2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)

2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI) | 978-1-6654-3881-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/CEI52496.2021.9574491

Generating Music with Generative Adversarial Networks and Long Short-Term Memory

Yuhan Dai[1, *, †]
[1]College of Letters and Science, Economics Department
University of California, Davis
Davis, United States
[*]Corresponding author's e-mail:
dyhdai@ucdavis.edu

Tong Xin[2, *, †]
[2]School of Computer Engineering and Science
Shanghai University
Shanghai, China
[*]Corresponding author's e-mail:
xintong18122858@shu.edu.cn

Yihao Zheng[3, *, †]
[3]College of Letters and Science, Statistics Department
University of California, Davis
Davis, United States
[*]Corresponding author's e-mail:
yhzheng@ucdaivs.edu
[†]These authors contributed equally.

***Abstract*—Music exerts an essential role in human beings' daily life. However, there are many restrictions in the field of traditional music creation. In recent years, with the development of artificial intelligence accelerated significantly, how to apply deep learning to music creation is an intriguing topic, and it is worth a wide range of exploration and discussions. In the paper, a deep learning model composed of Generative Adversarial Networks (GANs) and Long Short-Term Memory (LSTM) was introduced to implement the generation of music. The GANs are mainly composed of a generator and a discriminator, while LSTM serves as the basic unit of the generator and discriminator. Through analyzing the music file generated in different iteration times, the results demonstrate that the pieces of music we got via the model made up of GAN and LSTM are in a similar pattern.**

Keywords-deep learning; music generation; Generative Adversarial Networks; Long Short-Term Memory

I. INTRODUCTION

Music cannot only bring people aural enjoyment, but also express people's emotions. In the process of human history and social development, music plays an important role. However, due to the long cycle, high cost and complicated process, the traditional manual creation method has been unable to meet the society's increasing demand for music, and the use of computers for music generation will become a popular composition trend in the future. Computer music generation refers to the use of a computer as an aid to generate a specific note sequence, which contains musical elements such as pitch, rhythm, and chord, and the generated music sequence can be played through software to assist the composer in music creation so that in the process of music creation, the degree of involvement of the composer is minimal.

In the early development of artificial intelligence technology, scholars began to study how to apply artificial intelligence technology to music generation. At that time, there were mainly two ways to generate smart music: (1) Music creation based on Markov chain and other models in statistical analysis, Hiller Jr L A [1] used the mainframe computer Illiac to create a string quartet suite, which became the first computer-generated musical composition in history. (2) Simple pattern matching and machine learning based on music theory rules. For example, Chan M [2] integrates music theory into machine learning to generate notes.

In recent years, with the continuous and in-depth development of deep learning, deep neural networks for music generation have become a very important research direction. Chen [3] takes into account the timing of notes for the first time and uses cyclic neural networks to generate data. Kang S [4] disassembles music elements into various modes and concentration coefficients, and uses backpropagation algorithms to train neural networks to fit the music that meets user tastes. Huang A [5] further proposes an end-to-end deep neural network music generation method.

As Artificial intelligence continues to evolve and make breakthroughs, generating music automatically and intelligently is a field that has been developed further. In recent years, Generative Adversarial Networks (GANs) have become one of the hottest research fields among scholars because of their outstanding performance in data generation. Thus, it is sensible to come up with the idea of how to use GAN for music creation, which is intriguing and deserves attention.

The basic idea of GANs is derived from the zero-sum game, so generally GANs are composed of a generator and a discriminator. The generator is supposed to generate synthesized data from a given noise, whereas the discriminator tries to distinguish the output of the generator from the real ones. Thus, the two networks make progress in the confrontation. The data obtained by the generator will be more and more perfect and close to the real data, so that the desired pictures, sequences or even videos can be generated.

