Exploring Music Generation Using Deep Learning

Stanford CS230 Final Project

Peyton Chen
Department of Computer Science
Stanford University
peytonc@stanford.edu

Constance Horng
Department of Computer Science
Stanford University
constanh@stanford.edu

Jialin Zhuo
Department of Computer Science
Stanford University
zhuoj@stanford.edu

Abstract

In this paper, we explore the music generation task using the Lakh Pianoroll Dataset (LPD). We built two models: a baseline RNN model using independent instrument sampling and a CNN model using dependent instrument sampling. We found that the CNN model produced better results reflecting more musical cohesion and less randomness, with both models producing generally modern and impressionist-sounding music.

1 Introduction

For our project, we are exploring music generation using deep learning. Given our extensive background as musicians, we wanted to tackle the intersection between technology and music. Since music is a universal language that so many people around the world engage in, there is considerable attention around the potential of deep learning models to generate creative and pleasant pieces.

In this project, we experimented with both RNNs and CNNs to identify the best model architecture for music generation. We used both independent and dependent instrument sampling and compared their effects on the generated audio samples. Additionally, we utilized Python 3.9.10 and TensorFlow packages in a Conda environment to implement our models.

2 Related Work

Looking at early music generation techniques, considerable work centered around Recurrent Neural Networks (RNNs). For example, the Kotecha paper proposed a two-layer LSTM model that learned harmonic and melodic rhythmic probabilities from MIDI files of Bach's music [1], while the Ingale paper used a three-layer LSTM model for generating monophonic music sequences in ABC notation [2].

However, Convolutional Neural Networks (CNNs) have been quite promising recently. For example, in 2016, the WaveNet model proved successful in generating speech mimicking the human voice, synthesizing music, and generating piano music by applying dilated convolutions.

Generative adversarial networks (GANs) have also frequently been applied to music generation. MidiNet [3] uses Deep Convolutional Generative Adversarial Networks (DCGANs), which generate

CS230: Deep Learning, Spring 2022, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

multi-instrument music phrases that can be conditioned on the previous phrase and the current phrase's chord. Additionally, MuseGAN [4] uses multiple generators to create multi-instrument music phrases that can be conditioned with respect to the other instruments in the same phrase.

3 Approach

3.1 Model Architectures

First, we built an RNN for our baseline model of generating multi-instrumental music. At each time-step, we flattened the multi-instrument pianoroll data into a one-dimensional vector. Our trained RNN then leveraged the previously learned vectors to predict the vector at the next time-step.

One challenge we ran into was that it is difficult to produce aesthetic music while preserving the multi-instrumental nature of the track. Sampling too many notes from the mulitnomial distribution proved to make the music sound extremely cluttered and even with just sampling 1 note per time step per instrument, the model struggled to produce music that was complementary leading to a lot dissonance and seemingly random sounds.

Therefore, as inspired by the WaveNet paper [5], our next experiment was to explore CNNs, which allow us to make the instrument samples dependent on each other. Compared to RNNs, CNNs are more ideal for generating sequences and faster for training convolutional operations. WaveNets combine causal filters with dilated convolutions to effectively model the long-range temporal dependencies in audio signals. However, in order to achieve our goal of generating multi-instrumental music, we will extend the approach outlined in the paper by building different CNNs for different instruments (the LPD-5-Cleansed dataset contains music samples with five instruments). Since MuseGAN uses multiple generators, we reasoned that using multiple CNNs would be more optimal for the scope of our project. Ultimately, our CNN model was able to achieve better results due to its reliance on dependent instrument sampling, leading to more cohesive connections among the different parts and an overall higher aesthetic quality.

1. **Baseline RNN** At each time-step, we flattened the multi-instrument pianoroll data into a one-dimensional vector. Our trained RNN then leveraged the previously learned vectors to predict the vector at the next time-step.
2. **2D CNN** For our CNN model, we take in a 5x20x128 tensor, representing the 5 instruments, 20 time steps, and 128 possible notes. We train the model on the data by having it predict the next 20-note sequence. The model is symmetric as we want the input and output shapes to be the same, and we chose to omit dense layers as they made the model performance worse.

