Music Generation with Bi-Directional Long Short Term Memory Neural Networks

2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT) | 978-1-6654-5262-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCCNT54827.2022.9984228

Madhur Rajadhyaksha
Department of Information Technology
Sardar Patel Institute of Technology
Mumbai, India
madhur.rajadhyaksha@spit.ac.in

Neha Lakhani
Department of Information Technology
Sardar Patel Institute of Technolgy
Mumbai, India
neha.lakhani@spit.ac.in

Mohammad Anas Mudassir
Department of Information Technology
Sardar Patel Institute of Technology
Mumbai, India
mohammadanas.mudassir@spit.ac.in

Dr. Prasenjit Bhavathankar
Department of Information Technology
Sardar Patel Institute of Technology
Mumbai, India
p_bhavathankar@spit.ac.in

***Abstract*— Deep Learning has been utilized in music generation where long term dependencies and patterns need to be learned by the network. Recurrent Neural Network (RNN) units have been used in the form of Generative Adversarial Networks, Auto-Encoders and Long Short-Term Memory Networks but they have been plagued by issues with regard to varying offsets and different sequence lengths. This paper proposes the modification of bi-directional Long Short Term Memory Networks to produce new music as well as generate the succeeding notes in existing compositions.**

Keywords*— *Music generation, Bidirectional Recurrent Neural Networks, Long Short Term Memory, Self-attention, Deep Learning

INTRODUCTION

Music Generation requires an extensive understanding of music theory and experience with different pieces of music. It is categorised into two types: Abstractive Music Generation and Extractive Music Generation [3]. The former includes the use of information present in the input music files to generate a completely new piece of music. The latter includes the generation of output music using an algorithm which extracts musical phrases based on their importance in the input melody. In this paper, both abstractive and extractive music generation has been performed.

Deep learning is an ever-expanding domain which also has applications in the field of music generation. It involves training the model on a large dataset of musical input and using it to produce similar music with slight variations. Each melody consists of different notes, keys, pitches, velocities, values, timbres and many more. This makes handling audio files much more complex than handling images or texts [14]. Hence, it is tricky to convert the music into a format comprehensible to the models.

In this paper, the training is done using the MIDI and Audio Edited for Synchronous Tracks and Organization (MAESTRO) dataset [15]. It consists of audio recordings in MIDI format and has a duration of 172 hours which is significantly more than other datasets having similar music. The model is trained on two Bidirectional Long Short-Term Memory (Bi-LSTM) layers with a self-attention layer between them. It recognises and learns the patterns of the existing music files to generate new pieces which are melodious and of decent quality [1].

This paper works with the generation of homophonic music for which the piano channel of the input MIDI file is converted to a piano roll format and is read over two axes. The vertical axis of the matrix includes the different notes present and the horizontal axis is mapped to individual time units [4]. Each of the notes is considered from a global perspective where the neighbouring notes and harmonics affect the note itself [8].

This paper is organized as follows: Section II describes, in brief, the literature survey; Section III explains the network architecture which includes the deep learning models implemented; the methodology of implementation is discussed in section IV; Section V shows the results of the survey conducted to compare the computer-generated music with human compositions; Section VI finally concludes the paper.

I. LITERATURE SURVEY

Audio data has a sequential nature and multiple channels which makes it familiar with various deep learning models. Mogren et al. have generated polyphonous music using Generative Adversarial Networks (GANs). They take into account two variations, tones and intensities of the generated music. Even though their music is not comparable with real music, the results of its evaluation resemble the scores of real music. However, the model needs to work on playing multiple tones simultaneously [5].

GANs suffer from problems such as vanishing gradient, mode collapse and divergence mismatching [18][19]. To overcome the issue of vanishing gradient, Yang et al weakened the discriminator with two strategies. In every iteration, they updated the discriminator only once as opposed to the generator and conditioner which were updated twice. Secondly, they created the discriminator with two convolutional layers and one fully-connected layer. This gave them better results than traditional GANs [6].

