Music Generation with AI technology: Is It Possible?

Haowen Tang
Division of Science and Technology
BNU-HKBU United International Colloge
Zhuhai, China
n830026106@gmail.com

Yikun Gu
Division of Science and Technology
BNU-HKBU United International Colloge
Zhuhai, China
sherly28gyk@gmail.com

Xinyu Yang
Division of Science and Technology
BNU-HKBU United International Colloge
Zhuhai, China
xinyu_yang0113@163.com

2022 IEEE 5th International Conference on Electronics Technology (ICET) | 978-1-6654-8508-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICET55676.2022.9824149

***Abstract*—Music generation techniques have made tremendous progress in recent years. Through deep learning neural network frameworks, algorithms can generate music that rivals humans. However, current music generation models lack mainstream algorithms and specifications, including algorithm design criteria and model evaluation criteria, and exactly how to evaluate that music generated by a deep learning model is a good piece of music. In this case, we did a literature review for the three most common deep learning models currently used in music generation. These three models are Biaxial-LSTM, DeepJ and MuseGAN. We compared their application scenarios and model evaluation methods to provide a reference standard for subsequent researchers.**

Keywords—Music Generation, LSTM, Biaxial LSTM, DeepJ, MuseGAN, Deep Learning, GAN

I. Introduction

In the era of big data, the demand for short videos and game soundtracks has grown by leaps and bounds with the rapid development of streaming platforms. On the one hand, as long as it meets specific stylistic needs, people do not need music of very high artistic attainment; on the other hand, the demand for this type of music is often quite high, while few musicians would want to waste their time writing a lot of music of low substance. Thus, automatic composition using artificial intelligence technology and collaborative composition with musicians will become mainstream.

In this paper, we did a literature review focused on three main deep learning models for music generation, Biaxial-LSTM, DeepJ, and MuseGAN. As for these three models, we compared the application scenarios and model evaluation methods. The primary purpose is to help researchers find sufficient deep learning models for further music generation research. The Biaxial-LSTM has the feature of transposition invariance, which means that the harmonies are more dependent on the relative pitches rather than absolute pitches. The Biaxial-LSTM can generate polyphonic music, and the model evaluation mainly uses the Turing test. DeepJ can be seen as an improvement based on Biaxial-LSTM, generating specific-style polyphonic music. The model evaluation of DeepJ is also a Turing test. MuseGAN is an innovational approach based on Generative Adversarial Network that can be used to generate pop music with multiple tracks. Moreover, MuseGAN used many statistical indicators such as the ratio of empty bars and the number of used pitch classes per bar to avoid the response errors from the Turing test.

II. Methodology

A. Biaxial LSTM

Biaxial LSTM is a category of deep learning that can be considered as an extension to the LSTM model. While LSTM model can only be used to generate monophonic music, Biaxial LSTM can be used to generate polyphonic music which is a kind of music texture where multiple voices play at the same time.

Compared with LSTM model, one major improvement of Biaxial LSTM is that it has the feature of transposition invariance, which means that the harmonies are more depended on the relative pitches rather than absolute pitches. Relative pitches mean that when there is a key change for the same melody, it should still be considered as the same music.

1) Data representation

In Biaxial LSTM, a piano roll, that is a binary vector representing the notes played at each time step, is used to represent notes. In this representation, 1 is used to represent the note is being played, while 0 is used to represent the note is not being played. Therefore, any music can be represented as a N × T matrix where N represents the playable notes counts, and T represents the time steps counts. For instance, if there are two notes lasting for two time steps, and followed by two silence time steps, the corresponding representation of N=4 can be seen as:

$$t_{play} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Also, while holding a note is not the same as replaying a note, we need to distinguish these two events, so t_{replay} is also needed which looks similar to t_{play}.

2) Architecture

Biaxial LSTM generates polyphonic music by modeling every note in every time step as a probability, using all previous time steps and all notes already generated in the current time step as the condition. The conditional probability of play a note is constructed by two modules: the time-axis module and note-axis module.

$$\mathbf{P}(\mathbf{Y_{t,n}}|\mathbf{Y_{t,n-1}}, \mathbf{Y_{t,n-2}}, \ldots, \mathbf{Y_{t-1,N}}, \mathbf{Y_{t-1,N-1}}, \ldots, \mathbf{Y_{1,2}}, \mathbf{Y_{1,1}})$$

Each note is transformed to a tensor by itself and its surrounding octave, and the feature of each note is then sent into a LSTM unit, sharing weights on the time-axis between each note.

