Analysis of Generating Isolated Tracks from NES Music

Bryan Learn
blearn@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, PA 15213

Mike Tasota
tasota@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

A brief overview of our project.

1 Introduction

Although similar to text generation in many respects, there is still significant progress to be made in the area of music generation. There are a few distinctions between the problem of music generation and text generation that affect this. The number of potential states that a standard 88-key piano can take greatly exceeds the number of words in any language. Multiple voices and tempo also increases the complexity of music generation.

But music is much more than just playing a combination of frequencies over an interval of time. There are several types of voices and classes of instruments that music can take on, from percussion and bass to chords and melodies. And although several attempts already exist to learn instrument classes (Song from PI [1], MuseGAN [5]), these implementations work on fixed sets of instruments and introduce chord conditioning to keep the generated harmony fixed. In this paper we explore using deep learning models to learn isolated tracks from a dataset. We also explore the potential for removing instrument and chord restrictions to see how much can be learned by the models alone.

2 Related Work

In the MidiNet paper, Yang et al. created a CNN-GAN based model for MIDI generation [5]. They propose that their model could be possibly expanded to generate multiple tracks. This was demonstrated in the work done by MuseGAN. However, MuseGAN works with a fixed set of instruments, namely bass, drums, guitar, piano, and strings. Another related work in multiple track generation is from Chu et al. in their "Song from PI" RNN. Although it is capable of producing multiple tracks, it also conditioned the generation on scale types to help pick up regularities in their dataset [1].

2.1 Magenta

In our work, we explored learning isolated tracks by using the deep learning models in the Magenta project. Magenta is a research project from the Google Brain team that explores the role of machine learning in creating art and music. Utilizing the TensorFlow library, Magenta publishes its deep learning models for other researchers to use. For our work, we focused on two of the learning models. MelodyRNN implements melody generation using Long Short-Term Memory (LSTM) cells inside of a Recurrent Neural Network. MusicVAE is a hierarchical recurrent variational autoencoder which learns a latent space of musical sequences.

3   Dataset

The suggested input data for the music generation project topic was classical piano music pieces from the website www.piano-midi.de. We ultimately decided against this dataset because of its lack of well-defined tracks and sparse variety of instrument voices across all MIDI files. On top of this, the classical piano dataset was relatively small (less than 1,000 MIDI files). For our work, it was necessary to have access to a dataset of MIDI files which had consistent track types across the entire dataset. To accomplish this, we used the NES Music Database (NES-MDB) as presented by Donahue et al. [2]. The dataset consists of 5,278 songs from the soundtracks of 397 NES games. Each MIDI file consists of four instrument voices, each mapping to an NES synthesizer. These voices are two pulse-wave generators (P1, P2), a triangle-wave generator (TR), and a percussive noise generator (NO). Since these voices are consistent throughout all files in the dataset, we assumed that each voice represented a specific track type. For example, the percussive noise generator would be a drum track for all MIDI files in which it is defined.

4   Methods

To train on specific voices, we modified the original MIDI dataset to generate four single-track datasets for each of the four tracks present (P1, P2, TR, NO). This was achieved using the midicsv and csvmidi tools developed by John Walker. By converting MIDI files to CSV format, we were able to automate the process of extracting isolated tracks from each file in the dataset. Then we passed each of the modified single-track datasets through the MelodyRNN and MusicVAE pipeline. After training was initialized, the Magenta models generated event files which include the training loss from the model at specific checkpoints.

It is also worth noting that the dataset with the isolated drums track (NO) had to be modified further to properly train on the MelodyRNN model. As per the General MIDI standard, channel 10 is dedicated to percussion, and the first step taken by MelodyRNN was to remove all tracks on this channel. To resolve this, we simply modified our script for the isolated drums track to also modify the track's MIDI channel.

5   Experiments

Since our interests lie in learning individual voice classes, we used models trained on the original dataset as well as models trained on the four individual voice classes (P1, P2, TR, and NO). This allowed us to audibly compare differences in what the baseline model learned to what our voice-specific models learned. Also, for RNN, we compared our isolated track models to the baseline model from their calculated training losses.

5.1   Training the Baseline

We wanted a baseline comparison for the two Magenta models that we considered (MusicVAE and MelodyRNN). To accomplish this, we trained with the original NES-MDB Dataset with all four voices. For the MusicVAE baseline we used the Trio configuration, where "trio" refers to an ensemble of three performers or voices. MusicVAE's trio consists of melody, bass, and drums, and it uses chord conditioning to keep harmony fixed. We also trained the MelodyRNN baseline model, but the MelodyRNN configuration only produces one melody in contrast to the trio.

5.2   Isolated Track Training

We trained both the MusicVAE model and the MelodyRNN model on each of the four isolated tracks to create a total of 8 track-separated models. Each model was trained up to 8 hours. The MelodyRNN models that were trained for 8 hours were overfitting. To address this, we compared the performance of earlier versions of the models to the overfit models by evaluating test loss.