Bretan et al. have used autoencoders for predicting the next four beats in a musical sequence by comparing a variety of these architectures and their objective functions. They allow for 60 pitches and any notes that fall outside of this range are transposed into either the lowest or highest octaves

available [7]. The problem with autoencoders is that they are less suited for a generative model since they replicate the input music and are able to provide very little variation [16]. To overcome this, CNN has been used to learn a better representation of music and a Variational Auto Encoder to generate newer sequences of music by Koh et al. They have used Information Rate as a metric to measure the performance of a network [12].

LSTMs have been implemented by various researchers [2] [9] [11] since they incorporate dependencies naturally across time. Hewahi et al. conducted experiments to evaluate various parameters of their model but their work had one drawback. The LSTM model produced short melodies infinitely in certain cases [2]. Lackner et al. have composed a melody from a given chord sequence by converting the musical data into input and target matrices and applying an LSTM model to it. Their musical survey suggests that 40% of the LSTM composed music was misclassified as human melodies by the listeners [9]. Two separate LSTM models were developed in [11] for training dynamics and tempo separately over a dataset of Chopin's mazurkas. Despite a smaller dataset, the model successfully recognized the real-world dependencies.

Mao et al. have built a Biaxial LSTM model to generate polyphonic music based on a mixture of different composition styles. The style control is achieved using a learned distributed representation instead of one-hot encoding for each musician. They have extracted note features by using 1-dimensional convolution layer which is a unique contribution [10]. Dawande et al. suggest the use of bi-axial LSTM to maintain separate systems for tracking notes and prediction[23]. Lastly, Azmi et al. have implemented a Bi-directional LSTM model in which 3 hidden layers of 512 cells each are stacked on the input layer. Their loss function suggests 15 epochs give optimal output. They generated new music by inputting a single note and the results were significantly improved by the stacked nature of Bi-LSTM [13].

This paper aims to overcome the limitations of the above works by using a bi-directional LSTM model aided by a self-attention layer to perform training on the input music. The process of music generation involves five steps, conversion of MIDI to piano roll representation, conversion of the piano roll to input to and target matrices, training of the Bi-LSTM model, conversion of the output matrix back to piano roll representation and finally, piano roll to MIDI format.

II. Network Architecture

In the proposed system, music is generated via a predictive model, i.e, the pitch occurring at a particular timestamp is determined by the sequence of preceding pitches. A recurrent neural network-based model is trained on the MAESTRO dataset [20]. As mentioned in the methodology section, the architecture of the suggested model is based on two bidirectional LSTM layers functioning as an encoder-decoder system with a self-attention layer between them to effectively utilize long sequences from the first bi-directional LSTM and formulate a weighted input for the second one.

A. Bidirectional LSTM

RNNs maintain a memory based on historic sequential data, where the output is determined by long-distance attributes of the incoming sequence. Thus, it is ideal to employ RNN for predictive music generation. LSTM introduces sequence dependence by predicting the output on the basis of two parameters, the input at the current timestamp and the hidden state, which is determined by the state of an LSTM cell at the previous timestep. But in the case of music generation, humans generally compose a melody by placing notes in accordance with the preceding as well as following pitches [8]. This approach to composition renders a simple LSTM based architecture inadequate, and hence our proposed system implements bi-directional LSTM.

Bi-directional RNNs are a method of stacking two LSTM networks into a single layer, where the first RNN processes data in the forward direction while the second RNN processes data in the backward direction, as shown below in figure 1. Thus, the predictive analysis of the placement of each note in a melody can be formalized as a function of the pitches that precede and follow the pitch at the current timestamp. Both the forward and backward layers generate a probability distribution as an output, which is later passed into an activation function to obtain a generalized output.

