CS2951k Final

Devin Howard and Ebube Chuba

April 2019

1 Abstract

This project aims to create an accurate conduit between the production of real-world sounds and digital audio, essentially we're making a simple digital music producer. The inaccessibility in current music production is largely a result of the steep learning curve in Digital Audio Workstations (DAWs). The ability to produce a song out of the music that every individual can make lends itself to solving this. Currently, related work as far as creating sounds goes as far as music auto-encoders that have been used mainly to recreate specific genres of sound, specifically classical. Our work also creates a final product which is a multi-track sound built from its individual components rather than the single mp3 produced by other works. The application of this being that it allows a producer to further edit a produced sound.

Our approach is to frame song-learning as a reinforcement learning problem wherein we first use convolution over mel-spectrograms to get an embedding that accounts for temporal features. The reward function is a distance function between the input embedding and the exported final state embedding.

JacqueesBot is able to take in an input sample that will then be interpolated. The robot is able to take in a sound with no regard to its genre and length (to a reasonable length). JacqueesBot is able to then recreate the sample within a DAW and export it in a commonly used audio format.

2 Introduction

Music is one of the most ubiquitous and personal types of media that we consume. The music industry makes an estimated $43 billion dollars yearly. Although musical creativity is common, access and education to music production software is not as widespread. Even for those who are professionals, painstakingly producing the sound in your head can be cumbersome. Our project aims to address these issues by creating a straightforward agent that can learn to recreate that sound in a DAW, and serve that output to the user to continue their own edits.

JacqueesBot is this tool for automatic sample interpolation, a valuable tool to professionals and amateurs alike. With our current implementation, given enough time, JacqueesBot is able to find the perfect cover for any input sound.

Previous approaches to music generation usually are premised by deep learning systems that output similar sounds in the same genre as the input. One approach even uses reinforcement learning to fine tune the output of a music generation RNN (https://ai.google/research/pubs/pub45871). We take a different approach in that we want our AI to act like a musical producer—in the environments producers are in while making music. We also want the output (or final state) to be a in usable DAW format organized with the tracks created by the system.

We solve this problem by first training a convolutional neural network that does instrument classification based on the mel-spectrogram of a sound. After the network is trained we take the penultimate layer, before the sigmoid but after all the convolution, to use an embedding for each sound. This is how we represent the audio clips in the abstract soundspace.

Next we create a Monte Carlo tree search (MCTS) representation of the task. In this representation the state is the combination of all of the objects in the environment and conditions that are true for that environment. We define only one class of objects, Samples. Samples have the following attributes: instrument, pitch, insert time, and length. Samples also have individual embeddings, which are made by running them through the aforementioned neural network.

There are three actions in this framework: insert, hold, and finish. Insert a sample at a timestep, hold for a specific amount of time on a track, and finish a track to export and evaluate it.

The final reward function is a distance metric between the embedding of the input song and that of the final output audio. However, because embeddings were trained on short clips of audio (about 3s each), the reward will be the combination of a sliding window of both audio outputs. It will also linearly scale for the difference in length between the final exported song and the input audio. Once the reward converges, the song has been created.

3 Related Work

- Deep convolutional neural networks for predominant instrument recognition in polyphonic music by Yoonchang Han, Jaehun Kim, and Kyogu Lee: https://arxiv.org/pdf/1605.09507.pdf
- Quantitative Analysis of a Common Audio Similarity Measure by Jesper Højvang Jensen, Mads Græsbøll Christensen, Daniel P. W. Ellis, and Søren Holdt Jensen:
  https://labrosa.ee.columbia.edu/ dpwe/pubs/JensCEJ09-quantmfcc.pdf
- An Object-Oriented Representation for Efficient Reinforcement Learning by Carlos Diuk, Andre Cohen, and Michael L. Littman:
  http://carlosdiuk.github.io/papers/OORL.pdf

The related work as previously mentioned is mostly in the realm of using music auto-encoders to recreate specific genres of sound. Work has also been done in using auto-encoders to interpolate 2-bar loops, done separately for melodies and drum beats. These related projects don't encapsulate the full range of what we want JacqueesBot to be capable of. Specifically we want to remove those constraints on our song creation, allowing for any sample to be interpolated and the recreation of entire songs. Additionally, one of the most important features that we want is the ability to export created songs in a common format (mp3 or wav).

