Music Generation Using LSTM Model

Sankalp Wanjari #, Shubham Patil\*, Swapnil Thorat, Vinod Shende, Pranav Tupe #Prof. Sunil. A. Sushir

[sankalpwanjari85@gmail.com](mailto:Sankalpwanjari85@gmail.com)

[shubhamspatil1705@gmail.com](mailto:shubhamspatil1705@gmail.com)

[swapnilsthorat93@gmail.com](mailto:swapnilsthorat93@gmail.com)

[shendevinod9585@gmail.com](mailto:shendevinod9585@gmail.com)

[pranavtupe0508@gmail.com](mailto:pranavtupe0508@gmail.com) [sunils@isquareit.edu.in](mailto:sunils@isquareit.edu.in)

#Computer Engineering Department, International Institute of Information Technology, Pune

# *Abstract—* Music was traditionally considered as an analogue signal that was manually manufactured. Music has become increasingly visible to technology in recent years, which can now compose a suite of music without the need for human participation. To complete this goal, we must overcome a number of technological hurdles, which are described in detail in this work. The article includes a brief introduction to music and its components, as well as citations and analyses of relevant work done by many scholars in this field. The main goal of this study is to offer a method for generating musical notes using Recurrent Neural Networks (RNN), specifically Long Short-Term Memory (LSTM) networks. This technique is implemented using a model in which data is represented using the musical instrument digital interface (MIDI) file format for easy access and comprehension. Data pre-processing before putting it into the model, revealing reading techniques. It’s also covered how to process and prepare MIDI files for input. Over a single-layered LSTM network, the model utilized in this article is used to learn polyphonic musical note sequences. For greater learning, the model must be able to retain prior features of a musical sequence and its structure. This paper presents a description of the layered architecture employed in the LSTM model, as well as its interweaving connections for developing a neural network. This study provides a sneak glimpse into weight and bias distributions in each layer of the model, as well as a detailed depiction of losses and accuracy at each step and batch. After a comprehensive examination, the model delivered excellent results in the creation of new tunes.

***Keywords*— Music, Melodies, RNN, LSTM, Neural Network.**

I. INTRODUCTION

This paper construes an algorithm (Neural Network) based on LSTM networks which can be used to generate music and melodies automatically without any human intervention. The key goal is to develop a model which can learn from a set of musical notes, analyze them and then generate a pristine set of musical notes. This task is a real challenge because the model must have capabilities to recall past details and structure of musical notes for future projection of learning sequence. The model needs to learn the original sequences adjacent to past one and transform it for the learning system. This is motivated by the fact that while in principle any set of notes in music can be combined to form a chord, in practice only a few combinations are used. Thus, the presence or absence of certain notes can be used to infer whether or not a certain different note or group of notes might be likely to occur at the same time. Combined with an RNN to model probabilities along the time axis, this means that RNN is able to more accurately model polyphonic music than simpler networks.

1. LITERATURE SURVEY

Allen Huang states previous work in music generation has mainly been focused on creating a single melody.[1] More recent work on polyphonic music modelling, centered around time series probability density estimation, has met some partial success. One of the earliest papers on deep learning- generated music, written by Chen et al, generates one music with only one melody and no harmony. The authors also omitted dotted notes, rests, and all chords.[1] Midi files a restructured as a series of concurrent tracks, each containing a list of meta messages.[1].

They used a 2-layered Long Short-Term Memory (LSTM) recurrent neural network (RNN) architecture to produce a character level model to predict the next note in a sequence. [1] In their midi data experiments, they treated a midi message as a single token, whereas in piano roll experiment, they treated each unique

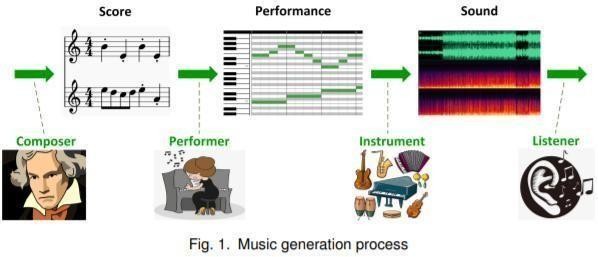
combination of notes across all time steps as a separate token. [1] Their architecture allowed the user to set various hyper parameters such as number of layers, hidden unit size, sequence length, batch size, and learning rate. [1] They also anneal their learning rate when they see that the rate of training error is decreasing slowly. The conclusion by Allen Haung is to show that a multi- layer LSTM, character-level language model applied to two separate data representations is capable of generating music that is at least comparable to sophisticated time series probability density techniques prevalent in the literature.