Inspired by the image generation [6], continuous Recurrent Neural Network with Generative Adversarial Network (C-RNN-GAN) [7] was put forward to realize the generation of music. C-RNN-GAN was trained by using the standard GAN loss function, and tone lengths, note frequencies, intensifies, and timing were constructed to characterize the music. Derived by WaveNet, Midinet [8] tried to use Convolutional Neural Network (CNN) to generate music and add confrontation training mechanisms into the training system. Instead of using pure continuous time series to generate melodies, Midinet generates the melodies one by one with a single bar as a unit. The author found that the melodies generated by this model were

978-0-7381-4649-2/21/$31.00 ©2021 IEEE

more realistic than those generated by the conventional Recurrent Neural Network (RNN) model. However, since Midinet failed to integrate the beat and strength of music into training, it had its own drawbacks. The authors of GanSynth [9] suggested that, unlike image data, music is periodic. Thus, how to retain the regularity of periodic signals is essential to the quality of the resulting results. Therefore, the author proposes a model for adversary training processing of music data in the spectral domain. Compared with the autoregressive model that generates audio by sequence, the music generation is accelerated to a large extent, the speed of which is approximately 50,000 times faster.

The goal of this project is to explore a new method to create original music. Unlike the traditional music creation process, based on our expectation of the project, we should be able to skip the professional music creation process and get the original audio sound track in just a few seconds. With the help of GAN, a normal person like you and me who don't have any previous experience in music is able to generate unique music through a simple click. Moreover, we would like to make some innovations by joining Long Short-term Memory (LSTM) into our GANs model. LSTM is an artificial cyclic neural network (RNN) structure for deep learning. Since LSTM has feedback connections, it is capable of processing individual data points, as well as entire data sequences. Typically, an LSTM cell consists of four elements, a cell, an input gate, an output gate, and a forgetting gate. Cells are responsible for remembering the values at arbitrary intervals, and these three gates are used to regulate the flow of information in and out of the cells. Currently, LSTM networks are ideal for a wide range of fields, such as classification, processing, and forecasting based on time series data [10].

II. METHODS

A. Data preparation and pre-processing

The music samples used in the project are downloaded from Kaggle. It contains 10 different genres and each genre has 100 pieces of music. The length of one single piece of music is 30 seconds. These genres are blues, classical, country, disco, hip hop, jazz, metal, pop, reggae and rock and all of them are wav files. Here, the sample rate of the music in our dataset is 44.1 kHz.

With these pieces of music, the dataset can be generated by splitting the music into small segments of n samples. The parameter n denotes the moves taken to split each song. More specifically, the original music will be divided into smaller sequences whose length is n, such as [1, 2, ..., n].

Then we can divide it into training sets and testing sets. In this paper, the ratio of training sets and testing sets is 6:1.

Furthermore, with the view to getting an ideal result, before being put into the GANs, these data must be normalized in advance. Since wav files are encoded with 16-bit signed values, then their range is from -32,768 to +32,768. After normalizing the data, the segments we get are with a range from -1 to +1. Thus, what the generator produces is also between -1 and +1 that afterwards can be denormalized by multiplying 32,768 in order to produce an ultimate Wav file, a playable one.

B. Generating Music based on GANs and LSTM

The deep learning model used in this paper is mainly GAN, which is composed of an Encoder, a Generator and a Discriminator, as shown in **Figure 1**, and the Generator and Discriminator are mainly composed of LSTM.

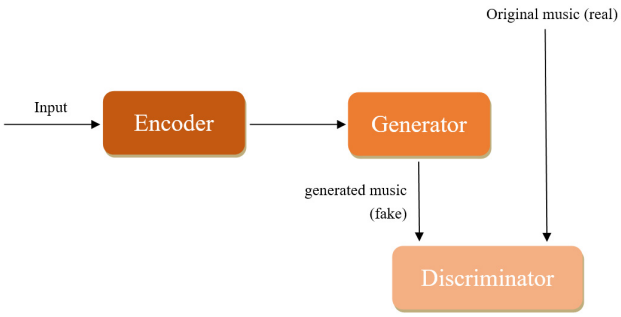


Figure 1 The GAN structure used for music generation.

