Pop Music Generation

Ronak Malde Department of Computer Science
Stanford University
`rmalde@stanford.edu`

Abstract

This project uses Seq2Seq models to generate pop music with chords and a matching melody. Unlike most state-of-the-art music generation algorithms, which tend to generate a linear sequence of notes, this project aims to emulate the style of pop music that has chords and melody, generated concurrently, that sound good together. The highest performing model found in this project was a system that trained two LSTM models independently - one for chords, and another for the melody, that were trained on the same sequence of note - in order to generate both chords and melodies at each time step. The final result was a generator that both sounded pleasant and emulated the style of pop music.

1 Introduction

The task for this project is to use a Sequence to Sequence models to generate pop music with chords and a melody. There have been many different attempts at generating music in past research, however most focus on generating a linear sequence of notes. In contemporary pop music, however, there is normally a harmony of chords and a melody of notes that are played in coordination. The motivation for this project is to generate both the chords and melody together to simulate pop music. The input for the algorithm is 100 notes and chords in a sequence. I then used an LSTM network to generate a note and chord at each time step. This was repeated to generate a large sequence of notes and chords.

2 Related work

Previous research in this area has used LSTMs for general polyphonic music generation that creates coherent musical pieces. This works by using a sequence of previous notes in order to generate one new note in the sequence, and repeating the process until a whole sequence is generated.[1] [2] This method of generating one new note at each time step is the current state of the art. The limitation of this, however, is that it is difficult to emulate the chords and matching melody of a pop song, because chord sequence depend heavily on the melody and vice-versa.

There are also other Seq2Seq models that can be used to achieve this same generation of music. Transformer models that use Attention across the encoding and decoding of the sequence have proven to be far more powerful than previous LSTM approaches in domains such as NLP. Applied to music, transformer models seem to have promise as well. There is an ongoing OpenAI project called MuseNet that uses transformer models rather than RNNs to generate sequences of music.[3]

3 Dataset and Features

The dataset that I used was the Lakh MIDI Dataset collected by Colin Raffel. There are several different versions of this dataset, but the version titled "LMD Aligned" most fits the needs of this project. There are 45,129 MIDI files contained in the dataset, each of which have been matched to song entries in the Million Song Dataset. The Million Song Dataset is a collection of over a million contemporary popular songs. By having a dataset that contains only songs from the Million Song Dataset, I could be sure that the training data describes pop music, which is the style that I was trying to generate with the model.

I loaded the dataset locally (approximately 1.3 GB zipped, 4.7 GB unzipped) and processed it on a Jupyter notebook. For preprocessing, I used a Python library by MIT called Music21 that can load MIDI files and has predefined note objects and chord objects, whose properties I then stored in an NumPy array to be used in the model. I also used the library to flatten all instruments to piano notes, because some of the scores had different instruments. Below is an example picture of one of the scores in the dataset, where we see the melody in the upper staff, and the chords in the lower staff:

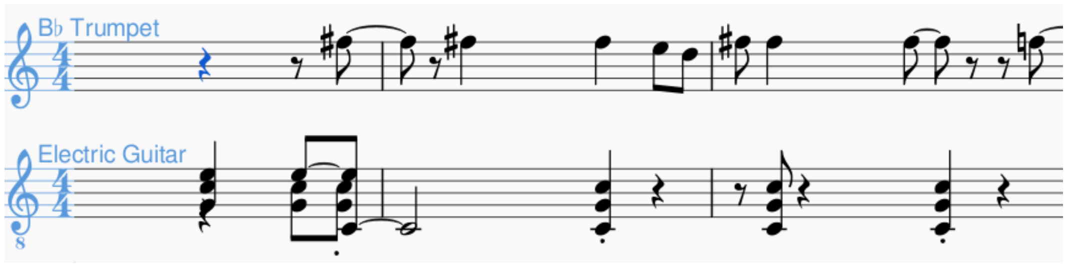

Figure 1: Sample from dataset

I conducted some basic analysis of the dataset to find statistics on the duration and tempo of the song.

