 122–129, 2004.
- [4] Òscar Celma. Music recommendation. In *Music recommendation and discovery*, pages 43–85. Springer, 2010.
- [5] Ali Taylan Cemgil. Bayesian inference for nonnegative matrix factorisation models. *Computational Intelligence and Neuroscience*, 2009.
- [6] O. Dikmen and C. Fevotte. Maximum marginal likelihood estimation for nonnegative dictionary learning in the Gamma-Poisson model. *IEEE Transactions on Signal Processing*, 60(10):5163–5175, 2012.
- [7] Douglas Eck, Paul Lamere, Thierry Bertin-Mahieux, and Stephen Green. Automatic generation of social tags for music recommendation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 385–392, 2008.
- [8] Yi Fang and Luo Si. Matrix co-factorization for recommendation with rich side information and implicit feedback. In *Proc. International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec)*, pages 65–69, 2011.
- [9] Cédric Févotte and Jérôme Idier. Algorithms for non-negative matrix factorization with the β -divergence. *Neural computation*, 23(9):2421–2456, 2011.
- [10] Prem Gopalan, Jake M. Hofman, and David M. Blei. Scalable recommendation with hierarchical Poisson factorization. In *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 326–335, 2015.
- [11] Prem K Gopalan, Laurent Charlin, and David Blei. Content-based recommendations with Poisson factorization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3176–3184. 2014.
- [12] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
- [13] Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. Addressing cold-start problem in recommendation systems. In *Proc. International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 208–211, 2008.
- [14] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.
- [15] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 556–562. 2001.
- [16] Dawen Liang, Minshu Zhan, and Daniel PW Ellis. Content-aware collaborative music recommendation using pre-trained neural networks. In *Proc. International Society for Music Information Retrieval (ISMIR)*, pages 295–301, 2015.
- [17] Hao Ma, Chao Liu, Irwin King, and Michael R. Lyu. Probabilistic factor models for web site recommendation. In *Proc. International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 265–274, 2011.
- [18] Brian McFee, Thierry Bertin-Mahieux, Daniel PW Ellis, and Gert RG Lanckriet. The million song dataset challenge. In *Proc. International Conference on World Wide Web (WWW)*, pages 909–916, 2012.
- [19] S. Rendle. Factorization machines. In *Proc. International Conference on Data Mining (ICDM)*, pages 995–1000, 2010.
- [20] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proc. International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 253–260, 2002.
- [21] N. Seichepine, S. Essid, C. Févotte, and O. Cappé. Soft nonnegative matrix co-factorization. *IEEE Transactions on Signal Processing*, 62(22):5940–5949, 2014.
- [22] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proc. International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 448–456, 2011.

Author Index

- Abdallah, Samer 725
 Abeßer, Jakob 306, 416, 577
 Aljanaki, Anna 615
 Andrade, Nazareno 732
 Andreux, Mathieu 327
 Aravind, Rangarajan 499
 Arzt, Andreas 592

 Badkobeh, Golnaz 233
 Balke, Stefan 306, 416
 Barone, Michael 529
 Basaran, Dogac 82
 Batliner, Anton 376, 461
 Bauer, Christine 678
 Bello, Juan Pablo 106, 453
 Bigo, Louis 355
 Bigoni, Francesco 128
 Bittner, Rachel 453, 514
 Bitton, Adrien 175
 Bozkurt, Baris 483
 Brunner, Gino 747
 Bugbee, Erin H. 341

 Calvo-Zaragoza, Jorge 34, 240, 248, 256
 Canno, Isabelle 424
 Cano, Estefanía 577
 Carsault, Tristan 18
 Castellanos, Francisco 256
 Cecconi, Cécile 424
 Chemla-Romeu-Santos, Axel 175
 Chen, Tsung-Ping 90
 Cheng, Tian 763
 Cherla, Srikanth 725
 Chi, Heng-Yu 168
 Choi, Keunwoo 506
 Cohen-Hadria, Alice 431
 Condit-Schultz, Nathaniel 66
 Costantini, Giovanni 376
 Crawford, Tim 233
 Crayencour, Hélène C. 106
 Cumming, Julie 348, 491