Li-Chia Yang proposed CNN-GAN based system named MidiNet which converts a noise into midi files using convolutional neural networks (CNNs).[2] In this model using CNN for generating melody (a series of MIDI notes). [2] In addition to the generator, it uses a discriminator to learn the distributions of melodies, making it a generative adversarial network (GAN).

It uses random noises as input to generator CNN. [2] The goal of the generator is to transform random noises into real midi file. [2] Meanwhile, a discriminator CNN that takes input from generator and predicts whether it is from a real or a generated midi, thereby informs the generator how to appear to be real. [2] This amounts to a generative adversarial network (GAN), which learns the generator and discriminator iteratively. It shows that it can be powerful alternative to RNNs.

* 1. OBJECTIVES

One key challenge with modeling music is selecting the data representation. Possible representations are signal, transformed signal, MIDI, text, etc. A relevant issue is the end destination of the generated music content. The format destination could be a human user, in which case the output would need to be human readable, for instance a musical score. The final output format is therefore readable by a computer, which in this case is a MIDI file. Another relevant factor is the level of supervision in the generation of the output. At one extreme is complete autonomy and automation with no human supervision

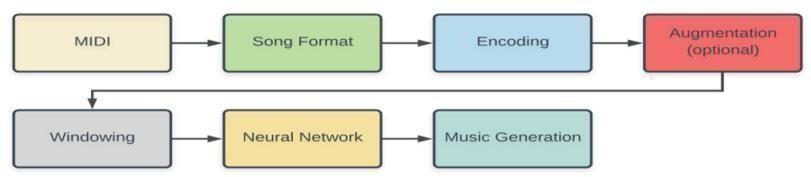


* 1. DATASET

This dataset is a subset (clean) of the Lakh MIDI dataset. The Lakh MIDI dataset is a collection of 176,581 unique MIDI files, 45,129 of which have been matched and aligned to entries in the Million Song Dataset. Its goal is to facilitate large-scale music information retrieval, both symbolic (using the MIDI files alone) and audio content- based (using information

extracted from the MIDI files as annotations for the matched audio files).

* 1. METHODOLOGY



*Fig 1. System Architecture*

The model is applied on polyphonic musical notes. The LSTM network is trained to acquire the knowledge of probability of occurrence of a musical note at current time. The output of the network at a time step t, conditioned on the previous notes’ state till time step t- 50 are fed into the input unit to recall past details and structure of notes. The LSTM layer depends on a selected input. Not all notes undergo the training process. Only some selected and specified notes are used to train this LSTM model, which are useful for effectively tuning the model that result in efficient information gain. With these inputs, LSTM layer learns the mapping and correlation between notes and their projection. Next to the LSTM layer, Dropout layer is used to create generalizations in the model. Once the model has learned the probability distributions of notes and sequences, we must combine all the LSTM cells with each other. This gap is subordinated with the help of Dense layer. Dense layer ensures that the model is fully connected. At the endpoint, the Activation layer is added to the model, which helps in deciding, which neurons (LSTM cells) should be activated and whether the information gained by the neuron is relevant, making activation function highly important in a deep neural network. After training the LSTM network, the model is ready to generate a new sequence of musical notes. To ensure better prediction and diverse output of sequences, a large and varied dataset was elicited with different variations in the structural composition of musical notes. The goal was to expose the model with diverse dataset which would lead to a better tuning of the model. The MIDI file format was used to extract dataset. MIDI files played an important role in extracting information about note sequence, note velocity and the time component.