The time-axis is inspired by CNN model, and is constructed by two stacked layer of LSTM units. Each LSTM unit has weight shared between each note, so that it can force the time-axis to learn the transposition invariant feature. Contextual input is also given for each LSTM unit on the time-axis, using the binary format to find the position of time step related to a 4/4 meter.

The model takes the notes features generated from the time-axis as input, and the note-axis LSTM then scans from the lowest note feature to the highest in order to predict each note conditioned on previous predicted notes. Before feeding into note-axis f, the chosen note y_i can be determined by the note feature x_i, concatenated with the lower chosen note y_{i-1}.

$$\boldsymbol{y_i = sample(f(x_i \oplus y_{i-1}))} \quad \boldsymbol{y_1 = sample(f(x_i \oplus 0))}$$

As shown in the Fig. 3, the $\oplus$ symbol represents a concatenation, ×2 means that the module is being stacked twice.

3) Limitations of biaxial LSTM

Although these two deep learning models has their own advantages compared with other earlier models, they are also very limited in many respects. LSTM model performs weak when generating polyphonic music. Additionally, these two models may succeed in generating music with different styles, but they will fail in pivoting music with different styles within one output. They also fail to include a large range of important features in music, such as dynamics, rhythmic patterns and melodic contour. They also will fail in distinguishing different instruments in music. For instance, in pop music, there is more than one instrument playing at the same time and have their own melody and rhythmic patterns.

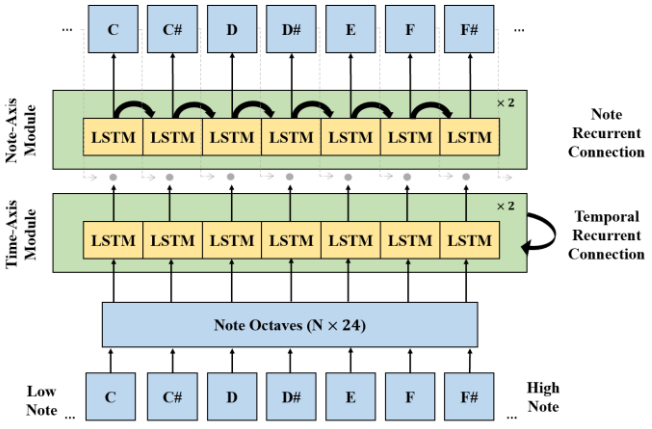

Fig. 1. Biaxial LSTM Model

Therefore, we will then introduce two deep learning models, DeepJ and MuseGAN, with corresponding implementations in the following sections. Both of these two models are related to music style, while the main difference is, the DeepJ model is applicable to all music styles, but the MuseGAN model is mostly used to generate pop music which has multiple instruments and tracks. These two models solve different problems and have their own advantages when generating music.

B. DeepJ

The DeepJ model is an end-to-end deep learning model inspired by Mao[3], Shin and Cottrell. It is developed based on Biaxial LSTM model. To demonstrate more clearly, you can refer to the mind graph of DeepJ model below. (Fig. 2)

As shown in the framework, there are five basic features concerning the DeepJ model. The most innovate part of the DeepJ model is that it had added two specific features that served the purpose of generating music based on the mixture of composer styles.

One is that the DeepJ model is style-specific, another one is that it adds consideration into the feature of music dynamics that is the volume changes of a piece of music. As this model is developed based on Biaxial LSTM, it has retained the features of generating polyphonic music texture and transposition invariance.

Now we will elaborate on the generation process of this model.