4 Technical Approach

We train a neural network on the IRMAS dataset of instruments as per the paper in (Han et. al.). This network is trained on a corpus of three-second long audio samples of different instruments and is tasked to classify that instrument in one of 11 different instrument classes. The audio files are pre-processed by converting the stereo signal to mono. This signal is then down-sampled to 22.05kHz. We then normalize the time signal by dividing by the max value. Then then mel-spectrogram of that audio is calculated with STFT windows of size 2048 and 128 mel bins as in the aforementioned paper. This spectrogram is finally normalized with a natural log before finally being passed in as input to the neural net. We use the output of the penultimate later for timbre embeddings for all sounds the agent will come across. The timbre embedding for the three second clipm c starting a time t is referred to as $TE(c_t)$.

Our MCTS agent is then structured to have a pre-selected set of instances from the class sample $s_1, s_2, \ldots, s_n$. Each sample has attributes `s.instrument`, `s.pitch`, `s.path`, `s.length`, and `s.track`.

For an input/inspiration sound I, and a final exported audio output O is the sum of the timbre reward and the pitch reward. We reduce pitch similarity to matching as best as possible the timbre and pitch of the original sound. The timbre reward is the distance in the embeddings between the timbre embedding of the input and corresponding clip in the outpout audio., The pitch reward is done by with the same process, but directly from the chromograph, $Chr(c_t)$, which gives us information for the magnitute of each pitch in a song. Because this is done in a sliding scale for each three seconds it can be also be seen as the sum:

$$\sum_t d(TE(I_t), TE(O_t)) + d(Chr(I_t), Chr(O_t))$$

However we also want to incentivize the agent to finish interpolating the whole song, so there is a length coefficient added that is lowest when the input and output audios match in length.

$$\left(\frac{-||I| - |O||}{|I|} + 1\right) \sum_t d(Embed(I_t), Embed(O_t))$$

5 Evaluation

Our goal in building with the technical approach we used was to first find a general heuristic for the type of sound recognition we were trying to do. The general idea being that two values that are relatively near each other should have recognizable features that are similar. The MCTS framework lends itself to doing this as we create a single class of objects, samples, whose many attributes we keep track of.

The reinforcement learning agent was built in the DAW taking advantage of certain built in tools including the abilities to: programmatically add and remove media files, export files on the fly, and log any changes made over the file's lifetime. Taking advantage of the DAW's api enables us to more easily alter and keep the status of the tracks we work with.

We know that we've primarily achieved our goal because we currently have the framework for our transition: a runnable RL agent to transition from state to state as it adds and removes samples from tracks in a DAW project. Additionally, we've built the convolutional network for finding the embedding of a slice of audio.

6 Conclusion

In this project, we've utilized several machine learning methods to train an AI to perform sample interpolation. A learned mel-spectrogram based embedding and a chromogram are used to calculate reward. This agent acts on the state a working DAW project, and adding and removing samples in 48x real time. Using straightforward state and action items that mimic that state and actions by musical producers, Monte Carlo Tree Search explores the action space for a state that "sounds like" the inspiration audio.

There are a few ways to improve our existing version. Investing in an optimizing our planner would do us well since there are other ways to search the problem that can include additional information we may already have for samples and instruments. However even sticking with MCTS, there are graph structures that are better tuned to less specific branching. Intentionally slimming the action space in some capacity could have a major impact on the runtime of the system.

Currently, our interaction with the DAW is rather limited, as it is only used to insert samples and export final outputs. A more advanced version would have actions like clipping samples and adding audio effects. For this proof of concept, we decided not to add this, but human producers do more than just insert.

Nonetheless, JacqueesBot has lots of potential as a tool for easy and accessible music production.