GANs and Poses: An Interactive Generative Music Installation Controlled by Dance Moves

Richard Vogl[*†]
Faculty of Informatics
TU Wien
Vienna, Austria
richard.vogl@tuwien.ac.at

Hamid Eghbal-zadeh[*]
Institute of Computational Perception
JKU Linz, Austria
hamid.eghbal-zadeh@jku.at

Gerhard Widmer
Institute of Computational Perception
JKU Linz, Austria
gerhard.widmer@jku.at

Peter Knees
Faculty of Informatics
TU Wien
Vienna, Austria
peter.knees@tuwien.ac.at

Abstract

GANs and Poses is an interactive multi-media installation using human poses to control artificial neural networks generating parts for a music arrangement. To provide incentive for interaction, a game-style UI rewards the audience with audio-visual feedback, which in turn drives the progress through the musical arrangement. This work uses a variety of artificial neural networks to *(i)* detect persons and their body posture, *(ii)* classify dance moves, and *(iii)* generate drum as well as melody patterns.

Introduction

In the recent past, generative artificial neural networks (NN) have gained popularity and received much attention in science as well as popular media. Particularly, Generative Adversarial Neural Networks (GANs) [2] and Recurrent Neural Networks (RNNs) are successfully applied for generating high resolution images [7, 8], audio [1], and music [9] with surprising quality. While first attempts to generate music with neural networks resulted in compositions lacking structural consistency, recent demonstrations are more convincing and show the potential of these methods [5].

In this work, we combine state-of-the-art NN-based human pose estimation with an experimental generative music creation method utilizing GANs with a recurrent-convolutional architecture. Using these technologies, a semi-human/semi-

* Equal contributions.
† Richard Vogl is also affiliated with the Institute of Computational Perception at JKU Linz, Austria.

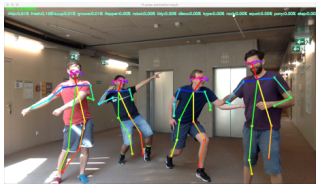


Figure 1: Webcam image overlaid with the output of the pose estimation system along with the estimated probabilities for different dance moves.

machine music-ensemble is created when the audience interactively controls song structure and generated musical elements using body gestures. To encourage user engagement, a simple game-style reward system is incorporated in the system. By using recognized dance moves as achievements and the individual parts of the music arrangement's structure as game levels, the UI presents incentives for users to interact with the system. Users can score points by performing various dance moves which results in visual and musical effects and progresses the music through the different levels/parts of the arrangement.

On an artistic level, a musical composition is driven by the engagement and expression of individual actors, each contributing their skills to a joint progression of a musical experience. Capturing multiple performers and their moves adds a social dimension to the installation, stimulating passive spectators and turning them into active performers, motivated to add to the generation of sound.

System

An overview of the system can be found in Figure 2. As mentioned, the input for the interactive installation is provided by a real-time multi-person pose recognition system. This system is implemented using a 12-layer convolutional neural network (CNN) designed to be efficient during inference (MobileNets [4]). The implementation follows the open-pose approach to real-time pose detection [10] and was trained on the MSCOCO Keypoints dataset [6][1]. Figure 1 shows the input and detected body parts of the pose estimator.

Using the detected stick-figure-like coordinates extracted from the input video, a dance move recognition system