1) Data representation

The DeepJ model applies the same note representation as the Biaxial model, while adding new argument that is the music dynamic. In this representation, dynamics in music is seen as the relative volume of a single note. We encode the dynamics feature using scalar representation by adding another vector $t_{dynamics}$ which performs ideally harmoniously. Still, we keep track of the dynamics for every note in an N × T matrix. The dynamic for each note is scaled between 0 and 1, where 0 denotes that there is no sound at the current time step, and 1 denotes that the current note has the loudest possible sound. The following is one example of the previous example of holding two notes for two time steps at 0.6 volume:

$$t_{dynamics} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.6 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.6 & 0.6 & 0 & 0 \end{bmatrix}$$

In DeepJ model, we use a dataset of 23 different composers from different classical music periods. In this model, the music style is encoded as a one-hot representation. In the generation process, different styles of artists are mixed by creating vectors to represent each desired styles and normalizing the vector to make the summation equal to 1. For example, if the romantic genre contains music pieces made by composers 1 to 5, we will then denote the romantic genre as:

$$\mathbf{s_{romantic}} = [\mathbf{0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, \ldots}]$$

In order to create music with better long-term structure, we encode extra contextual inputs to guide the model. We feed this model the position of a beat for this current bar as a one-hot vector, having a dimension equal to q, which is the number of quantization per bar. In this part, We use b_i to represent the one-hot vector, and choose $q = 16$ to capture some quick notes without causing too expensive computational cost.

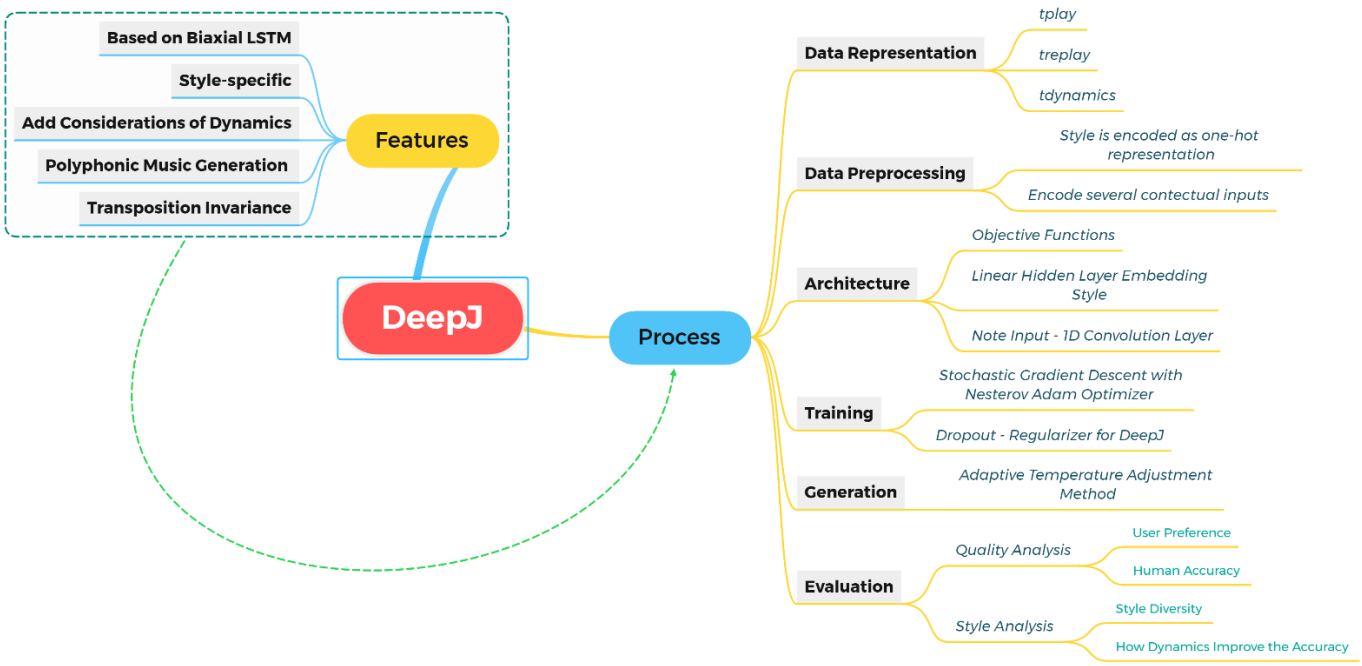

Fig. 2. DeepJ Framework

2) Objective function

In the DeepJ model, we generate three outputs in general, that is the play probability, replay probability and dynamics. The three outputs are trained together. For play and replay factors, these two are treated as a logistic regression problem obtained using binary cross entropy, while the dynamics is trained using mean squared error. The loss functions for the outputs are as follow.