	Mean	Std	Min	Max
Duration (seconds)	249.5	126.23	0.31	3034.9
Tempo (BPM)	123.89	35.05	0	302.33

Table 1: Statistics on Lakh MIDI Dataset

I used the mean duration during generation of the notes to generate a song that is approximately 250 seconds, to simulate the dataset. The tempo of the notes is interesting to note, so the LSTM should generate songs that are around 124 BPM.

4 Methods

All of the models constructed use a Seq2Seq structure with an encoder and decoder. During training, the vector of notes is passed into the model encoded by the Music21 library, and mapped to a set of indices, similar to word-embedding methods used in Natural Language Processing. This is because there is a fixed set of notes that the MIDI format supports (128 notes), so there is a "vocabulary size" of 128. During generation, a random selection of notes from the training data is fed into the model as a starter for generation. The model uses the last k=100 previous notes to generate one more additional note or chord to add to the sequence.

Baseline

The baseline model I built was a deep Seq2Seq model, with two LSTM layers of 512 units, and then two dense layers with a softmax activation at the end. I used categorical cross-entropy loss and rmsprop optimizer, built in Keras. For training, I fed in all of the midi note data, without distinguishing the melody from the chords, just to get a simple generation of the notes. Training was for 10 epochs.

Tuned Model

Before adding the functionality of chords and melodies, I first wanted to have the model produce a sequence that generated structures over the full duration, as an improved baseline to be compared against. After tuning hyperparameters and adding a dropout of 0.3, I trained a similar model, this time on 50 epochs, and got much better results, discussed below.

Final Approach

The unique part of this project is to generate both an accompaniment and a melody, and have them sound good together. I tried several approaches to achieve this, but I ultimately settled on the following approach: I created two separate models, one for generating notes and one for generating chords, called note_model and chord_model. Both models take the same input that contains both notes and chords. The, for training, the models are trained separately to optimize for different outputs. For training, note_model was trained to generate notes based on the input. Chords_model was trained to generate chords based on the input. For generating the music, both models are fed a starting sequence of notes and chords. Then, at each time step, note_model generates a note based on the previous k=100 notes and chords, and chord_model generates a chord based on the previous k notes and chords. These generated notes and chords are added to the sequence, which is then used by both models to generate at the next time step. In this manner, the two models work together to generate a sequence of notes and chords.

5 Experiments/Results/Discussion

The hyperparameters that I mainly changed were the batch size and the number of previous notes and chords fed in as input. In order to balance speed of training and risk of overfitting the data, I ultimately used a batch size of 128. And I used a previous sequence length of 100, because I found that this was long enough to capture the musical phrases, but larger than 100 the model started to break down.

To see quantitative results on the generation, I calculated the negative log likelihood of the generation by converting the sequence back to index embeddings. The ground truth of the log likelihood is the notes in the training data following the starter sequence used for generation. The likelihoods shown below are averaged over 10 different starter sequences:

Model	Avg NLL
Baseline	5.830
Baseline Tuned	4.297
Final	2.909

Table 2: Negative Log-likelihood

The NLL of the different models show that the final model is most similar to the likelihood of notes from the training data, whereas the baseline and the tuned baseline are further away.

In order to correctly assess the generation of music, one also needs to qualitatively examine the generated music. Below are samples of the generated MIDI converted to musical notation to easily view:

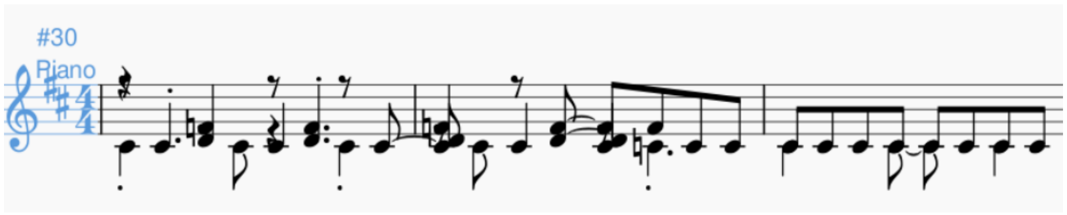

Figure 2: Sample generation from baseline model

Note: for the final model, the generated chords were manually moved to the bass clef of the sheet music, and the melody was manually moved to the treble clef to show the separate generation of chords and melody.