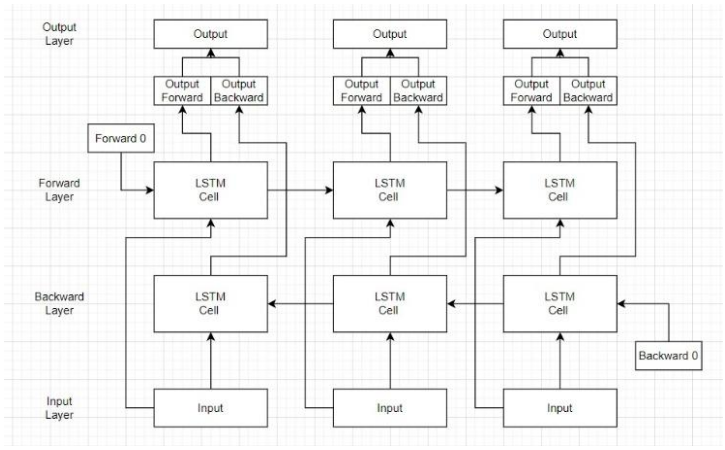

Fig. 1. Representation of Bi-Directional LSTM

Bidirectional LSTM layers are stacked in the proposed system forming a model where the output of one layer serves as the input for another, which significantly increases the performance of this architecture by providing a cascading cumulative effect while processing the data [14].

B. Self Attention Model

In a basic LSTM network, the output is determined by the input at the particular timestamp along with the hidden state of the LSTM cell. This approach is insufficient for generating harmonious music as composers generally refer to entire sequences of pitches for determining the appropriate note occurring at each time step. To integrate this approach, attention-based architecture is used. An attention layer is introduced between two LSTM layers, where the weighted sum of the output of the first layer acts as the input for the second LSTM layer [24]. The weight assigned to each feature determines the importance of that particular input with respect to the output. The following equation determines the value of the attention vector

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (1)$$

where Q is the set of queries, K is the key, V is the associated value and d_k represents the dimensions of the key.

To evaluate the value of each input feature, the output and corresponding sequential input function together across multiple timesteps. This is represented in figure 2 below.

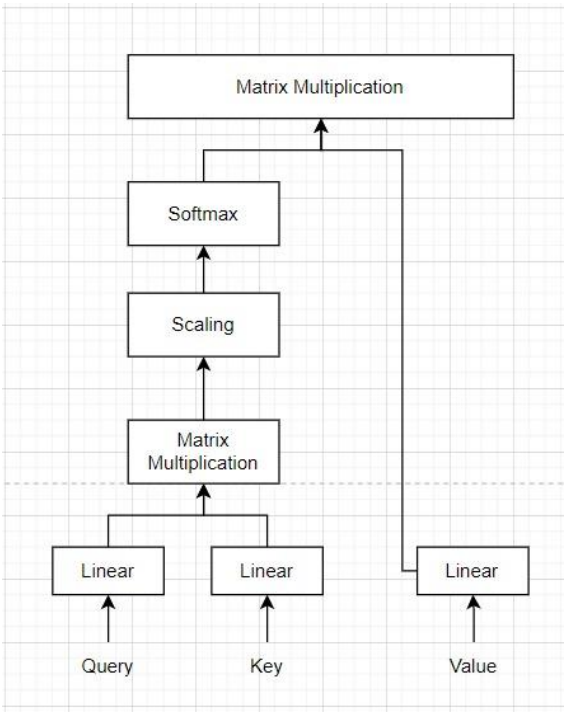

Fig. 2. Representation of Self Attention Model

Although in basic attention models, establishing the relevance of each contextual feature with respect to the output can become a challenging task [21]. To overcome this, the self-attention mechanism is utilized, where interaction between input embeddings becomes the determining factor for assigning importance to each feature in the network. The following process is iterated over to train a self-attention network:

1. The embeddings of the input vector are multiplied with an initialized weight vector to generate query, key and value vectors. The query refers to the current input at the particular timestep, the key acts as the input vectors situated in the proximity of the query, while the value is used as a multiplier to determine the influence of each individual input on one another
2. The query set is multiplied with the transpose of the key vector which produces the similarity matrix of the current input with other input features
3. As these values can approach extremely large values, they need to be scaled in accordance with the size of the input embedding vector before normalization. Thus, each value in the similarity index is divided by the square root of the number of dimensions of this matrix.
4. A softmax function is applied to the similarity matrix to obtain the self-attention scores corresponding to each input
5. Matrix multiplication is performed on the matrix thus obtained with the value matrix that was calculated in the first step

Alongside improving the overall accuracy of the system, incorporating a self-attention model significantly reduces the time complexity of the algorithm [22].

III. Methodology

The execution of the proposed work can be divided into 4 broad categories: pre-processing, training, post-processing, and output.

A. *Pre-processing*

In this stage, the music files are loaded into the program and are passed to the 'pretty_midi' library. Firstly, the file name is passed as the parameter to load the song in the variable followed by extracting the data of the first channel, i.e., the piano instrumental of the song. This is followed by specifying the *sampling frequency* for the song. The sampling frequency is the interval at which the library extracts the information about the notes being played at that instant. Thus, if the sampling frequency is 5 seconds and there exists a song having a duration of 5 minutes, i.e, 300 seconds, then the resultant piano roll will consist of notes being played at 60 (300/5) instances separated by 5 seconds each. Each column vector of the piano roll represents the notes played at varying intensities for the given timestamp

Following this, NumPy is used to discard the timestamps where zero notes are played. Keep in mind that each cell having a value greater than 0 represents a note being played denoted by the row number while the cell value represents the velocity/intensity of the note. Thus, the remaining timestamps are iterated and the notes being played at that instant are appended together in the form of a comma-separated string. For example, consider that for the 3rd timestamp the notes 23, 45, and 56 were played with the velocity 3, 5, and 2 respectively. Then, the dictionary will have an entry that looks like this: '3': '23,34,56' where the key represents the timestamp and the value represents the notes played respectively.

Moving to the final stage of preprocessing, one of the tasks is the representation of the notes in the form of integers which would make training easier. This is done by keeping two dictionaries, called index_to_note and note_to_index. In the first one, each integer index represents a unique string note and in the second dictionary the mapping is reversed, i.e., the note represents an integer value (index). The input matrix for a particular song is then generated by specifying the sequence length followed by iterating over the dictionary which specified the timestamp to note relation. If the timestamp is smaller than the sequence length, then the input vector is padded with dummy values equal to the difference between sequence length and timestamp. The remaining values are then occupied by the value obtained from note_to_index. Finally, the target matrix for a particular input vector is the next input vector itself as our network aims to generate the next note.

B. *Training*

The dataset used to train the network consists of 200 piano songs from the MAESTRO dataset provided by Magenta [17]. The training step consists of the architecture specified earlier in the form of a deep learning network using bi-lstm and self-attention layers. The length of the initial embedding layer is equal to the length of the input matrix. In addition, the length of the output layer is equal to the total number of the unique combinations of notes present in the dataset. The training consists of 4 epochs with each batch containing 16 songs and the sequence length for the input matrix being 50, with the Adam optimizer combined with Nesterov momentum (known as NAdam) being the optimizing function for the network. Since the softmax layer is used as output and the target matrix consists of integers having a wide range, the loss function utilized is sparse categorical cross-entropy as it saves us the trouble of one-hot encoding. The parameters are summarized in the table 1 below:

TABLE 1. PARAMETERS USED IN THIS WORK

Parameter	Value
Epochs	4
Batch Size	16
Sequence Length	50
Attention Regularizer Weight	0.00001
Frame per second	5
Optimizer	NAdam
Graphical Processing Unit (GPU)	NVIDIA K80 (12 GB)

C. *Post-processing*

Output in the softmax layer comprises of each neuron having a value between zero and one, and the sum of all neurons' values adding up to 1. Thus, the output notes after each prediction are derived by using the roulette wheel selection method which means the probability of a note being selected would be higher for the neuron which has a greater value. The indices from the output layer are then used as the key to the index_to_notes dictionary for obtaining the predicted note for a given input sequence. Finally, the output is converted into an input vector and fed to the network which forms a feedback loop for generating music.