Though we had preprocessed the data, they could not fit in the LSTMs, because the auto encoder received an input vector of constant size. Thus, the data are compressed with a series of convolution layers and max-pooling layers. More specifically, the function of the decoder is to train the encoder network, then only the encoder can be shown as planned.

The GAN mainly follows the game confrontation theory, and the generator and the discriminator are balanced through a zero-sum game. In other words, the generator constantly learns from the training data to generate new samples, while the discriminator tries to distinguish whether the data is provided by the generator or from the original training set.

In the GANs we designed, both the discriminator and generator are composed of two LSTM layers with one layer stacked on the top of another layer, which is suitable for retention of time-dependent patterns, especially when learning patterns cover a relatively long time.

The generator has three layers in total, two LSTMs and one dense layer. Here, the dense layer was used to synthesize the features extracted from the previous layer. Each node in the dense layer is connected to all the nodes in the previous layer, so it is also named as a fully connected layer. The function of the generator is to generate m sequences whose length is n. These sequences with noises, m sequences got from the original training set, are then mixed, and sent to the discriminator.

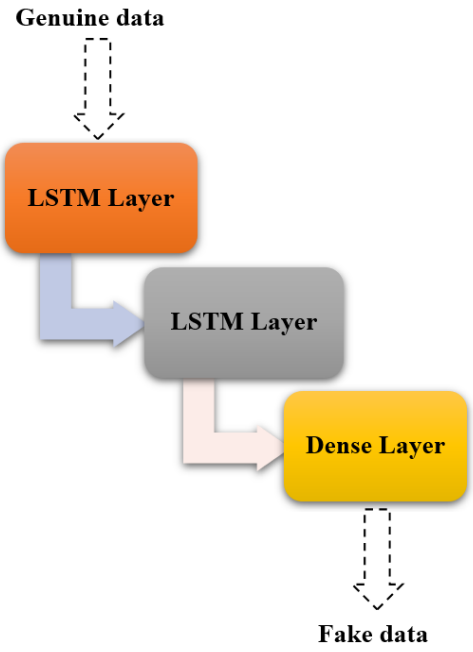


Figure 2 Structure of generator.

Similarly, the discriminator has two LSTMs layers and one dense layer. Then the discriminator predicts whether a piece of music is a real one or a fake one. Based on the results, the discriminator and the generator can update accordingly until the discriminator can no longer tell the fake one from the real one.

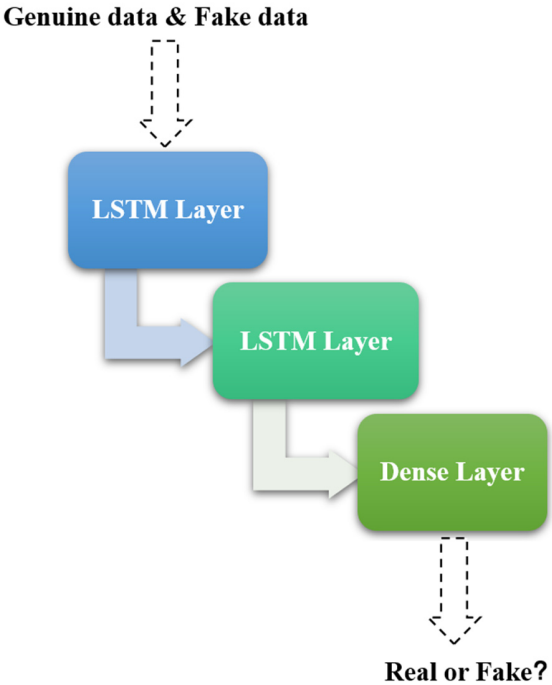


Figure 3 Structure of discriminator.

The reason why LSTMs were embedded in our GANs was that the time dependent patterns could be kept through it, and the structure of the LSTM can be summarized as in **Figure 4**.

Figure 4 Structure of LSTM.