[1] http://cocodataset.org

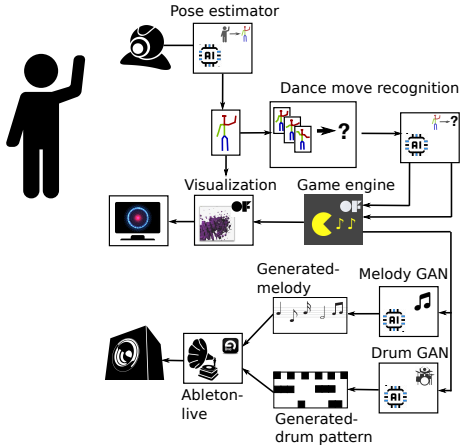
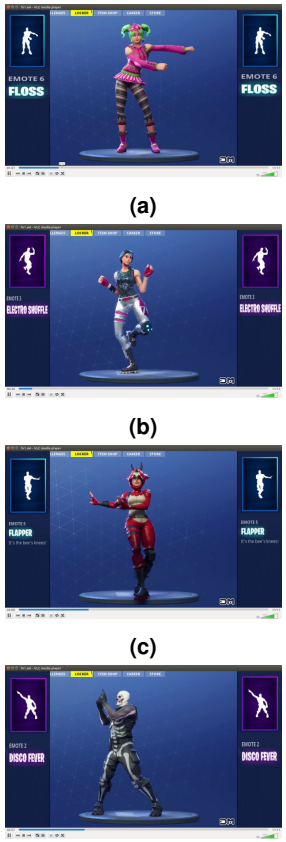


Figure 2: Overview of the system showing its individual parts and components.

is used to detect if any of the present persons are dancing. To this end, videos of character dances from the video game *Fortnite*[2] were used as training data. The videos of the game character dances represent clean samples of 26 different dances, and were chosen for their easy accessibility and volume of good samples of dance moves. Figure 3 shows samples of the screenshots for different dances of the collection. The input data for training was extracted using the pose estimation system, using additional data augmentation to increase robustness. A two-layer Long Short-Term Memory (LSTM) [3] neural network was trained using the 1.5 second sequences of the predicted pose coordinates as inputs and the 26 dance moves as output classes.

For both melody and rhythm generation, a convolutional-recurrent GAN architecture was used. The GANs were

[2] https://www.youtube.com/watch?v=2XtLEp0XAII



(a) **(b)** **(c)** **(d)**

Figure 3: Screenshots of selected dance moves from the video game *Fortnite*. Automatically extracted poses from these videos were used to train a simple RNN dance move classifier.

trained in a semi-supervised setup to allow parametrization of generated patterns, which is used to control complexity of the generated music to match the different parts of the music arrangement. As training data, a MIDI dataset extracted from a freely available online collection[3] was used.

A simple game-engine and user interface (UI) implemented in openFrameworks[4] represents the centerpiece of the system. As can be seen in Figure 4, the UI consists of a visualization of the detected poses, the current part/level of the musical arrangement represented by stars, as well as bars which indicate remaining time until a certain amount of dances has to be performed to achieve further progress. In Figure 4 we provide different screenshots of the UI while two persons are interacting with the installation. Communication with the pose detection software as well as with the pattern generation server is implemented using Open Sound Control[5].

While the drum part and lead melody is generated using GANs, the installation uses a predefined arrangement providing basic chord progressions for each of the levels/parts of the musical piece. This arrangement uses a live-set composed in Ableton Live[6], which is controlled via MIDI by the openFrameworks application.

Interaction

The installation shows a screen with stars representing the different stages of the music arrangement (e.g.: intro, verse, bridge, chorus – see Figure 4). When a person steps in front of the camera, a stick-figure model of the recognized body will be shown on screen. The played music and a particle animation which reacts to the pose and music is supposed to animate the person to move. As soon as dance-like moves are detected, visual feedback provides more incentive to dance. When enough dance moves are detected, the arrangement moves to the next part, which results in a change in chord progression, and adds additional elements such as generated drums as well as melody. When the arrangement is in a state other than the base-state, the audience has only limited amount of time to perform dance moves, until the arrangement falls back into the previous stage (the timer is indicated with a red bar). Since the change of segments is synchronized to the chord progression, a green timer is shown after sufficient dance moves are detected and indicates when the music will progress to the next part. A video that demonstrates this installation can be found at https://youtu.be/mg7_KzwlhC8.