D. *Output*

The initial input is a vector of sequence length 50 and the model is executed 300 times to generate the output. After each output note is obtained via roulette wheel selection, it is appended to the input vector and the new iteration then selects the last 50 notes as the input. Thus, this ensures that the output of the previous iteration becomes the input vector for the current iteration as illustrated in the figure 3 where the note with output 179 becomes part of the sequence considered as input.

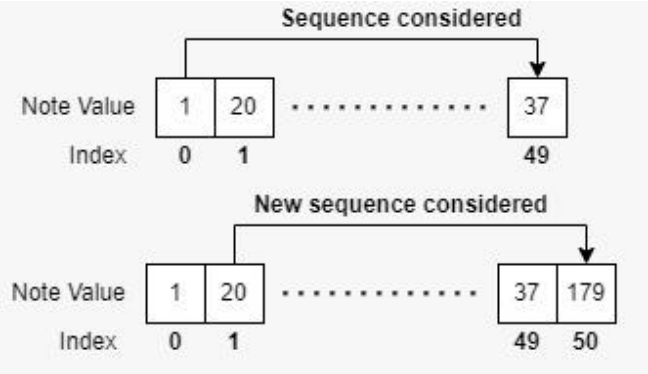

Fig. 3. Input and Output addressed using single vector

At the end of the for loop, the original input vector is now a large vector containing the sequence of notes predicted by the model. This is then processed by iterating over the vector and converting the index value to the note string using the index_to_notes dictionary. The piano roll is initialized as a matrix consisting of a range of notes as the number of rows and the number of times the model was executed as the columns with values as 0 for each cell. The notes denoted by a comma-separated string obtained from the index_to_notes dictionary are split into individual notes and the corresponding cell for the note in the piano roll is set to 1. Finally, using the pretty_midi library, the piano roll is converted to midi file with the velocity of all notes set to 100.

IV. Results And Evaluation

To judge the performance of the network, an online survey was conducted via Google Forms where 30 participants were asked to listen to two audio files and were subsequently asked two questions:

1. Which one of the two compositions did you like better?
2. Which one of them did you think was computer generated?

The results have been presented in the graphs below as figure 4 and figure 5. 'Audio 1' is human generated music and 'Audio 2' is computer generated music by our program.

