Generating Music Using GAN and LSTM

Yatri Patel
Department of Engineering and Computer Science
University of Tennessee at Chattanooga
jwb318@mocs.utc.edu

***Abstract*—Neural networks have been generating texts and images since last few years, but generating music is very different from generating video or an image. While the image made by a neural network was sold for over $400,000, which lead us to question that can such models also generate music.In this project, I have generated music with Generative Adversarial Networks (GANs) and Long Short Term Memory (LSTM) Models, based on a particular type of music. The two different types of approaches used in the paper make use of the musical instrument digital interface (MIDI) file format for easier access and better understanding of the music. Each of the model produced a distinct melodies, with a particular set of input.**

I. Introduction

The art of ordering tones or sound in succession, in combination is music. It is a temporal relationship to produce a composition of notes having continuity and unity. In other words, a musical note is any sound generated with the help of musical instruments or human voice. A musical note is a simple unit of music. Music and its notes have certain properties [1] concerning its quality and performance. This project is inspired [1], [5]

One method of generating music by utilizing existing music is genetic algorithm [2]. As stated in [2] genetic algorithm can highlight the strong rhythm in each fragment and combine them into distinct pieces of music. But it has low efficiency because every iteration process of it has a delay. It also lacks in deep stated rhythm recognition, which makes it inefficient.

So, in order to overcome the above problem we require a system that should be able to remember the previous note sequence and predict the next sequence and so on. This brings us to the Long Short Term Memory Model which is a version of Recurrent Neural Network (RNN). While, using LSTM in the General Adversarial Networks is also a way to overcome the above stated problem of the genetic algorithm.

In this project, I have used both the GAN and LSTM method to generate music. Further, in the paper, I will discuss about the data, its pre-processing and the model itself.

II. Data

One of the major difficulty while working with music, is representation of data. Hence, here I have used MIDI file format because firstly, it bears characteristics of a song in its metadata and secondly, it is commonly used, as considerable number of data sets are available. For the downloading the MIDI files, I web scrapped the data from [6] using the key words. Each different type of data set based on the keyword is used an an input to generate music of that specific category. Here, I have generated music using Movie and Piano themed music input.

Using the Music21 module, I was able to pre-process the data to get respective notes and chords. Music21 allows to read each track's notes and chords (groups of notes) into Python while maintaining their sequence. Each note is represented as its specific note-type or a corresponding number.

On converting each MIDI file into a stream object, I was able to get a list of all the notes and chords in the file. The pitch of every note object is appended using its string notation since the most significant parts of the note can be recreated using the string notation of the pitch. And we append every chord by encoding the id of every note in the chord together into a single string, with each note being separated by a dot. These encoding allows to easily decode the output generated by the network into the correct notes and chords.

I mapped the string-based categorical data to integer-based numerical data. This is done because neural networks perform far better on the numerical data. Next an input sequences is created for the network with their respective outputs. The output for each input sequence will be the first note or chord that comes after the sequence of notes in the input sequence in our list of notes. Finally, I normalized the input normalise the input and one-hot encode the output.

III. Model

Two types of model I have used for generating music are LSTM and GAN. LSTM cells are an advanced form of recurrent neural network (RNN) which are capable of remembering long-term dependencies which are used in this paper. GANs consist of a discriminator and a generator model which are trained and tested simultaneously. Further details about each model are discussed in the following sub sections.

A. LSTM based model

For LSTM model to generate music, here I have chosen a simple network consisting of three LSTM layers, three Dropout layers, two Dense layers and one activation layer, along with batch normalization layers to generate music. In order to calculate the loss for each iteration of the training I have used categorical cross entropy since each of our outputs only belongs to a single class and we have more than two classes to work with. While for the optimisation function of our network we will use a RMSprop optimizer as it is usually a very good choice for recurrent neural networks. The architecture of the model could be seen in Fig. 1.