 Dahl, Sofia 128
 de Carvalho, Verônica Oliveira 400
 de Padua, Renan 400
 de Valk, Reinier 281
 Delbouys, Rémi 370
 Demetriou, Andrew 514
 Demirel, Emir 483
 Dixon, Simon 334, 755, 777
 Donahue, Chris 475
 Dong, Hao-Wen 190
 Dorfer, Matthias 225, 784
 Duan, Zhiyao 204, 218
 Durand, Simon 438

 Eck, Douglas 50

 Ehmman, Andreas F. 637
 Elezi, Ismail 271
 Elsen, Erich 50
 Engel, Jesse 50
 Eremenko, Vsevolod 483
 Esling, Philippe 18, 175
 Essid, Slim 82, 106
 Ewert, Sebastian 334

 Falcão, Felipe 732
 Feisthauer, Laurent 355
 Févotte, Cédric 792
 Finkensiep, Christoph 547
 Frieler, Klaus 777
 Fuentes, Magdalena 106
 Fujinaga, Ichiro 66, 256, 348, 491
 Fukayama, Satoru 561, 687, 763

 Gebhardt, Roman B. 3
 Giraud, Mathieu 355
 Gómez, Juan S. 577
 Goto, Masataka 561, 687, 763
 Gouvert, Olivier 792
 Grachten, Maarten 26, 661
 Gupta, Chitralekha 529, 600
 Gururani, Siddharth 569

 Ha, Jung-Woo 717
 Hadjakos, Aristotelis 630
 Hajić jr., Jan 225
 Hantke, Simone 376
 Harasim, Daniel 152
 Harrison, Peter M. C. 160
 Hawthorne, Curtis 50
 Henkel, Florian 784
 Hennequin, Romain 370, 622, 645
 Hockman, Jason 58, 313
 Höger, Frank 777
 Hu, Xiao 362
 Humphrey, Eric 438
 Hung, Yun-Ning 135

 Ibrahim, Karim 529
 Itoyama, Katsutoshi 145

 Jansson, Andreas 514
 Jensenius, Alexander Refsum 74
 Jeong, Dasaem 120
 Ju, Yaolong 66

 Kelkar, Tejaswinee 74
 Kim, Taehoon 289
 Kinnaird, Katherine M. 341, 585
 Kompatsiaris, Yiannis 702
 Konrad, Andres 747
 Korzeniowski, Filip 10, 211, 264
 Kosta, Katerina 725
 Krasanakis, Emmanouil 702