$$\boldsymbol{L_{play}} = \sum \boldsymbol{t_{play} log(y_{play}) + (1 - t_{play}) log(1 - y_{play})}$$

Through experiments, we found that a naive definition for L_{replay} and $L_{dynamics}$ will result in poor performance with the replay probability low and the dynamics reaching to zero. A better way will be to mask out the loss whenever a note is not being played. The loss function for replay and dynamics will then be:

$$\boldsymbol{L_{replay}} = \sum \boldsymbol{t_{play}(t_{replay} log(y_{replay}) + (1 - t_{replay}) log(1 - y_{replay}))}$$

$$\boldsymbol{L_{dynamics}} = \sum \boldsymbol{t_{play}(t_{dynamics} - y_{dynamics})^2}$$

3) Architecture

The main difference of the architecture of DeepJ model compared with Biaxial LSTM is that we use style conditioning at each layer. A better way to represent the style of the music is to use learned distributed representation, rather than one-hot representation, since the style of different music is not necessarily orthogonal. Music in different periods, different genres could share the same features. Therefore, we project the one-hot style inputs to a music style embedding h into a linearly hidden layer W, and then use global conditioning to prevent the situation where the model will ignore the embedded style directly. For each LSTM level, we use a fully connected hidden layer that is activated by tanh to connect the style.

$$\boldsymbol{h_l' = tanh(W_l' h)}$$

where l represents the LSTM layer. For the i-th note and the input features xi, the output zi with the corresponding note is represented as:

$$\boldsymbol{zi = f(xi + h')}$$

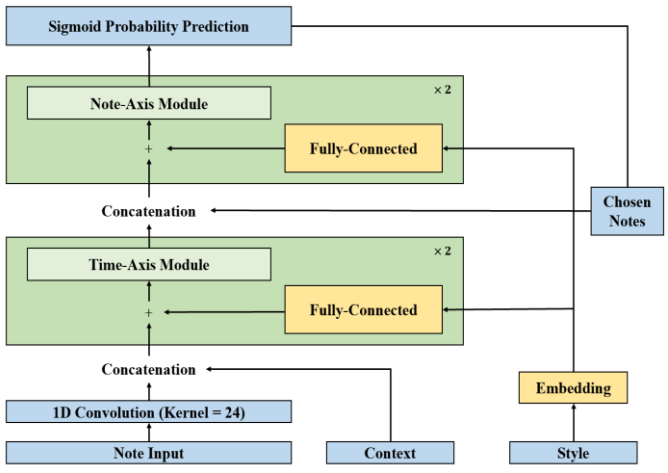

Fig. 3. Architecture of DeepJ

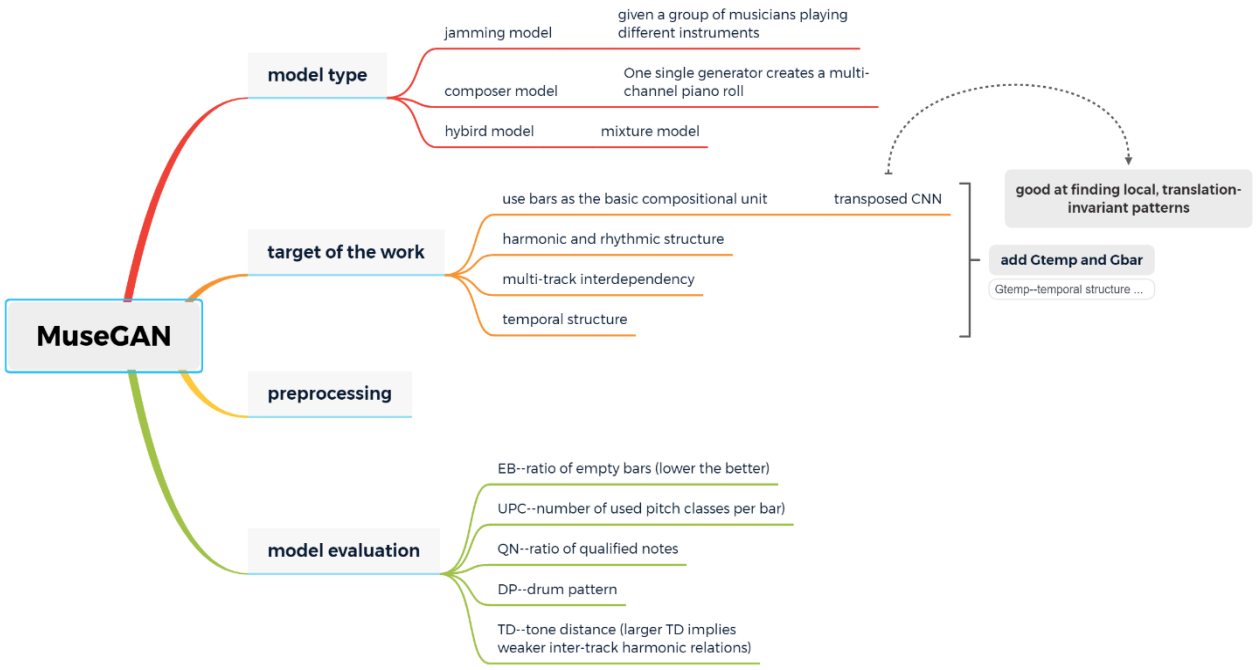

Fig. 4. MuseGAN Framework

Another improvement compared with previous Biaxial LSTM is that we add a $1D$ convolution layer with $kernel = 2$.

With the rapid development of generative models in recent years, generative adversarial neural networks have made very significant progress in data depth falsification. We are also pleased to find that this technique is now being used in music generation.

The mathematical theory that underpins GAN is extremely complex. Simply speaking, GAN is a class of machine learning techniques consisting of two simultaneously trained models: a generator, which is trained to generate pseudo-data, and a discriminator, which is trained to identify pseudo-data from real data. [5]

	generator	*discriminator*
input	A random vector	The discriminator has two sources of input: real samples from the training set and pseudo-samples from the generator.
output	The most convincing pseudo-sample.	Predict the probability that the input sample is true.
target	Generate pseudo-data that is not identical to the data in the training set.	Distinguish between pseudo-samples from the generator and real samples from the training set.

Fig. 5. Key information for generators and discriminators

The training process of GAN can be seen as a zero-sum game, and eventually the GAN model will reach a Nash equilibrium, i.e., the pseudo-samples generated by the generator are no different from the real data in the training set.

4) Apply GAN into pop music generation

Pop music is usually composed of multiple instruments/tracks. A modern orchestra often contains 5 instruments, which are bass, drums, guitar, strings and piano. The authors binarize the music tracks of different instruments by musPy in the preprocessing section to obtain a piano roll matrix. In the next experiments, the authors use three models, Jamming model, Composer model, Hybrid model, which have different implementation mechanisms, but they are all based on Generative Adversarial Neural Network (GAN).

Due to the existence of a long-term independent structure of multitrack pop music, the authors used bars as the most basic unit for model training. To extract as many melodic, harmonic and arpeggio features as possible in the music, the authors used a transposed convolutional neural network (Transposed CNN), a structure that extracts features by reducing the number of channels and increasing the matrix size.

Then the cost function of the model can be written as follows:

$$\boldsymbol{min_G max_D E_{x-p_d}}[\mathbf{log}(\boldsymbol{D}(\boldsymbol{x}))] + \boldsymbol{E_{z-p_z}}[\mathbf{1} - \mathbf{log}(\mathbf{D}(\mathbf{G}(\mathbf{z})))]$$

where p_d and p_z represent the distribution of real data and the prior distribution of z respectively.

5) Data pre-processing and representation

In the pre-processing process for multiple tracks of popular music, the authors used the musPy tool previously developed by their team to binarize the music tracks into a binarized image matrix. The horizontal coordinates of the binarization matrix represent time, while the vertical coordinates represent pitch. The black part on the image suggests that a note is played at the specific time step. [6]. To put in a formal way, a piano roll with M-track with length of one bar is represented as a tensor $x \in$

$\{0,1\}^{R*S*M}$, where R and S denote the counts of time steps and the counts of note candidates, respectively.