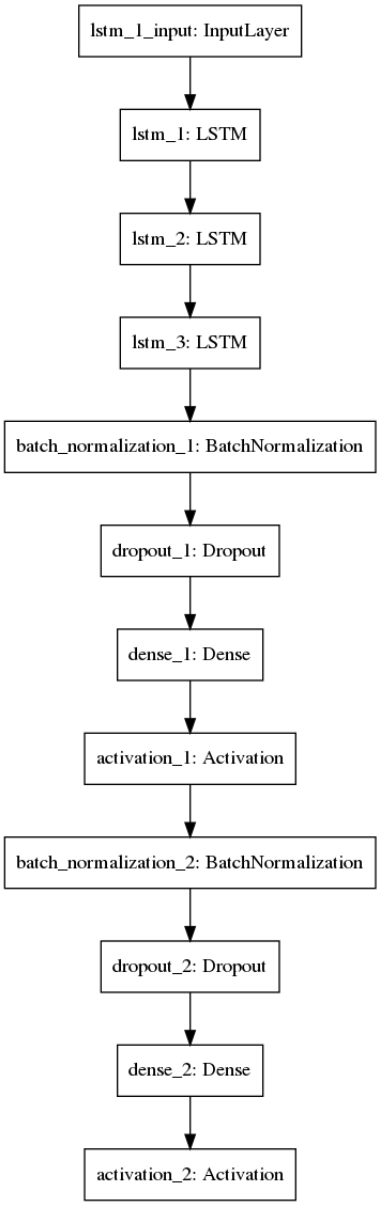

Fig. 1. LSTM model architecture

B. GAN based model

In GAN, the discriminator is presented real data — in our case, real songs — as well as fake data generated from random noise. For each sample, the discriminator's task is to correctly classify data as either real or fake. On the other hand, the generator's task is to generate fake data from random noise that is able to fool the discriminator into making more classification mistakes (i.e. calling a real song "fake" or vice versa). By stacking these two models, the discriminator and the generator models begin to compete against each other with the desired end result in our case being a generator that can easily fool the discriminator.

There are already a handful of articles that utilize a GAN for making music, such as MuseGAN and C-RNN-GAN [3], [7].

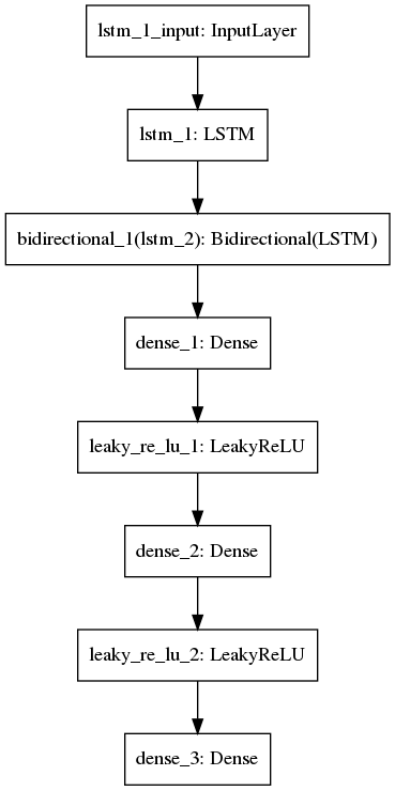

Fig. 2. LSTM model architecture

For my project, I decided to base my GAN off of the C-RNN-GAN which is a recurrent neural network with adversarial training.

The model architecture for discriminator and generator is shown in Fig. 2 and Fig. 3 respectively. The input of our generator model is similar to that of the LSTM model, except that the data has been normalized between +1 and -1. This is done to make things easier on the generator network, which is being fed random noise from a standard normal distribution with the expectation that it will generate output ranging between -1 and 1.

In this model, the first two layers are LSTM layers, one is a normal LSTM layer while the 2nd layer is a bidirectional LSTM layer, which allows the discriminator to learn from the music inputs as sequential data during training. Bidirectional LSTM layer is layer which involves duplicating the first recurrent layer in the network so that there are two layers side-by-side, which on providing the input sequence as-is as input to the first layer and providing a reversed copy of the input sequence to the second. Without these layers, the discriminator was unable to distinguish real music from fake music as long as the generator was able to figure out the discrete domain of the input data. With the LSTM, the generator now has to do more than just finding the domain of the real data; it also needs to learn that music follows certain patterns. Finally, I chose a sigmoid activation function, as we want out outputs to be a single 0 or 1, representing fake data or real data, respectively.

