Music Generation Using LSTM Model

Sankalp Wanjari
Department of Computer Engineering,
Hope Foundation's International
Institute of Information Technology,
Pune, India
sankalpwanjari85@gmail.com

Shubham Patil
Department of Computer Engineering,
Hope Foundation's International
Institute of Information Technology,
Pune, India
shubhamspatil1705@gmail.com

Pranav Tupe
Department of Computer Engineering,
Hope Foundation's International
Institute of Information Technology,
Pune, India
pranavtupe0508@gmail.com

Swapnil Thorat
Department of Computer Engineering,
Hope Foundation's International
Institute of Information Technology,
Pune, India
swapnilsthorat93@gmail.com

Vinod Shende
Department of Computer Engineering,
Hope Foundation's International
Institute of Information Technology,
Pune, India
shendevinod9585@gmail.com

Prof. Sunil. A. Sushir
Department of Computer Engineering,
Hope Foundation's International
Institute of Information Technology,
Pune, India
sunils@isquareit.edu.in

***Abstract*—Music was once thought of as an analog signal that was created by hand. In recent years, technology has become more aware of music, and it is now capable of creating a musical suite without the assistance of a human composer. We face a variety of technological challenges in achieving this goal, which are thoroughly discussed in this study. The article gives a quick overview of music and its elements and cites and evaluates a wide range of pertinent studies conducted by academics in this discipline. In particular, Long Short-Term Memory (LSTM) networks from recurrent neural networks (RNN) are used in this study to provide a mechanism for producing musical sounds. This method is put into practice utilizing a model in which data is provided using the MIDI file format for straight forward access and comprehension. Data pre-processing, which reveals reading methods, is done before it is entered into the model. The processing and preparation of MIDI files for input are also covered. The model employed in this article is used to learn polyphonic musical note sequences across a single-layered LSTM network. The model needs to be able to remember previous aspects of a musical sequence's structure in order to facilitate learning. The layered architecture used in the LSTM model and its intertwining connections for creating a neural network are described in this paper. This study includes a detailed illustration of losses and accuracy at each step and batch in addition to a sneak peek into weight and bias distributions in each layer of the model. Following a detailed analysis, the model produced outstanding outcomes when it came to writing new songs.**

***Keywords*— Music, Melodies, MIDI, RNN, Data pre-processing, LSTM, Neural Network.**

I. INTRODUCTION

This study develops a neural network technique based on LSTM networks that can be used to create melodies and music automatically, without the need for human input. The main objective is to create a model that can study a set of musical notes, learn from them, and then produce a perfect set of notes. This problem is difficult since the model needs to be able to remember past information and the structure of musical notes in order to project future learning steps. The original sequences that are next to the previous one must be learned by the model and transformed for the learning system. The reason for this is that, despite the fact that any combination of musical notes can, in theory, result in a chord, very few combinations are actually used. Therefore, it is possible to determine whether a particular note or set of notes is likely to occur at the same moment from the presence or absence of other notes. This implies that RNN can model polyphonic music more precisely than simpler networks when combined with an RNN to predict probability along the time axis.

The reason for this is that, despite the fact that any combination of musical notes can, in theory, result in a chord, very few combinations are actually used. As a result, it is possible to determine whether or not a certain note or combination of notes is likely to occur at the same moment based on the presence or absence of other notes. This indicates that RNN is able to model polyphonic music more accurately than simpler networks when combined with an RNN to predict probability along the time axis. In recent years, technology has become more aware of music, and it is now capable of creating a musical suite without the assistance of a human composer. We face a variety of technological challenges in achieving this goal, which are thoroughly discussed in this study. The article gives a quick overview of music and its elements and cites and evaluates a wide range of pertinent studies conducted by academics in this discipline. In particular, Long Short-Term Memory (LSTM) networks from recurrent neural networks (RNN) are used in this study to provide a

mechanism for producing musical sounds. For the purpose of easy access, this method is built using a model in which data is represented using the MIDI file format.

One key challenge with modeling music is selecting the data representation. 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.

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).