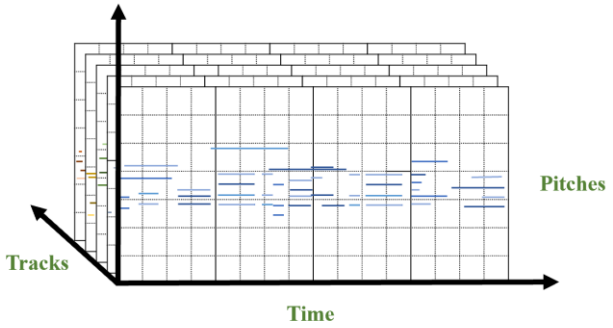

Fig. 6. Multitrack piano-roll representation (binarization matrix)

In the process of data pre-processing, authors found that the tracks in original dataset tend to play only a few notes in the entire songs. This will make the data training sample sparse, which will hinder the training process of the model. The authors dealt with the problem of unbalanced training data by merging tracks of similar music. However, the authors also mentioned that this method increases the noise of the data, but the effect is better than the training samples with a large number of empty bars. The authors stored the pre-processed data in Lakh Piano roll Dataset (LPD), and the subsequent LPD-5-matched dataset saved data with higher confidence so that to improve the training of the model. Since music generation requires an extremely large number of training samples, notes below C1 and C8 are so sparse that they cannot meet the training requirements. The authors eliminated the high pitches and low pitches during pre-processing. The size of the target output tensor is 4 (bar) x 96 (time step) x 84 (note) x 5 (track).

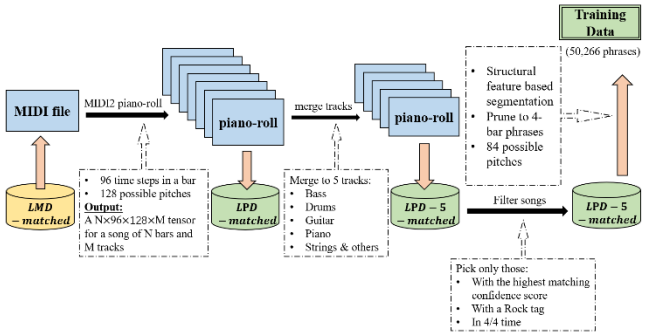

Fig. 7. Flow chart of pre-processing

6) Architecture

GANs implements adversarial learning mainly by constructing two sub-network generators and discriminators. (The number of generators and discriminators corresponds to the number of instrument types.) The generators map randomly generated noise Z sampled from a precedent distribution to our data space. In more technical terms, the goal of the generator is to generate samples that match the data distribution of the training dataset. The training process can be modeled as a game process between a generator G and a discriminator D. The authors used three models based on GANs, which are Jamming model, Composer model and Hybrid model respectively.

For the Jamming model, it is equivalent to each member of the band having an independent random noise Z. Each member generates music based on its own GAN model. The problem of this model is that it cannot capture the dependencies between different instruments.

The composer model is a modification of the previous interference model. The interference model is equivalent to each band member playing his or her own music, which makes the resulting music sound very messy. The composer model acts as a conductor by introducing a shared random vector z.

For the Hybrid model, the random vector z represents the random vector between tracks and z_i for the random noise of each member of the band in the interference model. The hybrid model has a more flexible network structure and can dynamically invoke different layers and filter sizes.

To model the temporal structure of multi-track music, the generator consists of two sub networks, the temporal structure generator G_{temp} and the bar generator G_{bar}. G_{temp} maps a noise vector z to a sequence of some latent vectors, $z = \{z^t\}_{t=1}^{T}$. The resulting z, which is expected to carry temporal information, is then be used by G_{bar} to generate piano-rolls sequentially.