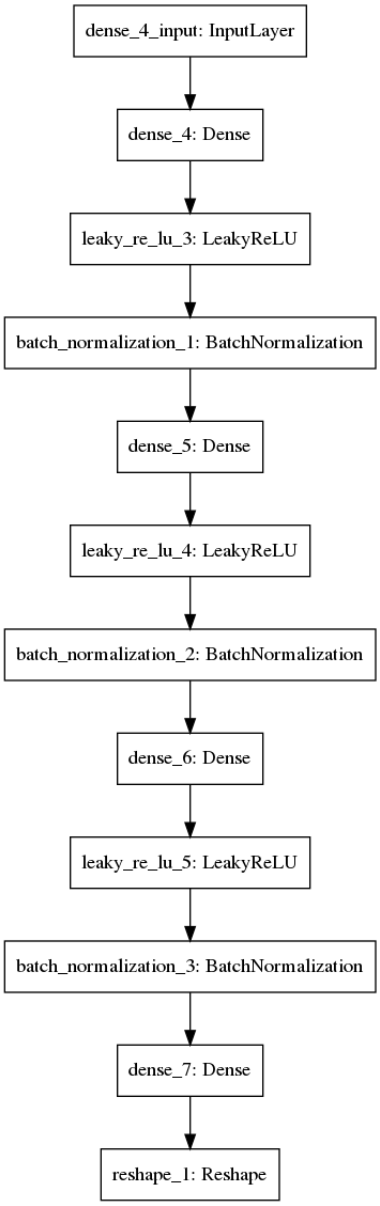

Fig. 3. LSTM model architecture

For the generator, the network architecture is simply a multi-layered perceptron that received random inputs equal to the size of the latent dimension (we used 1000 as it led to better learning with minimal slowdown when compared to a size of 100). With these two networks created, we can begin training a stacked model by giving the generator a batch of random noise from a standard normal distribution and have it process this noise in order to make a batch of sequences of 100 notes encoded as numbers. This data is then passed to the discriminator as a set of fake data, and the discriminator goes through a training iteration. Afterwards, the generator makes another set of fake data, and this is passed to the stacked model as fake data to train both the discriminator and the generator.

C. Music Generation

After training both of the models, we want to generate new MIDI files as our output files. For both models, we want to use the model to make a prediction from an input and create an encoded output that corresponds to a sequence of notes and chords. For the LSTM model, we sample a 100 note sequence from the corpus and input that into the network and have it make a prediction, which is our encoded output. For the GAN model, we feed the generator a random sequence of numbers sampled from a standard normal distribution and have it make its prediction. With these predictions, we can use Music21 module to turn our predicted sequences into new MIDI files.

IV. Results

The music generated by both the models can be heard on [4].

When comparing the output of the two models, we found that the GAN is superior in terms of authenticity in its output MIDIs and its training time. The LSTM model tends to start each song by repeating the same note for several seconds before adjusting. With faster training, GAN was able to give out better outputs in terms of stuck chords, while on the other hand LSTM took longer to train, but had a better output when trained for 200 epochs, taking almost 6 hours.

The plots for the two different types of music generated using LSTM and GAN areas below:

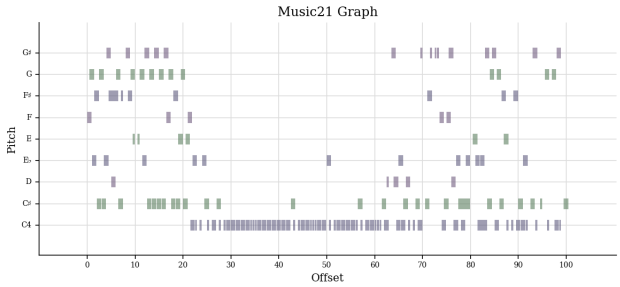

Fig. 4. GAN - Plot of note quarter length by pitch for generated piano themed music

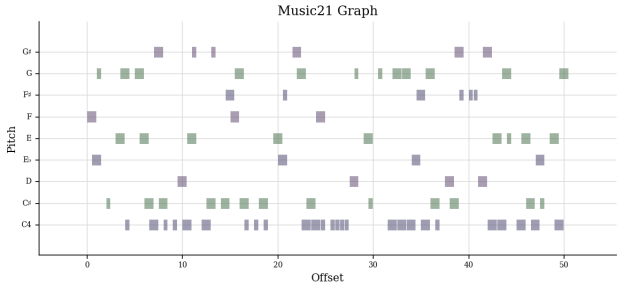

Fig. 5. GAN - Plot of note quarter length by pitch for generated movie themed music

Fig. 4 shows the plots generated on the music generated by GAN model using the piano themed single instrument music as input. The music generated does have stuck chords between

offset 30 and 40, and could be improved by training for a longer time 4. But the one generated on the movie themed music have less stuck chords and has more definite rhythm to it 5. This is also justified by the Loss vs epoch graph in Fig. 6 and Fig. 7, where after a particular epoch the discriminator and generator start competing with each other.