The model is applied on polyphonic musical notes. The LSTM network is trained to acquire the knowledge of probability of occurrence of a musical note at current time. The output of the network at a time step t, conditioned on the previous notes’ state till time step t- 50 are fed into the input unit to recall past details and

structure of notes. The LSTM layer depends on a selected input. Not all notes undergo the training process. Only some selected and specified notes are used to train this LSTM model, which are useful for effectively tuning the model, that result in efficient information gain. With these inputs, LSTM layer learns the mapping and correlation between notes and their projection. Next to the LSTM layer, Dropout layer is used to create generalizations in the model. Once the model has learned the probability distributions of notes and sequences, we must combine all the LSTM cells with each other. This gap is subordinated with the help of Dense layer. Dense layer ensures that the model is fully connected. At the endpoint, the Activation layer is added to the model, which helps in deciding, which neurons (LSTM cells) should be activated and whether the information gained by the neuron is relevant, making activation function highly important in a deep neural network. After training the LSTM network, the model is ready to generate a new sequence of musical notes. To ensure better prediction and diverse output of sequences, a large and varied dataset was elicited with different variations in the structural composition of musical notes. The goal was to expose the model with diverse dataset which would lead to a better tuning of the model. The MIDI file format was used to extract dataset. MIDI files played an important role in extracting information about note sequence, note velocity and the time component. A model was compiled using Moon et al. as a suggested guide for dropouts, Dropout of 0.75 was applied to the LSTM layer. The optimizer selected was RMSprop. The learning rate selected was 1e-4 for model optimization.

* 1. IMPLEMENTATION

1. Dataset: The dataset we utilized is a subset (clean) of the Lakh MIDI dataset. The Lakh MIDI dataset is an assortment of 176,581 special MIDI documents, 45,129 of which have been coordinated and adjusted to passages in the Million Song Dataset. From this subset, we incorporate just those tunes for which we had the option to get type data, as characterized by their relating appearance in the MSD. We had the option to obtain this sort data for 3827 tunes, including the accompanying types: Pop Rock, Folk, Country, Electronic, Blues, Latin, Reggae, RnB, Rap, International, Vocal, New Age, and Jazz.
2. Import libraries: Music 21 is a Python library created by MIT for figuring out music information. MIDI is a standard organization for putting away music documents. MIDI represents Musical Instrument Digital Interface. MIDI documents contain the directions as opposed to the genuine sound. Thus, it possesses next to no memory. That is the reason it is for the most part liked while moving documents.
3. Reading and Understanding the Data: The initial step to executing the brain network is to inspect the information we will be working with. The information parts into two article types: Notes and Chords. Note

objects contain data about the pitch, octave, and offset of the Note. Pitch alludes to the recurrence of the sound, or how high or low it endlessly is addressed with the letters [A, B, C, D, E, F, G], with A being the most noteworthy and G being the least. Octave alludes to which set of pitches you use on a piano. Counterbalance alludes to where the note is situated in the piece.

1. Preparing the Data: In the first place, we will stack the information into an array. We start by stacking each document into a Music21 stream object utilizing the converter. Parse(file) work. Utilizing that stream object, we get a rundown of the relative multitude of notes and harmonies in the document. We annex the pitch of each and every note object utilizing its string documentation since the main pieces of the note can be reproduced utilizing the string documentation of the pitch. Furthermore, we add each harmony by encoding the id of each and every note in the harmony together into a solitary string, with each note being isolated by a speck. Since we have placed every one of the notes and harmonies into a consecutive rundown, we can make the groupings that will act as the contribution of our organization. To begin with, we will make a planning capacity to plan from string-based all-out information to whole number based mathematical information. This is done on the grounds that brain network performs much better with whole number based mathematical information than string- based unmitigated information Then, we need to make input arrangements for the organization and their separate results. The result for each information grouping will be the principal note or harmony that comes after the arrangement of notes in the info succession in our rundown of notes. The last advance in setting up the information for the organization is to standardize the information and one- hot encode the result.
2. Model: At long last we get to planning the model engineering. In our model we utilize four unique kinds of layers: LSTM layers, Dropout layers, Dense layers or completely associated layers, Activation layer. For each LSTM, Dense, and Activation layer the principal boundary is the number of hubs the layer that ought to have. For the Dropout layer the main boundary is the small portion of info units that ought to be dropped during preparing. For the principal layer we need to give a novel boundary called input shape. The reason for the boundary is to illuminate the organization regarding the state of the information it will prepare. The last layer ought to continuously contain similar measure of hubs as the number various results our framework has. This guarantees that the result of the organization will plan straightforwardly to our classes.
3. Generating Music: To have the option to utilize the brain organization to create music you should place it into similar state as before. Now we can utilize the prepared model to begin producing notes. Since we have a full rundown of note groupings available to us, we will pick an irregular record in the rundown as our beginning stage, this permits us to rerun the age code without transforming anything and come by various