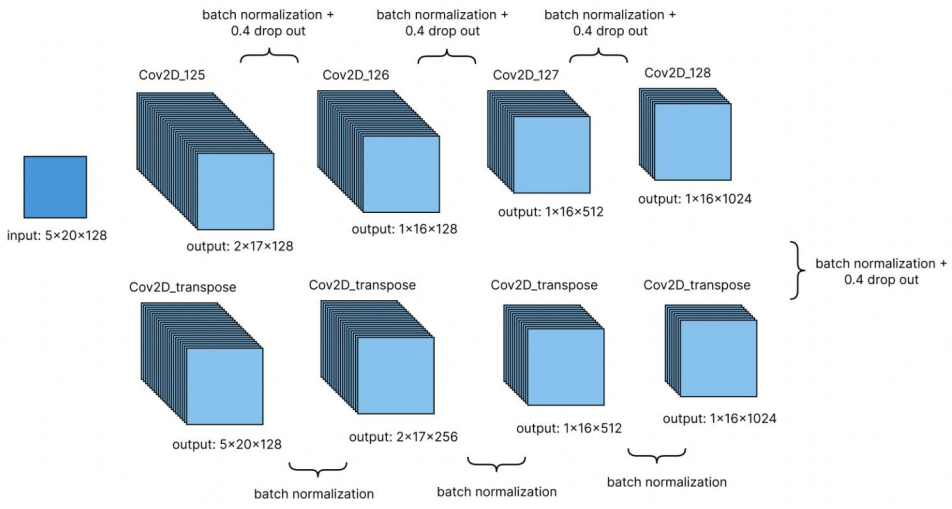

Figure 1: Our 2D CNN model architecture

4 Experiments

4.1 Data

We used the Lakh Pianoroll Dateset (LPD) derived from Lakh MIDI dateset.[4] [6] The raw data consists of 174,154 multitrack pianorolls. Each pianoroll is a music storing format that uses a score-like matrix to represents a music piece. The values represent the notes' velocities, while the vertical and horizontal axes represent note pitch and time. We used symbolic timing so that each beat has the same length, and a temporal resolution of 24 per beat to encapsulate common temporal patterns. For note pitch, there are 128 possibilities in the range of C-1 to G9. To include tracks for piano, drums, guitar, bass, and strings for generating complex music, we used the LPD-5 version of the dataset. Finally, we used a cleansed subset of 21,245 MIDI files to standardize the time signature to 4/4 and keep only the file with the highest confidence score for each song. All data was normalized before being passed in as input to our models.

4.2 Evaluation method

Music quality and aesthetic are quite subjective, so we employed a blind rating evaluation of the generated music samples performed by 10 human participants. We played 3 samples from the LPD5 dataset to get a baseline reference, then our generated RNN and CNN music camples. We asked for ratings on a scale from 1 to 10. Our final evaluation metric for each sample was the average of the ratings received across all the participants.

4.3 Experimental details

We trained our baseline RNN 100 epochs using AWS p3.2xlarge instance with a Deep Learning AMI GPU PyTorch 1.11.0 (Amazon Linux 2) 20220328.

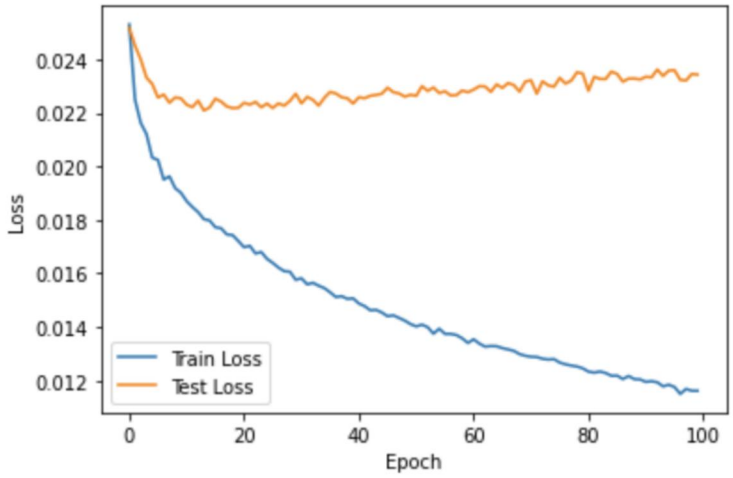

Figure 2: A graph of the train and test loss over each epoch for our baseline RNN model

We trained our 2D CNN model 100 epochs using an Apple M1 Macbook Pro running metal accelerated TensorFlow. The CNN trained much faster and so we were able to use a local environment rather than running on an EC2 instance.

With both models, we explored different sequence lengths to train on. This means selecting a sequence length for input and having the model generate an equal length output. We found an optimal sequence length of about 10 time steps for the RNN and 20 for the CNN. For the CNN model, many different configurations were tried including the aforementioned Dense layers in conjunction with the convolution layers. Ultimately, it was decided that we needed an appropriately deep convolutional network of 4 layers and then to deconvolute the output back to its original size as output.

4.4 Results

Our RNN model was able to successfully generate three 30-second sample containing the five instruments drums, strings, bass, guitar, and piano. After gathering a preliminary round of ratings for the music sample from a pool of 10 participants, the average rating received was a 4.2 out of 10 for the 3 audio samples with baseline_generated2.mp3 and baseline_generated3.mp3 understandably receiving much poorer ratings than baseline_generated.mp3.

Our CNN model proved to produce better results due to the dependent instrumental sampling. Upon analysis, the chords throughout the generated sample are spread out amongst the instruments, making the music less random and more cohesive. All audio samples, which have been converted to mp3 files, can be found here.

Model	**Average Rating**
LPD5 Samples	9.3
Baseline RNN 1	7.1
Baseline RNN 2	5.9
Baseline RNN 3	5.5
Baseline RNN 4	4.7
2D CNN 1	6.9
2D CNN 2	8.4
2D CNN 3	7.9
2D CNN 4	6.7
2D CNN 5	7.1
2D CNN 6	5.5

Table 1: Average Ratings for the Baseline RNN and CNN models.