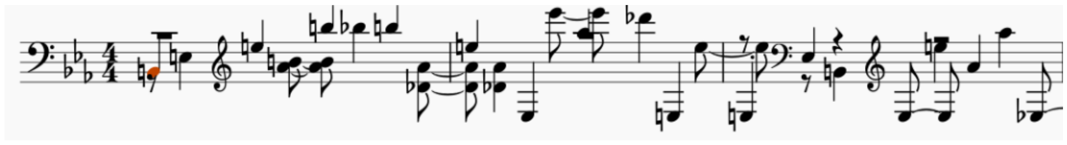

Figure 3: Sample generation from tuned baseline model

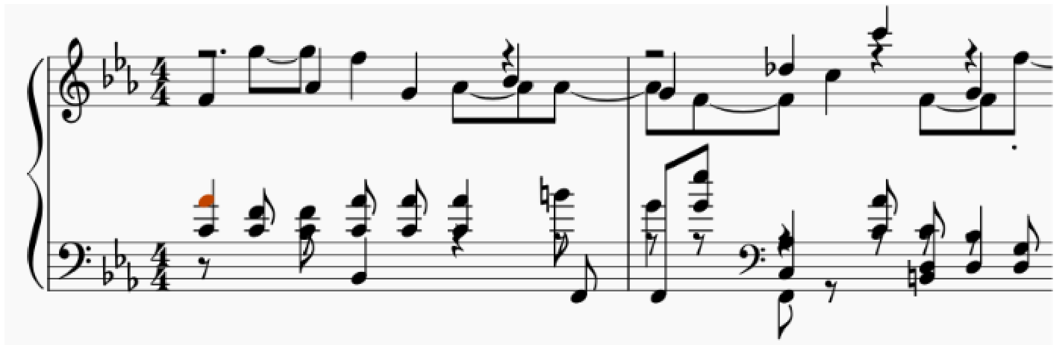

Figure 4: Sample generation from final model

As seen by the sheet music above, the baseline model tended to produce a few varied notes and then converge to one note and continue generating that same note. This is probably why the NLL of the baseline was also extremely high, because there was little to no variation in the generation. The problem of variation was overcome in the tuned baseline. After training more epochs and tuning hyperparameters, the model generated notes in different patterns and often matched phrases from the starter sequence. In addition, the music generated sounded very pleasant, in contrast to the original baseline.

In the final model, we see that the NLL is much lower, showing that it is similar to the training data that was used to start the sequence. In addition, we can see and hear the chords that were generated by the two models. It is also interesting to note that the final model has less accidentals (sharps and flats) in the music than the other models, meaning it better matched the musical key that the starter notes were played in. This model generated a melody and chords, as expected, that both sound pleasant together and emulate the style of pop music. One surprising aspect of the generation is that throughout the generation, the song starts to switch keys from the original starter notes. Everything still sounds good together, but normally pop songs do not switch keys so often.

6 Conclusion/Future Work

I found that by separating the task into two separately trained LSTM's (the final model) that had the same input but trained on different outputs, I was able to generate chords and notes that sounded good together and matched the style of pop music. This achieved the task much better than trying to train one single model that generated both notes and chords, as seen by the qualitative analysis above. Also, generating notes and chords separately helped to maintin structure of the music by having the music stay in the same musical key across several time steps.

As shown in the above sheet music, the chords also generate at every time step when the melody is generated. In normal music, however, the accompaniment chords usually hold for several of the melody notes. For future work, in order to make the music sound more natural and similar to pop music, I would experiment with generating longer accompaniment chords for the shorter melody notes. In addition, the music that I generated did not have the overarching structure of a pop song (verse, chorus, bridge, etc.), so I could also work on a model that retains this long-term structure of the song.

References

[1] Kotecha, N., and P. Young. "Generating Music using an LSTM Network. arXiv 2018." arXiv preprint arXiv:1804.07300.

[2] Choi, Keunwoo, George Fazekas, and Mark Sandler. "Text-based LSTM networks for automatic music composition." arXiv preprint arXiv:1604.05358 (2016).

[3] Payne, Christine. 2019. “MuseNet.” OpenAI. OpenAI. April 25, 2019.

[4] “The Lakh MIDI Dataset v0.1.” 2016. Colinraffel.Com. 2016. https://colinraffel.com/projects/lmd/.