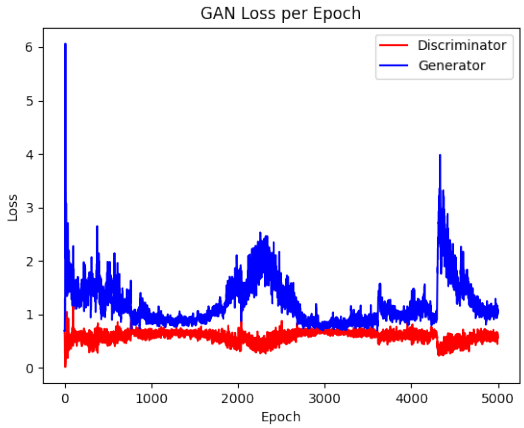

Fig. 6. GAN - Loss vs Epoch graph for piano themed music generated

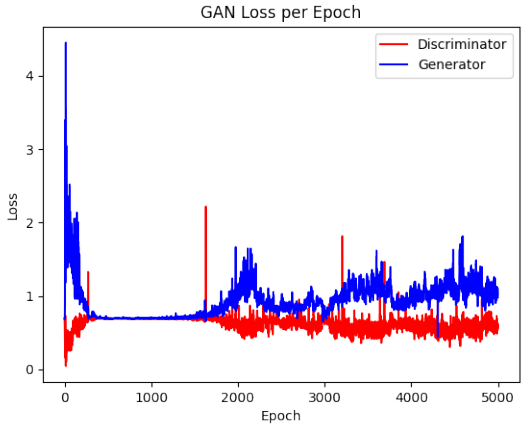

Fig. 7. GAN - Loss vs Epoch graph for movie themed music generated

While for the LSTM model outputs, the music generated had a lot more pattern recognition and repetition as could be seen in Fig. 9 and 8. Although even the LSTM model had stuck chords, it was less prominent in the output compared to that of the GAN models. LSTM model was able to use a wide variety of notes, chords and pitches compared to GAN.

V. Conclusion

With the models we have used above, even the best music generated has some areas with stuck cords, which were more prominent in the GAN model rather than the LSTM model. With more data and more epochs I think we would be able to overcome the stuck chords, as in the LSTM models trained

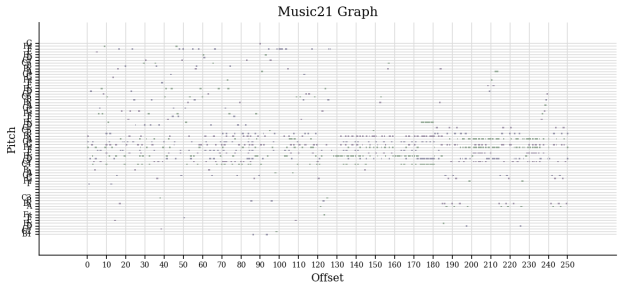

Fig. 8. LSTM - Plot of note quarter length by pitch for generated movie themed music

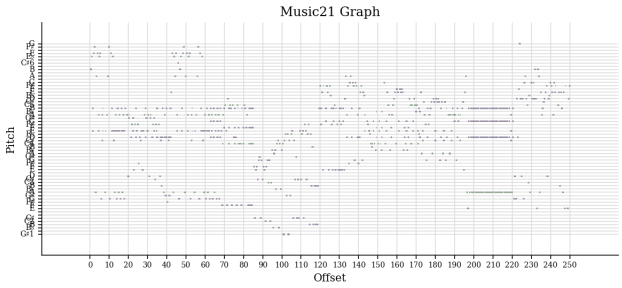

Fig. 9. LSTM - Plot of note quarter length by pitch for generated piano themed music

for 200 epochs there were less number of stuck chord sections than the ones in the model with less number of epochs.

Where the LSTM outperforms the GAN is in terms of fixating on certain patterns. We found that the LSTM is better able to hone in on specific patterns and recycle them throughout the course of a song. The model is also able to get out of certain note loops and transition into another set of note loops. As far as the GAN is concerned, it can only pick up on basic concepts, such as escalating notes in a low-to-high fashion, and does not fall into the more nuanced patterns.

Music being a qualititive entity, both the models performed well, but LSTM out performed with its ability to learn patterns and repeat them in the song along with its wide use of chords and notes. While on the other hand GAN was able to develop a song with confined chords and notes. We can overcome and improve the output using deeper networks of generator and discriminator, which could be done by increase the LSTM layers in the neural network.

With limited computing power and memory, I was able to train my LSTM model for 200 epochs, which took almost 6 hours to train while GAN took around 2.5 hours for training on 5000 epochs. Hence with higher computing power an using deeper networks, we would be able to get better outputs from GAN.

References

[1] Agarwala, Nipun, Yuki Inoue, and Axel Sly. "Music composition using recurrent neural networks." CS 224n: Natural Language Processing with Deep Learning, Spring (2017).

[2] Alfonseca, Manuel, Manuel Cebrián, and Alfonso Ortega. "A simple genetic algorithm for music generation by means of algorithmic information theory." 2007 IEEE Congress on Evolutionary Computation. IEEE, 2007.
[3] Dong, Hao-Wen, et al. "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.
[4] "Generated Music Playlist." https://soundcloud.com/yatri-patel-793078277/sets/lstm-gan-neural-network."
[5] Mangal, Sanidhya, Rahul Modak, and Poorva Joshi. "LSTM Based Music Generation System." arXiv preprint arXiv:1908.01080 (2019).
[6] "MIDI File data source." https://www.midiworld.com/
[7] Mogren, Olof. "C-RNN-GAN: Continuous recurrent neural networks with adversarial training." arXiv preprint arXiv:1611.09904 (2016).

APPENDIX

Other types of graphs and code could be seen in the GitHub repository : https://github.com/yatri1609/Music-Generation