$$\boldsymbol{G(z) = \{G_{bar}(G_{temp}(z)^{(t)})\}_{t=1}^{T}}$$

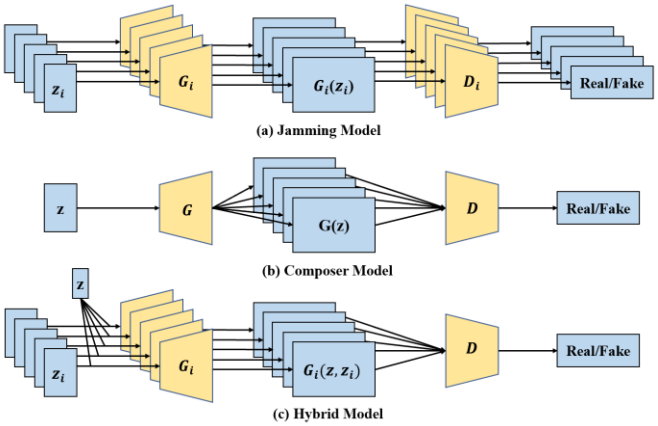

Fig. 8. Introduction of three GANs models

III. Comparison

A. Between Biaxial LSTM, DeepJ and MuseGAN

For the three models, although they are all cutting-edge research in the field of AI-generated music, they all have very different model architectures, which results in different music being generated by each of them.

Biaxial LSTM is an upgraded model based on LSTM can be applied to various music styles. Biaxial LSTM can distinguish relative pitches and therefore can handle polyphonic music, which is the biggest improvement over the LSTM model. While for the simple LSTM model, it can only handle monophonic music. However, the Biaxial LSTM is not able to make the style of the output music consistent. The music generated by this model is independent of the music style.

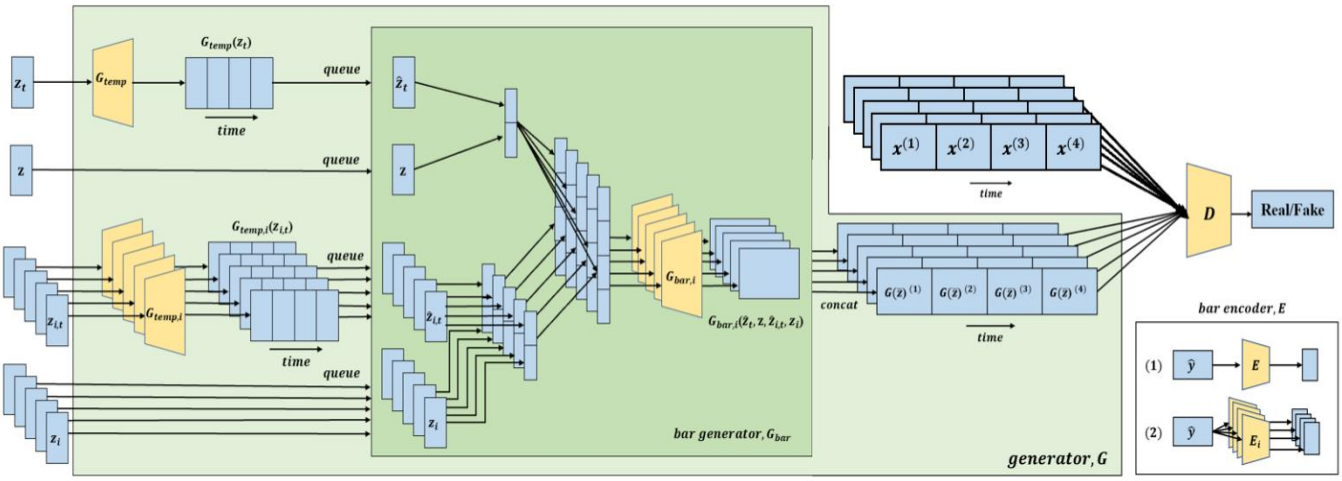

Fig. 9. GAN with temporal structure

Therefore, in the next section, we will focus on comparing the DeepJ and MuseGAN models, as two models related to music styles, in order to do better processing for the music style.

In this paper, the DeepJ model is mainly used for the generation of classical music, and the generated music style also focus on classical music. However, theoretically it can be extended to generate other types of music, only the model needs to be modified in the future. The MuseGAN model, on the other hand, is mainly targeted at modern pop music and usually contains a variety of instruments such as drums, bass, guitar, orchestra and piano. This model is related to musical style, but is more suitable for polyphonic music generation with multiple tracks.

Even though the two models are both in the music generation area, the research objectives of them are also unique and have a lot of differences. Therefore, the features considered in the datasets are also different. In the following sections, we will compare the research objectives, features in datasets and the evaluation methods taken by DeepJ model and the MuseGAN model.