When training drums for MelodyRNN, we learned that the basic MelodyRNN configuration ignored all information from the MIDI channel associated with percussion. For learning drums, a separate model exists in the Magenta project (DrumsRNN). However, our training indicated that training percussion on DrumsRNN only produces one note value despite having multiple percussive note

values in the training data. We successfully preserved this information in the training by modifying the MIDI files in our NO dataset to use a non-percussive MIDI channel and training with MelodyRNN.
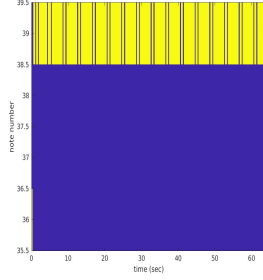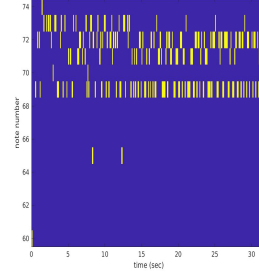

Figure 1: DrumsRNN: percussive track

Figure 2: MelodyRNN: percussive track

5.3   Generated Tracks

After generating the isolated tracks, we were interested in converting the generated MIDI files into synthesized NES tracks. As part of the NES-MDB project, there is a tool for converting MIDI files from the dataset into audio files generated from synthesized track voices. However, we learned that the generated format from the models was not enough to generate synthesized voices. This is because the original dataset included MIDI control event information to produce more expressive voices, and this information was not generated as output from the models.

6   Results

We evaluated the trained models by listening to the generated music and comparing the training loss across model types. For MelodyRNN, the training loss trend for each isolated track was less than the training loss for the baseline model. The graph for training loss in the MusicVAE model only includes models which trained beyond the initial checkpoint (P1, P2, TR). When considering the number of isolated tracks that were learned, the RNN models outperform the VAE models. The MusicVAE baseline and percussive track (NO) never went beyond the initial checkpoint in training. This resulted in the models generating mostly random noise (see Figure 3). For MusicVAE, it is interesting that training never went beyond the initial checkpoint for all training data that included percussion. Even when modifying the files to use a non-percussive MIDI channel type, the MusicVAE model failed to train beyond its initial checkpoint.
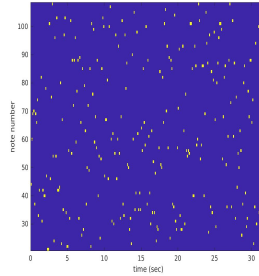

Figure 3: MusicVAE: percussive track

An additional evaluation on the generated music indicated that the RNN models produced better results than the VAE models. These results were surprising as our preliminary results seemed to indicate that the MusicVAE model was capable of generating more interesting melodies. This prior analysis was based on pre-trained models provided by the Magenta project. For the MusicVAE model, the training set consisted of over 1.5 million MIDI files scraped over the web [3]. Compared to the size of the NES-MDB dataset, it is likely that the shortcomings of our results for the VAE models were from a lack of training data.
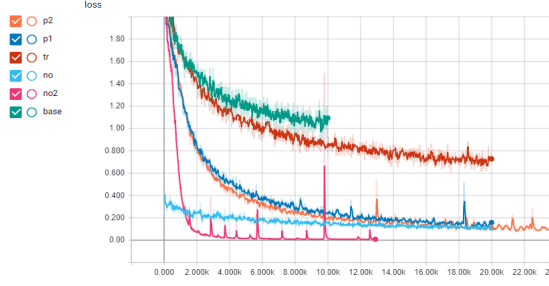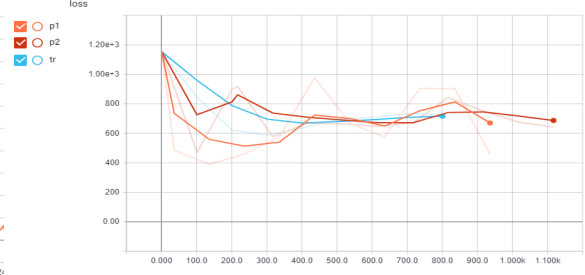
Figure 4: MelodyRNN: training loss

Figure 5: MusicVAE: training loss

7 Conclusion and Future Work

8 Acknowledgements

References

[1] Chu, Hang, et al. "Song From PI: A Musically Plausible Network for Pop Music Generation." *[1611.03477] Song From PI: A Musically Plausible Network for Pop Music Generation,* 10 Nov. 2016, arxiv.org/abs/1611.03477.

[2] Donahue, Chris et al. "The NES Music Database: A multi-instrumental dataset with expressive performance attributes" *[1806.04278] The NES Music Database: A multi-instrumental dataset with expressive performance attributes,* 12 June 2018, arxiv.org/abs/1806.04278.

[3] Engel, Jesse, et al. "Latent Constraints: Learning to Generate Conditionally from Unconditional Generative Models." *[1711.05772] Latent Constraints: Learning to Generate Conditionally from Unconditional Generative Models.,* 21 Dec 2017, arxiv.org/abs/1711.05772

[4] Simon, Ian, et al. "Learning a Latent Space of Multitrack Measures." *[1806.00195] Learning a Latent Space of Multitrack Measures,* 1 June 2018, arxiv.org/abs/1806.00195.

[5] Yang, Li-Chia, et al. "MidiNet: A Convolutional Generative Adversarial Network for Symbolic-Domain Music Generation." *[1703.10847] MidiNet: A Convolutional Generative Adversarial Network for Symbolic-Domain Music Generation,* 18 July 2017, arxiv.org/abs/1703.10847.