Overall, each network is in a dense feed-forward layer that determines the generated sample's ultimate value. And the structure of the generator and discriminator can be presented in **Figure 2** and **Figure 3,** respectively.

Since the whole process is a zero-sum game, loss functions should be chosen wisely to get optimal results. For the discriminator and generator, two different loss functions were applied. MSE is used as the loss function of the generator, while binary cross-entropy is utilized in the discriminator.

MSE stands for mean squared error, which is a measure of the quality of an estimator that calculates the average squared difference between the generated values and the real ones. Thus, the loss function is demonstrated in formula (1).

$$L = \frac{1}{N}\sum_{j=1}^{N}\|x_j - x'_j\|_2^2 \quad (1)$$

Here, N is the number of samples, x_j and x'_j are the value of real one and predicted one respectively, $\|x_j - x'_j\|_2^2$ is the Euclidean distance of x_j and x'_j.

Unlike MSE, the binary cross entropy loss function returns the loss of an example by computing the following average, as is shown in formula (2).

$$L = -\frac{1}{N}\sum_{j=1}^{N} x_j \cdot \log x'_j + (1-x_j)\cdot \log(1-x'_j) \quad (2)$$

Here, the scalar value in the output of the model is denoted as x'_j, while x_j is represented as the target value. The parameter, N, is the total number of scalar values.

III. Results and Discussions

A. Experimental setup

The python environment used to develop the program is Google Colaboratory (Colab). Colab is developed by Google's research team. In Colab, developers can simply code and execute python-based programs through the web browser. Colab also provides users with free access to computing resources such as GPUs and TPUs. The python packages used in this application are Scipy and Keras. They were acting as audio processors and model trainers respectively.

B. Results

The configuration of GAN model is shown in **Table 1** and the structure of the signal is shown in **Table 2**.

TABLE 1 CONFIGURATION OF GAN MODEL

Generator				
Number of Epoch	Valid Label Shape	Fake Label Shape	True Shape	Fake Shape
5	(3, 1)	(3, 1)	(3, 6, 1)	(3, 6, 1)

TABLE 2 STRUCTURE OF SIGNAL

Sample Rate	Length of Signal	Signal Data Type
22050	44100	PCM16bit

In **Table 3**, the loss, accuracy, validation loss, and validation accuracy of generated music investigated with six different epochs are shown. We got constant accuracy and validation accuracy. Also, the loss and validation loss showed in the table are all small. Besides, the loss keeps decreasing after each epoch by definition and the validation loss is always smaller than the training loss, which means that there is some under-fitting of the model.

TABLE 3 LOSS AND ACCURACY OF LSTM-GAN MODEL

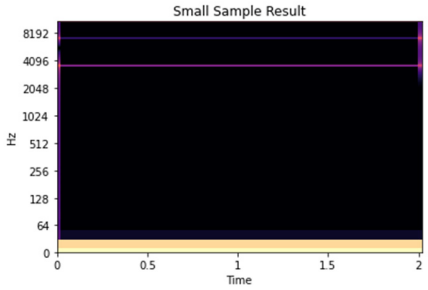
Number of Epoch	Loss	Accuracy	Validation Loss	Validation Accuracy
5	1.177	9.007	9.406	7.493
6	1.173	9.007	9.506	7.493
7	1.167	9.007	9.879	7.493
8	1.168	9.007	1.002	7.493
40	1.144	9.007	9.786	7.493
50	1.141	9.007	9.471	7.493

Then, we describable the results in Table 4, which shows the data of the structure of LSTM model. The present study confirms the method and the structure of the LSTM model that there are 3 layers including 2 LSTM layer and 1 dense layer. In addition, according to the summary of the model, all the parameters are trainable, which means there is no non-trainable parameter. This is an important finding in the understanding of the effectiveness of the model.