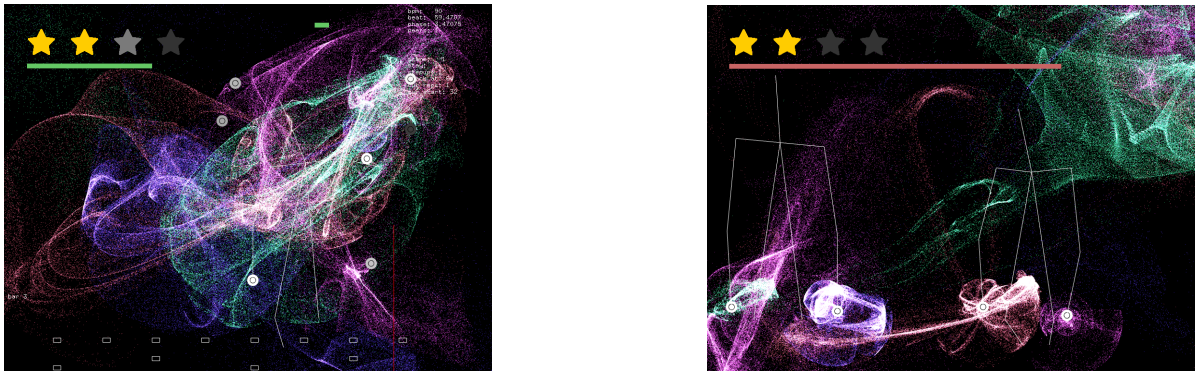
Requirements and Setup

The minimum requirements for setting up the installation are:

- An electricity outlet to power the computer the system runs on.
- A reasonably well lit space which allows one or more persons to stand in front of the webcam in a distance of 2-3m.
- A platform to put the computer on, ideally chest height, to allow good visibility of the screen while providing a natural perspective of the webcam.

The setup is straight forward since all the components of the installation can be run on a MacBook Pro with required GPU, webcam, and screen built in. We can bring and provide the laptop with pre-installed software for the duration of the conference. Additionally required headphones and a

[3] http://www.midiworld.com
[4] https://openframeworks.cc
[5] http://opensoundcontrol.org/
[6] https://www.ableton.com/en/live



(a) **(b)**

Figure 4: Screenshots of the installation in action. Image *a* shows the UI with additional developmentdebug information such as the song state (upper right corner) and generated drum and melody patterns (lower half of screen). Image *b* shows a screen of the clean, camera ready UI without debug information.

headphone amplifier (to allow multiple people hearing the audio) can also be provided by us.

For best presentation and UX, the following additional components are desirable, however, not essential for a successful setup: *(i)* a big screen and external webcam (HDMI and USB respectively) *(ii)* external speakers, and *(iii)* space for more than 2 people.

Acknowledgements

This work has been partly funded by the Austrian FFG under the BRIDGE 1 project *SmarterJam* (858514), as well as by the Austrian Ministries BMVIT and BMWFW, and the Province of Upper Austria, via the COMET Center SCCH.

REFERENCES

1. Chris Donahue, Julian McAuley, and Miller Puckette. 2018. Synthesizing Audio with Generative Adversarial Networks. *arXiv preprint arXiv:1802.04208* (2018). https://arxiv.org/abs/1802.04208
2. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
3. Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
4. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861* (2017). https://arxiv.org/abs/1704.04861
5. Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Andrew Dai, Matt Hoffman, Curtis Hawthorne, and Douglas Eck. 2018. Generating Structured Music Through Self-Attention. In *The 2018 Joint Workshop on Machine Learning for Music*.
6. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.
7. Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. 2018. Which Training Methods for GANs do actually Converge?. In *International Conference on Machine Learning*. 3478–3487.
8. Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759* (2016). https://arxiv.org/abs/1601.06759
9. Adam Roberts, Jesse Engel, Sageev Oore, and Douglas Eck. 2018. Learning Latent Representations of Music to Generate Interactive Musical Palettes. (2018).
10. Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. 2016. Convolutional pose machines. In *proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA.