LSTM's for a Generative Model for Video Game Music

Timothy J. Romney
CSE 190: Machine Learning for Music and Audio
University of California, San Diego
9500 Gilman Dr, La Jolla, CA 92093
tromney@ucsd.edu

Abstract

A network architecture using long-short-term memory (LSTM) networks is explored in an attempt to produce a generative model for video game music. A collection of video game music in the form of MIDI sequences, primarily from the Final Fantasy series, is used as the training dataset. A model was trained consecutively using varying values for adjustment (number of epochs, sequence length). The weights were learned at a high computational cost, taking many hours to complete the training processes of 100 epochs. The resulting weights were saved and used to generate sequences of 128 eight notes in length. The results were fairly similar to the input training data, but were still quite unique. The LSTM's use of sequential information was deemed very useful for generating consecutive note sequences.

1 Introduction

LSTM's have a very wide variety of use cases (such as speech or hand-writing recognition) but in this paper, we will just explore the use case of creating a generative model for MIDI sequences. One problem with the output generated from LSTM's is that the output may converge to a single note or a single repeated phrase and get stuck in a 'loop'. However, video game music is often very repetitive, as it must be played continuously in various areas in the game. Using this knowledge, a LSTM seemed to be a more motivated choice than that of other architectures such as a Generative Adversarial Network (GAN) or a Convolutional Neural Network (CNN). Python is used for the implementation of the training process and MIDI generation, with heavy usage of Music21 (a Python toolkit for musicology) and Keras (a high-level API for Tensorflow).

1.1 Background

In an effort to explore resources for learning machine learning, I came across the article "How to Generate Music Using a LSTM Neural Network in Keras." as seen in the 'Reference' section. This gentleman's article served as a stepping stone for me when looking at generative modeling using LSTMs.

The MIDI files used as the training dataset can be found at the following link:

`https://github.com/Skuldur/Classical-Piano-Composer/tree/master/midi_songs`

The video game songs used were compiled in the dataset of 92 MIDI files obtained from various sources on the internet, with a majority of the songs being piano arrangements of the music of Nobuo Uematsu, the composer behind a majority of the songs from video game soundtracks namely for Final Fantasy and Chrono Trigger on the Super Nintendo Entertainment System. .

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

1.2 Related Works

There have been many related studied done concerning LSTM's use for music generation, such as 'Generating Music using an LSTM Network' by Nikhil Kotecha and Paul Young affiliated with Columbia University found below:

`https://arxiv.org/ftp/arxiv/papers/1804/1804.07300.pdf`

Another related study is 'Generation of music pieces using machine learning: long short-term memory neural networks approach' by Nabil Hewahi and their colleagues:

`https://www.tandfonline.com/doi/full/10.1080/25765299.2019.1649972`

These studies, along with many other over the past few years are interesting takes on the use of LSTM network architectures for music generation, and I would suggest these to look over first when diving deeper into the topic.

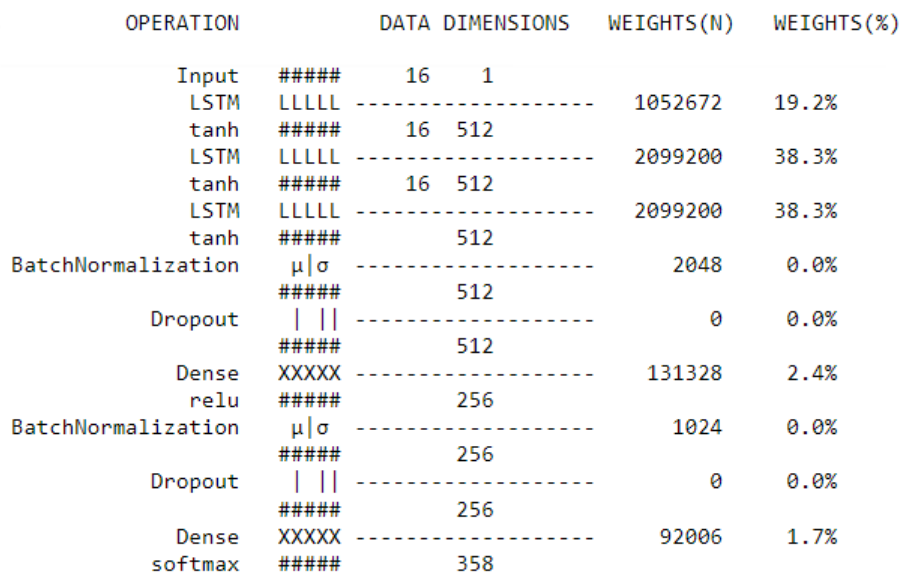
2 Architecture

The input and output of the model are the notes and chords that are possible given the collection of midi files. These are normalized and mapped to integer values in an effort to streamline the training process. One-hot encoding is used in order to encode the categorical labels.

The model contains three LSTM layers in which the first two take sequences and output other sequences and the third outputs a matrix. Dropout layers are used intermediately to help prevent over-fitting (Batch Normalization layers are used before each Dropout layer). Also, the network contains a few dense layers and has both ReLU and softmax activation functions in order to better calculate the output of the network at various points.

The figure below is a direct visualization of the network at each of it's layers:

Figure 1.1: Keras Network Architecture



The model.fit() function from keras is used to train the network and it is trained for 100 epochs with a batch size of 128 samples. These numbers could be tweaked in order to have finer control over the training process. In order to determine how well the model is doing after each epoch, categorical cross entropy is used to calculate the loss for each iteration

3 Experiments

After training the model, MIDI sequences of 30 seconds in length were generated using the weights that were found throughout the iterative training process.