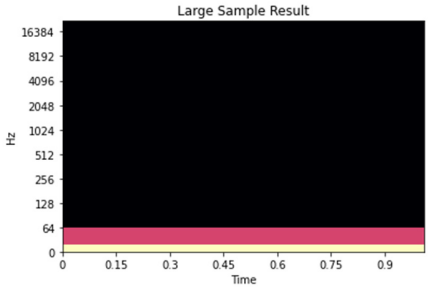
TABLE 4 STRUCTURE OF LSTM MODEL

Layer Type	LSTM	LSTM	Dense
Output Shape	(None, 6, 128)	(None, 128)	(None, 1)
Parametric Number	66560	131584	129
Total Parametric Number	198273		
Trainable Parametric Number	198273		
Non-trainable Parametric Number	0		

The comparison of the waveform and spectrograms of the generated audio files between investigating with 10 original blue music samples and 100 original blue music samples is shown in **Figure 5**. In the first column, the small sample result is with epoch 2 and sample size 10. The second column is the result of 100 sample sizes, and the results are shown at epochs 50. As the figure shown, the results given by small sample sizes are better than the large sample sizes.



(a)



(b)

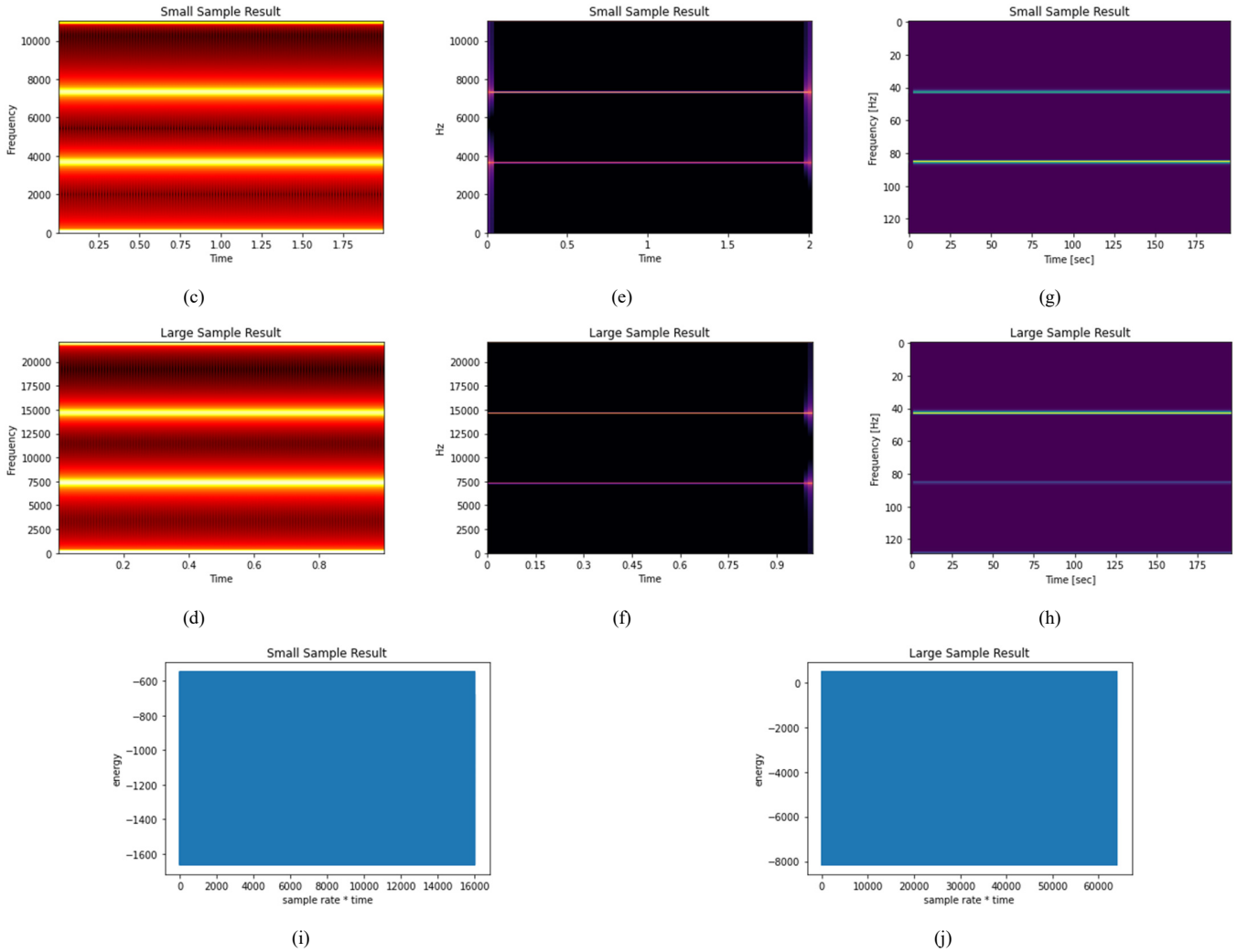


Figure 5 Waveform and spectrograms of generated music trained for small and large sample size respectively.
(a)(c)(e)(g)(i): Trained for 10 music clips with 2 epochs
(b)(d)(f)(h)(j): Trained for 100 music clips with 50 epochs

At the very first time of the implementation, we were faced with sorts of barriers. The wave file we saved showed that the length of the music was 0, and we fixed it by using another way to save the file. Then, the music we generated was silent. After many trials, we found that the output sequence is either ascending order or descending order. We improved the structure of the GAN model and fixed some mistakes that appeared in the function train GAN. Also, we found that increasing the epoch number and the sample size can lead to a better text result.

C. *Discussion*

In this experiment, tons of attempts in different directions have been done to obtain more ideal results, including increasing the music sample size, increasing the number of epochs of our training process, changing the frequency of the generated outputs when converting the output sequences to output .wav files and executing the program with graphics processing unit (GPU) provided by Colab instead of CPU on our personal computer. Consequently, the