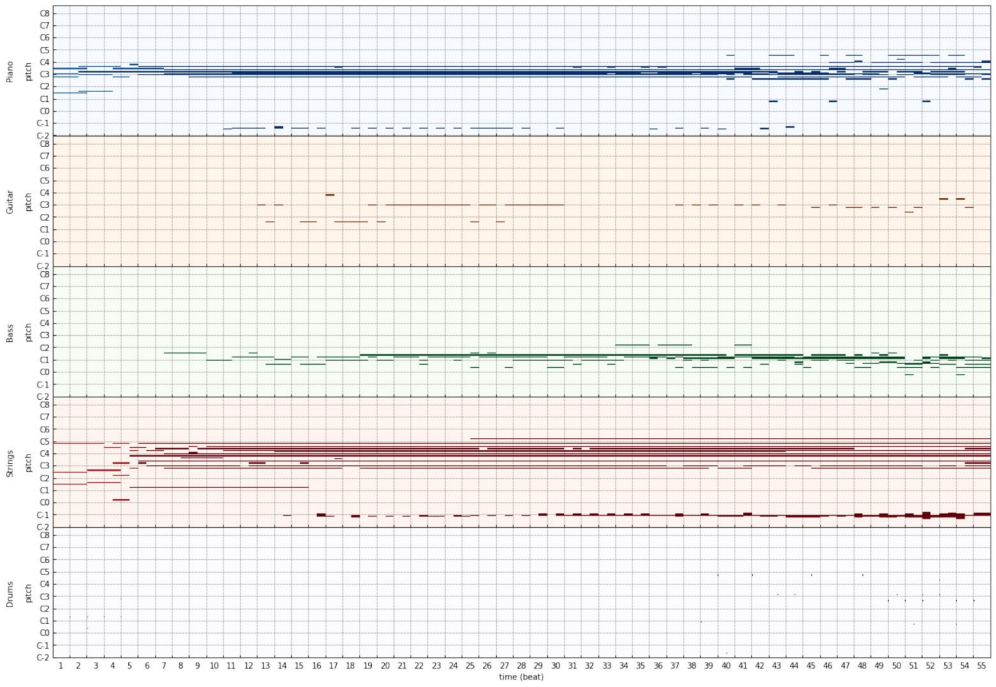

Figure 3: A visualization of the generated pitches for each instrument at each time step

5 Analysis

Overall, the CNN model generally performed better than the baseline RNN model, as expected. For the RNN's generated samples, the highest average rating was 7.1 and the lowest average rating was 4.7. For the CNN's generated samples, the highest average rating was 8.4, and the lowest average rating was 5.5. As mentioned before, the higher enjoyability of the CNN's generated samples is most likely attributed to its use of dependent instrument sampling, which makes the chords at each time step more cohesive among the five instruments. The CNN also exercises better control over not having all 5 instruments fighting for the foreground sound at all times.

Additionally, both our baseline RNN and CNN models fell short of the high average rating (9.3) received by the LPD5 dataset samples. This makes sense because the samples in the LPD5 dataset are more classical-inclined with less dissonance, which sound more familiar to the average person. However, all of the generated samples across both models sounded quite dissonant with a slight creepy quality. This could be because even though the model learned chords for each time step, it does not understand cohesion among the entire piece. Therefore, the resulting audio samples consisted of dissonant chords that were still pieced together.

We also received feedback that baseline_generated2.mp3 and baseline_generated3.mp3, as well as cnn1.mp3 and cnn4.mp3, sounded very similar to each other. This explains the closeness in their scores. This feedback made us realize that the generated music lacked some diversity. We would like to investigate this issue in future work so that our model can generate more creative pieces.

6 Conclusion

We have presented a CNN multi-track music generation model, which has a conditional mechanism to exploit versatile prior knowledge of music while using dependent instrument sampling. Our evaluation shows that it can be a good alternative to RNNs as the generated music samples received higher ratings across the board.

In the future, we would like to explore GANs to see if we can improve the aesthetic quality of our generated samples. This was used in MidiNet, a convolutional generative adversarial network for symbolic-domain music generation that produced desirable results. [3] We would like to explore the method of generating an entire audio clip froma. single latent vector, which allows for easier extraction of features such as pitch and timbre. We are also interested in finding out if implementing GANs can solve the issue of generating similar sounding music.

References

[1] Nikhil Kotecha and Paul Young. Generating music using an lstm network, 2018.

[2] Vaishali Ingale, Anush Mohan, Divit Adlakha, Krishan Kumar, and Mohit Gupta. Music generation using three-layered lstm, 2021.

[3] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation using 1d and 2d conditions, 2017.

[4] 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, 2017.

[5] 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, 2016.

[6] Colin Raffel. Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching.

7 Contributions

All group members contributed equally. We all worked on the code, report, and video presentation components of our final project.