I experimented with different parameters for deciding on the first note, but went with a random starting point is chosen for the sequence, which would be the first note in our generated finite sequence. Different step sizes were also used, but ultimately, a step size of 0.5 with respect to the tempo was used (which translates to eighth notes). This seemed to work nicely since many of the battle themes from final fantasy consist of an eighth note ostinato (a repeated rhythmic phrase).

The main parameter I found that was useful to modulate was the memory length, which was the number of notes the network would be influenced from up to the point in which we are generating a new note. 16 eighth notes (which is the equivalent to 2 measures) was found to be the sweet spot in which the training time did not take excessive amounts of time on my GPU.

4 Results

After training the model with an input memory length of 16 for around 2 hours (60-70 seconds per epoch), the results were shockingly impressive. Although the model may have been able to run for 200 epochs instead of 100, the loss seemed to converge to a value < 1 sooner than expected.

A single MIDI file was generated from the Long short-term memory (LSTM) model and was used to create a song by utilizing various MIDI instruments. In order to give the song some more rhythmic content, a user defined drum loop was also used. Windows Media Player Visualizer was used for the graphic visualizations.

The following link is a one minute long track made by arranging the MIDI file that was generated by the network:

`https://www.youtube.com/watch?v=WFrF8otTROg`

An interesting anomaly that occurred was two outputted sequences using different parameters contained very similar musical information.Figure 2.1 and Figure 2.2 can both be seen to contain a chord structure very similar, but more importantly the exact same bass pattern, alternating from D-flat to F-sharp.

Figure 2.1: Piano Roll A (epoch=100, memory-length=32)

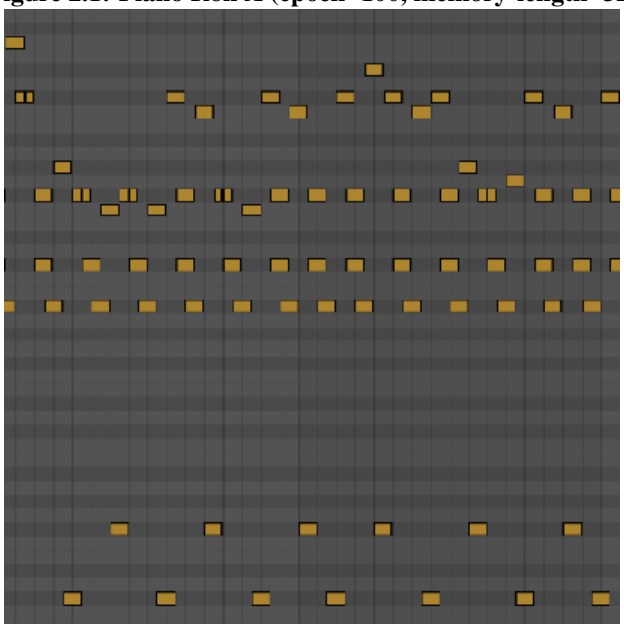
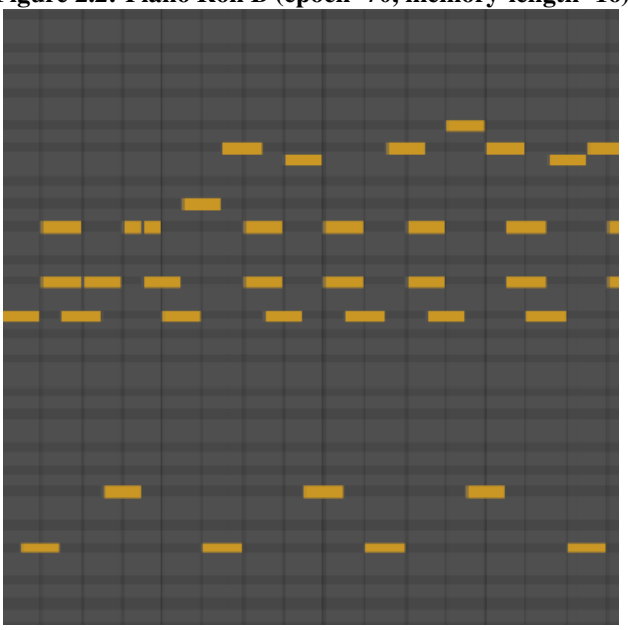


Figure 2.2: Piano Roll B (epoch=70, memory-length=16)



5 Conclusion

In conclusion, the use of LSTM networks seem to be well suited for the task of generating video game music. This is likely due to it's repetitive nature, simple but powerful melodies, simple rhythmic content, and interesting syncopation. The music that is generated consecutively from the model is not always perfect, and there is no current metric for determining quality of output besides a human discriminator. There is a lot of future work that can be done with respect to this project, namely a way of determining output quality and a way of implementing the use of rests and space in the music. For rests to be accounted for, the quantized nature of the MIDI generation would need to be refactored so that the architecture models a probability that there is no note being played.

A link to a GitHub repository containing the project source code is listed below:

`https://github.com/timromney/LSTM_Musical_Generative_Model`

Acknowledgements

I'd like to thank Professor Shlomo Dubnov and TA Ross Greer from CSE 190 - Summer 2020 at University of California, San Diego. They facilitated my study of Machine Learning topics for Music and Audio over the past 5 weeks, and helped me learn the concepts I needed to know in order to finish this project. Also I'd like to acknowledge Sigurður Skúli Sigurgeirsson for being the inspiration for this project, his work helped get me started looking at LSTM's for music generation.

References

[1] Skúli, Sigurður. "Skuldur/Classical-Piano-Composer." GitHub, github.com/Skuldur/Classical-Piano-Composer.

[2] Skúli, Sigurður. "How to Generate Music Using a LSTM Neural Network in Keras." Medium, Towards Data Science, 9 Dec. 2017, towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5.