outcomes like clockwork. Here we likewise need to make a planning capacity to disentangle the result of the organization. This capacity will plan from mathematical information to all out information (from numbers to notes). Then, at that point, we gather every one of the results from the organization into a solitary exhibit. Since we have every one of the encoded portrayals of the notes and harmonies in a cluster, we can begin translating them and making a variety of Note and Chord objects. In the event that the example is a Chord, we need to separate the string into a variety of notes. Then, at that point, we circle through the string portrayal of each note and make a Note object for every one of them. Then we can make a Chord object containing every one of these notes. On the off chance that the example is a Note, we make a Note object utilizing the string portrayal of the contribute contained the example. Since we have a rundown of Notes and Chords produced by the organization, we can make a Music21 Stream object involving the rundown as a boundary. Then at last to make the MIDI document to contain the music produced by the organization we utilize the compose work in the Music21 tool stash to compose the stream to a record.

* 1. CONCLUSIONS

This system achieves the goal of designing a model which can be used to generate music and melodies automatically without any human intervention. The model is capable to recall the previous details of the dataset and generate a polyphonic music using a single layered LSTM model, proficient enough to learn harmonic and melodic note sequence from MIDI files of Pop music.

* 1. FUTURE SCOPE

Future work will aim to test how well this model scales on much larger dataset, such as the Million Song Dataset. We would like to observe effects on this model by adding more LSTM units and try different combinations of hyper parameters to see how well this model performs. We believe follow up research can optimize this model further with lots of computation

We have generated a good quality music, but there is a huge scope of improvement in it. First, starting and ending music can be added in every new generated tune to give a tune a better start and better ending. By doing this, our generated music will become melodious. Second, the model can be trained with more tunes. Here, we have trained our model with only 405 musical tunes. By training the model with more musical tunes, our model will not only expose to more variety of music but the number of classes will also increase. By this more melodious and at the same time more variety of music can be generated through the model. Third, model can also be trained with multi-instrument tunes. As of now, the music generated is of only one piece of instrument. It would be interesting to listen what music the model

will produce if itis trained on multi-instrument music. Finally, a method can be added into the model which can handle unknown notes in the music. By filtering unknown notes and replacing them with known notes, model can generate more robust quality music.

REFERENCES

1. Lejaren A. Hiller and Leonard M. Isaacson, ‘Experimental Music: Composition with an Electronic Computer’, McGraw- Hill. (2011)
2. Jose D. Fern´andez and Francisco Vico, ‘AI methods in algorithmic composition: A comprehensive survey’, Journal of Artificial Intelligence Research, (2013).
3. Hochreiter S, Schmidhuber J. ‘Long short-term memory[J]’. Neural computation, (2009).
4. Mark Steedman. ‘A generative grammar for Jazz chord sequences’. Music Perception, (2015)
5. Franc¸ois Pachet, Pierre Roy, and Gabriele Barbieri. ‘Finite-length Markov processes with constraints.’ In Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), (July 2011).
6. Jer´ ome Nika, Marc Chemillier, and G ˆ erard Assayag, ´ ‘Improtek: introducing scenarios into human-computer music improvisation,’ Computers in Entertainment (CIE), (2016).
7. David Cope and Melanie J Mayer, ‘Experiments in musical intelligence’, AR editions Madison, (1996).
8. Raymond P Whorley and Darrell Conklin, ‘Music generation from statistical models of harmony,’ Journal of New Music Research, vol. 45, no. 2, (2016).
9. Derek Wells and Hala ElAarag, ‘A novel approach for automated music composition using memetic algorithms,’ in Proceedings of the 49th Annual Southeast Regional Conference. ACM, (2011).
10. Mason Bretan, Gil Weinberg, and Larry Heck, “A unit selection methodology for music generation using deep neural networks,” arXiv preprint, (2016).