II. RELATED WORK

Allen Huang states previous work in music generation has 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 can generate 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.

III. METHODOLOGY

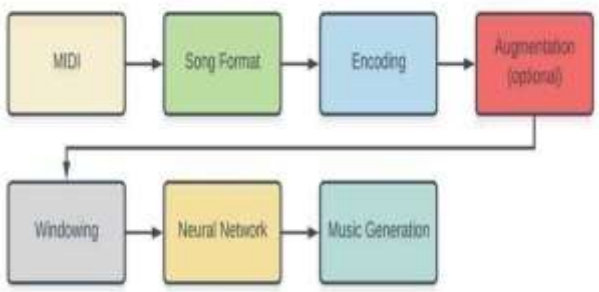

Fig 1. System Architecture

The model is used with polyphonic musical notes. The LSTM network is trained to learn the likelihood that a musical note will occur right now. In order to recall prior information and note structure, the input unit receives the output of the network at a time step t, conditional on the status of the preceding notes up to time step t- 50. A chosen input affects the LSTM layer. Not every note goes through the training procedure. This LSTM model is trained using only a few carefully chosen and specified notes, which can be tuned to produce an effective information gain. These inputs teach the LSTM layer how to map and correlate notes with their projection. The Dropout layer is used to generalize the model in addition to the LSTM layer. All the LSTM cells must be combined once the model has figured out the probabilities associated with notes and sequences. With the aid of the dense layer, this space is subordinated. The model is fully connected because of the dense layer. The decision-making function of the activation layer, which is introduced to the model at the endpoint and aids in determining which neurons (LSTM cells) should be activated and if the information acquired by the neuron is relevant, is crucial in a deep neural network. The LSTM network may now generate new musical note sequences after being trained. A big and diversified dataset was elicited with numerous variations in the structural structure of musical notes in order to assure better prediction and diverse production of sequences. The objective was to expose the model to a variety of datasets so that it could be better tuned. The extracted dataset was in the MIDI file format. In order to extract data about note sequencing, note velocity, and the time component, MIDI files were crucial.

The polyphonic musical notes are used to test the model. The LSTM network is trained to learn the likelihood that a musical note will occur right now. The input unit receives the output of the network at a time step t, conditional on the state of the notes up to time step t–50, in order to recall previous details and note structure. A chosen input affects the LSTM layer. Not every note goes through the training procedure. This LSTM model is trained using only a few carefully chosen and specified notes, which are also essential for fine-tuning the model and achieving effective information gain. These inputs teach the LSTM layer how to map and correlate notes with their projection. The Dropout layer is used to generalize the model in addition to the LSTM layer. All the LSTM cells must be combined

once the model has figured out the probabilities associated with notes and sequences. With the aid of the dense layer, this space is subordinated. The model is fully connected because of the dense layer. The decision-making function of the activation layer, which is added to the model at the endpoint and assists in determining which neurons (LSTM cells) should be activated and if the information acquired by the neuron is relevant, is crucial in a deep neural network. The LSTM network may now generate new musical note sequences after being trained. A big and diversified dataset was elicited with numerous variations in the structural structure of musical notes in order to assure better prediction and diverse production of sequences. The objective was to expose the model to a variety of datasets so that it could be better tuned. The extracted dataset was in the MIDI file format. In order to extract data about note sequencing, note velocity, and the time component, MIDI files were essential. A model was built using Moon et al.'s recommended dropouts as a guide, and the LSTM layer was given a dropout value of 0.75. RMSprop was the optimizer that was chosen. The learning rate selected was 1e-4 for model optimization.

IV. IMPLEMENTATION

A. Dataset

A clean selection of the Lakh MIDI dataset served as the basis for the dataset we used. A collection of 176,581 unique MIDI documents makes up the Lakh MIDI dataset, 45,129 of which have been coordinated with and altered to specific parts in the Million Song Dataset. We only include the songs from this subgroup for which we had the option to obtain type data, as evidenced by their associated appearance in the MSD. The 3827 songs that were available for this sorting option included the Pop Rock, Folk, Country, Electronic, Blues, Latin, Reggae, RnB, Rap, International, Vocal, New Age, and Jazz genres.