output obtained from the research is abundant, and profound finds and conclusions had been drawn from comparing and analyzing the results. Although they did not fully satisfy our project expectations, the conclusions and experiences gained from the research still pave the road for the team's further research. The following passages shift focus to the discussion about the inspiration that came with the multiple attempts.

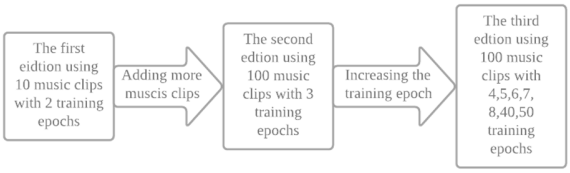


Figure 6 Workflow

Dataset size and the number of the training epoch are two key elements that played a decisive role in the success or failure

of GAN training. Hence, the basic outline of the code upgrading was also based on this concept (**Figure 6**Figure **6** Workflow). The first attempt had been tried was increasing the sample size dramatically from 10 music clips to 100 music clips. With the boost of sample size, normal personal computers were instantly running out of computing power for such a massive calculation, and it was when Colab debut. After running each training epoch with the help of Colab, the application returns a summary for the training process, like the accuracy, loss, val_loss (validation loss), val_accuracy (validation accuracy) (**Table 5**) and finally the audio .wav file. From **Table 5**, it could be easily noticed that after upgrading to 100 music clips, both the accuracy and val_accuracy have increased gigantically, which means increasing the sample size enhances the overall accuracy. However, the accuracy in this case is still too low to get an authentic simulated audio file. Not surprisingly, the audio file's sound was a high pitch sound with no clear pattern. In this case, the problem could be a lack of training epochs. Since by looking closer to the sequences generated by the GAN, the range and style of the sequences are very close to the sequences converted from the sample data. This means the GAN is probably still functioning its job; however, the lack of training epochs makes it hard to capture the features imbedded in the sample clips.