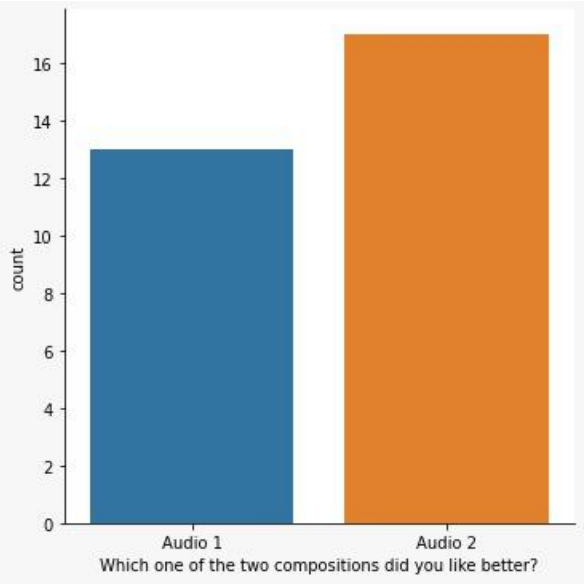

Fig. 4. Results show 'Audio 2' is preferred

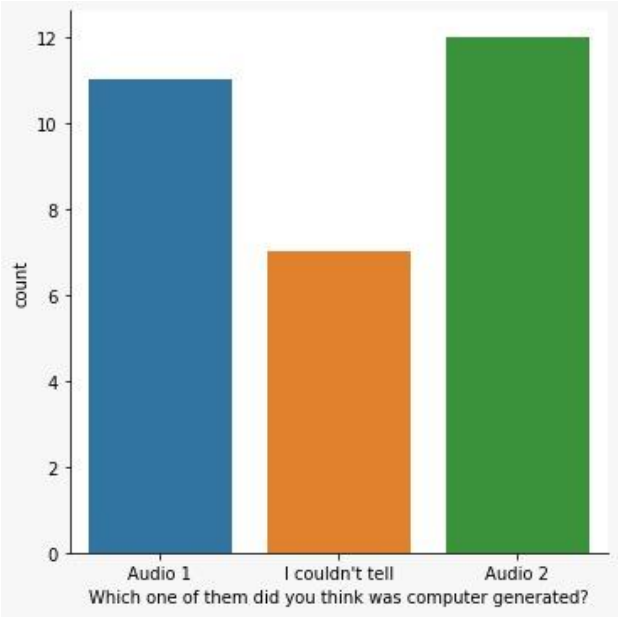

Fig. 5. Results show 'Audio 2' was identified as machine generated and yet it was preferred by the survey participants as seen in figure 4.

The results showed that the significant number of users preferred the audio generated by our program and couldn't figure out that it was generated by a machine, which is a strong indicator of how well the network performed.

V. Conclusion

This work has analyzed the musical structure of piano compositions and argues for the effectiveness of bi-directional LSTM networks combined with the self-attention model for a generation of music. The music files used for training were in MIDI format which provided better output than traditional .mp3 music files used for training. Further, the proposed model is able to overcome the drawbacks of the other architectures such as GANs and Auto-Encoders by not being susceptible to mode-collapse or generating overly repetitive music respectively. However, one of the challenges in this domain is the lack of quantifiable metrics, which is why we have utilized a survey for judging the quality of the results. Subsequently, it was found that more than half of the participants had chosen the output generated by our program as the better music. In addition, this program was able to execute within the constraints of limited GPU memory available while still delivering improved performance.

References

[1] Eva P.S. Rani, S.V. Praneeth, V.R.K. Reddy, M.J. Sathish, "MUSIC GENERATION USING DEEP LEARNING", International Research Journal of Engineering and Technology, vol. 7, no. 3, 2020.

[2] N. Hewahi, S. AlSaigal, and S. AlJanahi, "Generation of music pieces using machine learning: long short-term memory neural networks approach," Arab j. basic appl. sci., vol. 26, no. 1, pp. 397–413, 2019.

[3] A. Pal, S. Saha, R. Anita, "Musenet : Music Generation using Abstractive and Generative Methods", International Journal of Innovative Technology and Exploring Engineering, vol. 9, no. 6, pp. 784-788, 2020.

[4] Y. Huang, X. Huang, Q. Cai, "Music Generation Based on Convolution-LSTM", Computer and Information Science, vol. 11, no. 3, pp. 50-56, 2018.

[5] O. Mogren, "C-RNN-GAN: Continuous recurrent neural networks with adversarial training," arXiv [cs.AI], 2016.

[6] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," arXiv [cs.SD], 2017

[7] [7] M. Bretan, S. Oore, D. Eck, and L. Heck, "Learning and evaluating musical features with deep autoencoders," arXiv [cs.SD], 2017.

[8] T. Jiang, Q. Xiao and X. Yin, "Music Generation Using Bidirectional Recurrent Network," 2019 IEEE 2nd International Conference on Electronics Technology (ICET), 2019, pp. 564-569, doi: 10.1109/ELTECH.2019.8839399.