B. Import libraries

MIT developed the Python package Music 21 to find out information about music. Music documents are typically stored using the MIDI format. Musical Instrument Digital Interface is what MIDI stands for. MIDI documents just contain the instructions, not the actual sound. It therefore has almost no memory. It is preferred when transporting documents because of this.

C. Reading and Understanding the Data

Examining the data we'll be dealing with is the first stage in running the brain network. Notes and Chords are the two article types that make up the information. Information on the pitch, octave, and offset of the Note is included in Note objects. The term "pitch" refers to how often a sound occurs or how high or low it is. It is denoted by the letters "A," "B," "C," "D," "E," "F," and "G," with A being the most notable and G being the least. The term "octave" alludes to the piano's pitch range. Counterbalance makes a reference to the note's position within the composition.

D. Preparing the Data

The information will initially be stacked into an array. The first thing we do is use the converter to stack each document into a Music21 stream object. file parsing work We can acquire a breakdown of the document's relative abundance of notes and harmonies by using that stream object. Since the key components of the note may be replicated using the pitch's string documentation, we append the pitch of every note object using that documentation. Additionally, we add each harmony by stringing each note's unique identifier into its own string and isolating it from the others with a speck. We can create the groupings that will serve as the organization's contribution because we have arranged all of the notes and harmonies in a sequential rundown. We will first create a capability for planning from string-based all-out information to mathematical information based on whole numbers. This is done on the theory that brain networks operate significantly better when given whole number-based mathematical information than when given unmodified information based on strings. The next step is to set up the organization's input and distinct results. The outcome for each grouping of information will be the main note or harmony that follows the arrangement of notes 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.

E. Model

We finally begin the model engineering planning. Four distinct types of layers—LSTM layers, Dropout layers, Dense layers or totally associated layers, and Activation layers—are used in our model. The primary boundary for each LSTM, Dense, and Activation layer is how many hubs it should have. The main barrier for the Dropout layer is the small percentage of information units that must be dropped during preparation. We must provide a fresh boundary for the principal layer termed input shape. The boundary's purpose is to educate the organization about the quality of the information it will be preparing. The final layer should always have an equivalent number of hubs to the variety of results our framework has. This ensures that the organization's outcome will be simple to apply to our classes.

F. Generating Music

You must put the brain organization in a similar state to earlier in order to have the option of using it to compose music. We can now start creating notes using the predefined model. We will choose an irregular record in the rundown as our starting point because we have access to a complete list of note groupings. This enables us to rerun the age code without making any changes and produce different results in a predictable manner. Here, we also need to have the ability to plan in order to separate the organization's results. This capability will plan using both mathematical and comprehensive facts. Then, at that time, we compile all of the organization's results into a single exhibit. We can start translating them and creating different Note and Chord objects now that we have all of the encoded representations of the notes and harmonies in a cluster. In the case of a chord, we must divide the string into various notes. Then, at that point, we go around and create a Note object for each note represented by a string. After that, we can create a Chord object that contains each of these notes. If the example is a Note, we create a Note object using the string representation of the contribution that contained the example. We may create a Music21 Stream object using the rundown as a boundary as we have a list of Notes and Chords created by the company. Finally, we use the compose function in the Music21 tool cache to compose the stream to a record in order to create the MIDI document that will contain the music created by the organization.

III. CONCLUSIONS

The objective of creating a model that may be used to automatically produce music and melodies without human input is accomplished by this method. The model is competent enough to learn harmonic and melodic note sequence from MIDI files of Pop music, and it can retain the past details of the dataset and produce polyphonic music using a single layered LSTM model.

IV. FUTURE SCOPE

Future research will examine the model's scalability on even bigger datasets, such the Million Song Dataset. We want to examine how adding more LSTM units and experimenting with different combinations of hyperparameters affects this model's performance. We think that more study and extensive computation will allow us to further improve this model.