- Kulis, Brian 182
Kumar, Aparna 514
Kwak, Nojun 289
Kwon, Taegyun 120
- Lafay, Grégoire 771
Lagrange, Mathieu 771
Lattner, Stefan 26, 592, 661
Lau, Josephine 671
Lee, Jin Ha 671
Lee, Jongpil 717
Lee, Kyogu 289
Lee, Kyungyun 506
Lehner, Bernhard 321
Lerch, Alexander 445, 569, 608
Leresche, Françoise 424
Levé, Florence 355
Lewis, David 233
Li, Bochen 218
Li, Fanjie 362
Li, Haizhou 600
Liebman, Elad 695
Lisena, Pasquale 424
Lu, Wei Tsung 521, 740
Luo, Yin-Jyun 653
Lustig, Ethan 204
Lykartsis, Athanasios 3
- Maezawa, Akira 218
Malheiro, Ricardo 383
Mallat, Stéphane 327
Manilow, Ethan 297
Manzelli, Rachel 182
Mao, Huanru Henry 475
McAuley, Julian 475
McFee, Brian 106, 438
McGuirl, Melissa R. 341
McKay, Cory 348, 491
McLeod, Andrew 42, 113
McVicar, Matt 725
Medeot, Gabriele 725
Meseguer-Brocal, Gabriel 431
Meur, Thierry Le 424
Mishra, Saumitra 755
Mitkas, Pericles 702
Moussallam, Manuel 370, 622, 645
Müller, Meinard 98, 306, 409, 416
Murthy, Hema 499
- Nakamura, Eita 145
Nam, Juhan 120, 506, 717
Neuwirth, Markus 547
Newton-Rex, Ed 725
Ng, Jeremy T. D. 362
Nieto, Oriol 637
Nika, Jerome 18
- Oberlin, Thomas 792
O'Donnell, Timothy J. 152
Ofner, André 392
Oore, Sageev 50
- Pacha, Alexander 240
Paiva, Rui Pedro 383
Panda, Renato 383
Papadopoulos, Symeon 702
Parada-Cabaleiro, Emilia 376, 461
Pardo, Bryan 297, 468
Park, Jangyeon 717
Park, Jiyoung 717
Park, Sungheon 289
Pauwels, Johan 453
Pearce, Marcus T. 160
Pecina, Pavel 225
Peeters, Geoffroy 82, 431
Pertusa, Antonio 34
Pfleiderer, Martin 777
Piccoli, Francesco 370
Pichl, Martin 709
Pons, Jordi 637
Pritchard, Liz 671
Prockup, Matthew 637
Puyrenier, Frédéric 424
- Raffel, Colin 50
Ren, Iris Yuping 539
Rezende, Solange 400
Rizo, David 248
Roberts, Adam 50
Rohrmeier, Martin 152, 547
Román, Miguel A. 34
Rossignol, Mathias 771
Roy, Udit 74
Royo-Letelier, Jimena 370, 622, 645
Rudolph, Günter 554
- Savard, Claire 341
Schedl, Markus 678
Scherer, Klaus 376
Schinas, Emmanouil 702
Schlüter, Jan 321
Schmidhuber, Jürgen 271
Schmidt, Erik M. 637
Schmitt, Maximilian 376, 461
Schreiber, Hendrik 98, 409
Schuller, Bjoern 376, 461
Sears, David 211
Seetharaman, Prem 297, 468
Selvi, Marco 725
Serra, Xavier 483, 637
Shih, Shun-Yao 168
Siahkamari, Ali 182
Silva, Diego Furtado 400, 732
Simon, Ian 50

- Soleymani, Mohammad 615
 Song, Jialin 50
 Southall, Carl 58, 313
 Spinelli, Louis 671
 Stables, Ryan 58, 313
 Stadelmann, Thilo 271
 Steedman, Mark 42, 113
 Stein, Michael 3
 Stober, Sebastian 392
 Stoller, Daniel 334
 Stone, Peter 695
 Stuchbery, Jonathan 491
 Sturm, Bob L. 755
 Su, Li 90, 521, 653, 740
 Subramanian, Vinod 608
 Summers, Cameron 569
 Swierstra, Wouter 539
- Takahashi, Takumi 561
 Thakkar, Vijay 182
 Todorov, Konstantin 424
 Tong, Rong 600
 Tralie, Christopher 197
 Tran, Viet-Anh 622
 Troncy, Raphaël 424
 Tsukuda, Kosetsu 687
 Tsushima, Hiroaki 145
 Tuggener, Lukas 271
- VanderStel, Joseph 204
 Vatolkin, Igor 554
 Veltkamp, Remco 539
 Vigliensoni, Gabriel 256
 Viraraghavan, Venkata 499
 Voisin, Martine 424
 Volk, Anja 539
- Wahl, Alison 468
 Waloschek, Simon 630
 Wang, Ye 529, 600
 Wang, Yuyi 747
 Wattenhofer, Roger 747
 Webster, Kevin 725
 Weiss, Christof 416
 Weyde, Tillman 281
 White, Corey N. 695
 Widmer, Gerhard 10, 26, 211, 225, 264, 661, 784
 Wilkins, Julia 468
 Wu, Chih-Wei 445
- Xi, Qingyang 453
- Yan, Yujia 204
 Yang, Yi-Hsuan 135, 190
 Ye, Xuzhou 453
 Yoshii, Kazuyoshi 145
- Zangerle, Eva 709