TABLE 5 THE RESULT OF THE FIRST EDITING TOWARDS THE APPLICATION

All the trails have 3 training epochs		
	10 music clips	*100 music clips*
Accuracy	1.7993e-04	9.0071e-04
Loss	2.7026e-05	1.1901e-04
Val_accuracy	5.4875e-04	7.4928e-04
Val_loss	2.4748e-05	1.0110e-04

Note: the accuracy, loss, val_accuracy, and val_loss here are the return after running the final training epoch.

With the same dataset, the training epoch had been increased from 3 epochs to 4, 5, 6, 7, 8, 40 and 50 epochs (**Table 3**). Even though with the help of the Colab, the running time that each training epoch consumes remained enormous, 1300 to 1500 seconds per epoch (**Table 3**). From **Table 3**, it is obvious to see that the loss and val_loss constantly decrease with the increase of the number of training epochs. On the other hand, the accuracy and val_accuracy remained steady from time to time. This is a very confusing part of this application that the accuracy keeps unchanged while loss steadily drops. Meanwhile, all the output .wav files were corrupted and cannot be played. A lot of effort had been spent to figure out what caused the .wav file to corrupt and had tried to change the frequency at which the generated sequences were converted to .wav files, but there is no observable change.

D. Potential reasons and further improvements

With the increase of training times, the generated WAV files are broken, all of these clues point in one direction: there may be some problems with the GAN model architecture. Future research may combine new ideas, such as Musical Instrument Digital Interface (MIDI), to train the new GAN model. MIDI is a technical standard that describes a communications protocol, digital interface, and electrical connectors that connect a wide variety of electronic musical instruments, computers, and related audio devices for playing, editing, and recording music [11].

IV. CONCLUSION

In the paper, we tried to use GANs and LSTM to generate music. LSTM is the basic unit of the generator and discriminator in GANs. Based on different training times, we implemented many experiments, and the results turned out that with the number of iterations increased, there was an increment in the accuracy. Eventually, we could get some short pieces of music, which verified the adaptability of our work. However, most of the music generated by the model was far from satisfactory, indicating that our word still had flaws. Thus, a comprehensive analysis was made, and we will keep optimizing it in the near future.

REFERENCES

[1] Hiller Jr L A, Isaacson L M. Musical composition with a high speed digital computer[C]//Audio Engineering Society Convention 9. Audio Engineering Society, 1957.

[2] Chan M, Potter J, Schubert E. Improving algorithmic music composition with machine learning[C]//Proceedings of the 9th International Conference on Music Perception and Cognition, ICMPC. 2006.

[3] Chen C C J, Miikkulainen R. Creating melodies with evolving recurrent neural networks[C]//IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222). IEEE, 2001, 3: 2241-2246.

[4] Kang S, Ok S Y, Kang Y M. Automatic music generation and machine learning based evaluation[C]//International Conference on Multimedia and Signal Processing. Springer, Berlin, Heidelberg, 2012: 436-443.

[5] Huang A, Wu R. Deep learning for music [J]. arXiv preprint arXiv:1606.04930, 2016.

[6] Denton E,Chintala S,Fergus R,et al. Deep generative image models using a Laplacian pyramid of adversarial networks[C]//Advances in Neural Information Processing Systems. 2015: 1486-1494.

[7] Mogren O. C-RNN-GAN: continuous recurrent neural networks with adversarial training [EB/OL]. (2016-11-29). https://arxiv. org/abs/1611.09904.

[8] Yang L C,Chou S Y,Yang Y H.MidiNet:a convolutional generative adversarial network for symbolic-domain music generation[EB/OL]. (2018-10-29).https://arxiv.org/pdf/1703.10847v1.

[9] Engel J,Agrawal K K, Chen S,et al. GANSynth: adversarial neural audio synthesis

[10] https://en.wikipedia.org/wiki/Long_short-term_memory

[11] Swift, Andrew. (May 1997), "A brief Introduction to MIDI", SURPRISE, Imperial College of Science Technology and Medicine, archived from the original on 30 August 2012, retrieved 22 August 2012