Although the music we produced is of a respectable caliber, there is still much room for development. First, beginning and closing music can be added to every newly generated tune to improve the beginning and ending of the song. This will make our computer-generated music melodious. Second, more tunes can be used to train the model. Here, we've used just 405 songs to train our algorithm. The model will be exposed to a wider range of music by being trained with more musical pieces, and the number of lessons will rise as well. This allows the model to produce music that is both more melodic and more varied. Third, multi-instrument melodies can be used to train the model. Currently, there is only one instrument used in the music that is produced. If the model is trained on multi-instrument music, it will be interesting to hear the music that it creates. Finally, a mechanism that can handle unidentified musical notes can be incorporated into the model. The model can produce music of higher quality by filtering out unknown notes and substituting recognized notes.

REFERENCES

[1] [1] Lejaren A. Hiller and Leonard M. Isaacson. Experimental Music: Composition with an Electronic Computer. McGraw-Hill, 1959.

[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, 1997, 9(8): 1735-1780.

[4] Mark Steedman. A generative grammar for Jazz chord sequences. Music Perception, 2(1):52–77, 1984.

[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), pages 635–642, Barcelona, Spain, July 2011.

[6] Jer´ ome Nika, Marc Chemillier, and G ˆ erard Assayag, ´ "Improtek: introducing scenarios into human-computer music improvisation," Computers inEntertainment (CIE), vol. 14, no. 2, pp.4, 2016.

[7] David Cope and Melanie J Mayer, Experiments in musical intelligence, vol. 12, 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, pp. 160–183, 2016.

[9] Mason Bretan, GilWeinberg, and Larry Heck, "A unit selection methodology for music generation using deep neural networks," arXiv preprint arXiv:1612.03789, 2016.

[10] 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, pp. 155–159.

[11] Gabriele Medeot, Srikanth Cherla, Katerina Kosta, Matt McVicar, S Abdalla, M Selvi, E Rex, and K Webster, "Structure net: Inducing structure in generated melodies," in The 19th International Society for Music Information Retrieval Conference, 2018.

[12] Hang Chu, Raquel Urtasun, and Sanja Fidler, "Song from pi: A musically plausible network for pop music generation," arXiv preprint arXiv:1611.03477, 2016. [12] Daniel D Johnson, "Generating polyphonic music using tied parallel networks," in International conference on evolutionary and biologically inspired music and art. Springer, 2017, pp. 128–143.

[13] Michael Chan, John Potter, and Emery Schubert, "Improving algorithmic music composition with machine learning," in Proceedings of the 9th International Conference on Music Perception and Cognition, ICMPC, 2006

[14] Sepp Hochreiter and Jurgen Schmidhuber, "Long short- ¨ term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] Kevin Joslyn, Naifan Zhuang, and Kien A Hua, "Deep segment hash learning for music generation," arXiv preprint arXiv:1805.12176, 2018.

[16] Rongshu Sun, Jingiing Zhang, Wei Jiang, and Yuexin Hu, "Segmentation of pop music based on histogram clustering," in 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP- BMEI). IEEE, 2018, pp. 1–5.

[17] Cheng-i Wang, Gautham J Mysore, and Shlomo Dubnov, "Re-visiting the music segmentation problem with crowdsourcing.," in ISMIR, 2017, pp. 738–744.

[18] Gabriel Sargent, Fred´ eric Bimbot, and Emmanuel Vin- ´ cent, "Estimating the structural segmentation of popular music pieces under regularity constraints," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 25, no. 2, pp. 344–358, 2017.

[19] Kevin Joslyn, Kai Li, and Kien A Hua, "Cross-modal retrieval using deep de- correlated subspace ranking hashing," in Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval. ACM, 2018, pp. 55–63.

[20] Sarthak Agarwal, Vaibhav Saxena, Vaibhav Singal, and Swati Aggarwal, "Lstm based music generation with dataset preprocessing and reconstruction techniques," in 2018 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2018, pp. 455–462.

[21] Y Cem Subakan and Paris Smaragdis, "Diagonal rnns in symbolic music modeling," in 2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA). IEEE, 2017, pp. 354–358.

[22] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," arXiv preprint arXiv:1206.6392, 2012.