[9] K. Lackner, "Composing a melody with long-short term memory (LSTM) Recurrent Neural Networks," 2016.

[10] H. H. Mao, T. Shin and G. Cottrell, "DeepJ: Style-Specific Music Generation," 2018 IEEE 12th International Conference on Semantic Computing (ICSC), 2018, pp. 377-382, doi: 10.1109/ICSC.2018.00077.

[11] M. K. Jędrzejewska, A. Zjawiński and B. Stasiak, "Generating Musical Expression of MIDI Music with LSTM Neural Network," 2018 11th International Conference on Human System Interaction (HSI), 2018, pp. 132-138, doi: 10.1109/HSI.2018.8431033.

[12] E. S. Koh, S. Dubnov and D. Wright, "Rethinking Recurrent Latent Variable Model for Music Composition," 2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP), 2018, pp. 1-6, doi: 10.1109/MMSP.2018.8547061.

[13] Syeda Sarah Azmi, Shreekara C, Shwetha Baliga, "Music generation using Bidirectional Recurrent Neural Nets," International Research Journal of Engineering and Technology, Volume: 07 Issue: 05, May 2020, e-ISSN: 2395-0056, p-ISSN: 2395-0072.

[14] Qi Lyu, Zhiyong Wu, Jun Zhu, and Helen Meng. 2015. "Modelling high-dimensional sequences with LSTM-RTRBM: application to polyphonic music generation" 24th International Conference on Artificial Intelligence (IJCAI'15). AAAI Press, 4138–4139.

[15] F. Shah, T. Naik and N. Vyas, "LSTM Based Music Generation," 2019 International Conference on Machine Learning and Data Engineering (iCMLDE), 2019, pp. 48-53, doi: 10.1109/iCMLDE49015.2019.00020.

[16] Ivan P. Yamshchikov, Alexey Tikhonov, "Music generation with variational recurrent autoencoder supported by history", SN Appl. Sci. 2, 1937 (2020). doi: https://doi.org/10.1007/s42452-020-03715-w

[17] Mateusz Modrzejewski, Mateusz Dorobek & Przemysław Rokita, "Application of Deep Neural Networks to Music Composition Based on MIDI Datasets and Graphical Representation,"Artificial Intelligence and Soft Computing, 2019, Volume 11508, ISBN : 978-3-030-20911-7

[18] Zhang, Z., Luo, C., and Yu, J. Towards the gradient vanishing, divergence mismatching and mode collapse of generative adversarial nets. In Proceedings of the 28th ACM International Conference on

Information and Knowledge Management (New York, NY, USA, 2019), CIKM '19, Association for Computing Machinery, p. 2377–2380.

[19] Martin Arjovsky, Soumith Chintala, and Leon Bottou. ´ Wasserstein GAN. arXiv preprint arXiv:1701.07875, 2017.

[20] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, Douglas Eck, "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset," 2019 International Conference on Learning Representations(ICLR), 2019, arXiv:1810.12247 [cs.SD]

[21] Xiaowen Huang, Shengsheng Qian, Quan Fang, Jitao Sang, and Changsheng Xu. 2018. CSAN: "Contextual Self-Attention Network for User Sequential Recommendation,"In Proceedings of the 26th ACM international conference on Multimedia (MM '18). Association for Computing Machinery, New York, NY, USA, 447–455. DOI:https://doi.org/10.1145/3240508.3240609

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention is all you need," In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010

[23] Akanksha Dawande , Uday Chourasia , Priyanka Dixit, "Music Generation and Composition Using Machine Learning", International Journal Of Engineering Research & Technology (IJERT) Volume 10, Issue 12 (December 2021)

[24] Soydaner, D. "Attention mechanism in neural networks: where it comes and where it goes". Neural Comput & Applic 34, 13371–13385 (2022). https://doi.org/10.1007/s00521-022-07366-3