Performing Structured Improvisations with Pre-existing Generative Musical Models

Pablo Samuel Castro
Google Brain
psc@google.com

Abstract

The quality of outputs produced by generative models for music have seen a dramatic improvement in the last few years. However, most models perform in "offline" mode, with few restrictions on the processing time. Integrating these types of models into a live structured performance poses a challenge because of the necessity to respect the beat and harmony. In this paper we propose a framework which enables the integration of out-of-the-box models by leveraging the musician's creativity and expertise.

1 Introduction

As the popularity and quality of machine learning models grows, artists have been increasingly drawn to incorporate them into their craft. In music, recent works in generative models produce music that is both realistic and semantically consistent (Roberts et al., 2018), closely resembles human performance (Huang et al., 2018), and can aid in automatic composition[1]. The increased realism of these models is typically accompanied with an increase in the amount of time it takes to produce outputs, which often results in their inadequacy for live performance. This is particularly problematic in structured improvisation, such as in traditional jazz, where the music produced must respect the beat and harmony of the piece.

There have been previous efforts to incorporate machine learning into live musical performance by creating new digital instruments, such as the Wekinator (Fiebrink, 2009) and a method trained to emulate the performer's tonal and melodic trends (Thom, 2001). More recently, a number of creative web applications combine some of the models mentioned above with interactive interfaces that allow users with no musical training to create music[2]. These digital instruments, however, tend to operate in isolation; specifically, they prescribe the beat and harmony, rather than conforming to pre-existing beats and harmonies. In jazz performance[3], especially when performing with other musicians, one must respect the beat and harmony when improvising. In this work we propose a framework enabling the integration of out-of-the-box generative models for this type of performance.

2 Proposed Framework

Our approach takes its inspiration from call-and-response improvisations that are common in traditional jazz. In these sections two or more musicians take turns improvising over the same piece, and each musician usually incorporates melodies and/or rhythms played by previous musicians into their improvisations. This collective musical dialogue is often placed at the end of the improvisatory section of a piece, as it tends to increase the intensity and energy of the music. Our setup assumes a

[1] https://www.ampermusic.com/

[2] One good example is https://incredible-spinners.glitch.me/

[3] We are excluding free jazz here, which allows for more flexibility in performance.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

piano keyboard connected to a computer via MIDI, along with an optional controller for enabling more MIDI control messages. We use SuperCollider[4] to detect all MIDI events and pipe them through to a Python backend running on the same machine via OSC messages. The Python backend processes the notes and may then send an OSC message containing notes to be played to SuperCollider, which either generates the sound or forwards them to an external MIDI controller for producing the sound. The code is available at https://github.com/psc-g/Psc2/tree/master/research/nips2018/src.

The Python backend processes MIDI control messages to change the time signature, number of bars per phrase, whether to turn the click track on or off, as well as the *phase* of the system (see below). At the heart of the backend is a *playback array*, which keeps a list of notes to send to SuperCollider for playback. Each item in the list contains the pitch, the type of instrument to be used for playback, as well as the timestep on which to play it (the *onset*). The playback array is kept sorted by event onset. A processor thread is created at the start which loops over the playback array, timing the messages sent to SuperCollider by considering the event onsets and the current qpm (quarter-notes per minute). The playback array is a global object that can be modified by other threads running in the backend. The structured improvisation proceeds in the following phases:

1. **tap:** key presses are used to mark the desired tempo. A *click track* is created by adding equally spaced events to the playback array.
2. **bass:** Using the click track as reference, the user can input a bassline that spans the number of bars selected. The performed bassline will then be added to the playback array.
3. **drums:** Once a bassline has been added, the system deterministically adds a kick drum on the first beat, closed hi-hat every 8th note, and a snare hit for every bassline onset. In a separate process thread, this drumbeat is sent to Magenta's DrumsRNN model[5], which is tasked with generating a new drumbeat of the same length, using the provided beat as a primer. Once the drum beat is generated, it is added to the playback array.
4. **chords:** The musician can play chords, spanning the number of bars selected, on top of the bassline and drums to add harmonic context for the improvisation. The played chords will be added to the playback array to be played along with the drumbeat and bassline.
5. **call:** Now that the playback loop contains a bassline, drumbeat, chords, and optional click track, the musician can improvise. All the pitches played during the improvisation are stored in a separate buffer; when this buffer is full, a separate process thread is created where the improvised melody is sent to Magenta's PerformanceRNN model[6], tasked with continuing the melody. Once the new melody is generated by the model, it is put in the *improvising array*; note that only pitch information is stored: rhythmic information generated by the model is discarded.
6. **response:** When the *improvising array* has been filled by PerformanceRNN, the system will "intercept" the notes played by the user and replace them with the notes in the *improvising array*. This results in a hybrid improvisation: the user controls the rhythm of the notes, but the actual pitches are controlled by the model.

3 Discussion and Future Work

We view our proposal as an initial step in integrating new machine learning technologies into standard improvisational performances. It is worth stressing that the success of this approach relies heavily on the musical and creative expertise of the user. While most of the existing musical offerings using machine learning are aimed at musical experts and novices alike, our approach is specifically geared towards the expert. Our intent is to push expert improvisers into uncomfortable territory, which will hopefully result in new and exciting art which would have been difficult for the artist to approach by themselves. We would like to incorporate different aspects of the machine learning process into performance; for instance, using the gradients/weights/losses of a neural network to control the filters of an analog synthesizer[7]. We will continue to release the different techniques to our github in the hope that it proves useful to other musicians.

[4] https://supercollider.github.io/

[5] https://github.com/tensorflow/magenta/tree/master/magenta/models/drums_rnn

[6] https://github.com/tensorflow/magenta/tree/master/magenta/models/performance_rnn

[7] We explored a similar idea in https://sound-of-learning.glitch.me/

References

Fiebrink, R. (2009). Wekinator. `http://www.wekinator.org/`.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. (2018). Music transformer.

Roberts, A., Engel, J., Raffel, C., Hawthorne, C., and Eck, D. (2018). A hierarchical latent vector model for learning long-term structure in music. In *ICML*.

Thom, B. (2001). Machine learning techniques for real-time improvisational solo trading. In *ICMC*.