B. Research Objectives

1) DeepJ model

Although the algorithms before DeepJ can generate listenable music, they are unable to maintain the consistency in style, and cannot generate a mixture of styles. The research objective of DeepJ is to do style conditioning and take into considerations for dynamics, thus brings music closer to human composition. Certain "mood" can be generated also, to fit the style that the user wishes to compose.

2) MuseGAN model

Since the existing convolutional generative adversarial networks (GANs) models for generating music have some limits and cumbersome pre-processing such as, hard thresholding (HT) or Bernoulli sampling (BS), the research objective of MuseGAN is to improve a convolutional GAN model that can use binary neurons to directly creates binary-valued piano-rolls.

C. Features in Datasets

1) DeepJ model

The dataset of DeepJ includes MIDI files of pieces composed by 23 well-known composers in the three major periods of classical music (Baroque period, classical period, and romantic period). The soundtracks are all polyphonic piano pieces. To reduce the input dimension of the notes, we cut the range of the MIDI values to 36-84 that is four octaves.

2) MuseGAN model

The goal of MuseGAN is to generate pop music of multiple tracks in piano-roll format. In order to make the generated music keeps its instruments characteristic, in other word, preserve the multitrack inter-dependency, they try to use different tracks to represent the instruments. Besides, in this dataset, the temporal resolution was set as 24 time steps for one beat, thus, the concept of velocity was ignored. Also, there are totally 84 possibilities for note pitch from C1 to B7. Therefore, the size of the target output tensor is $4(\text{bar}) \times 96(\text{time step}) \times 84(\text{pitch}) \times 5(\text{track})$.

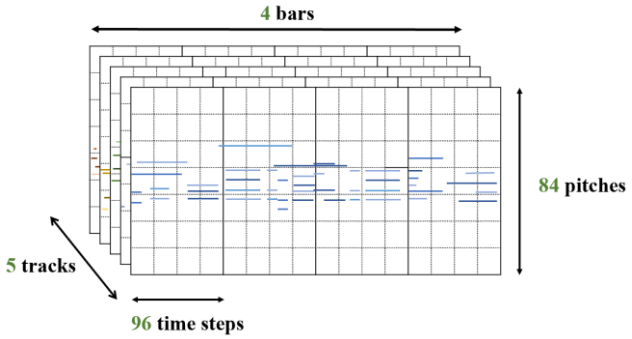

Fig. 10. Size of the target output tensor

IV. Evaluation

A. DeepJ

The evaluation process for DeepJ is split into two parts: quality analysis and style analysis. In quality analysis, an evaluation was taken based on the user preference between

Biaxial and DeepJ. 20 users are selected to listen 15 random samples generated by DeepJ and Biaxial, respectively. The statistical results show that 210 out of 300 users prefer music generated by the DeepJ model.

Style analysis carries out its test by examining the style diversity generated. 20 people with musical background are selected to classify the classical period for the specific piece they heard. 10 of them is tested by the control samples, which are real classical music composed by human, another 10 is tested by the generated samples. The results showed that there is no significant difference between the accuracy of classifying between DeepJ and human composers. An interesting finding is that, since there are composers of the transition between different eras, the music generated by DeepJ gives higher accuracy in classifying. This is because music generated for certain genre has the average characteristics of its class, and therefore easier to be recognized.

T-SNE (T-Distributed Stochastic Neighbor Embedding) visualization is used to do cluster analysis. (Figure 11) We may find that composers from the same period tend to stay together. Beethoven, who is the yellow point right next to the red point, is a representation for the transition between classical and romantic periods, which proved our observation. We also made another 3-D visualization with x-axis, y-axis, and z-axis representing baroque, classical and romantic period, respectively.

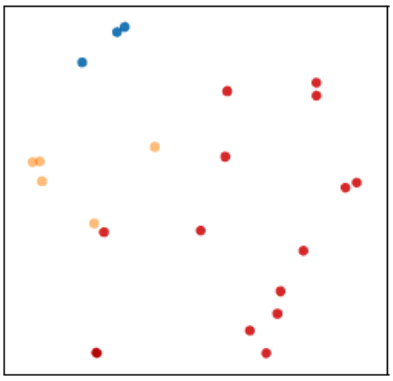

Fig. 11. T-SNE visualization

Another evaluation result is that, the adding of dynamics helps to improve the quality of music, and the reason is that, dynamics will help to provide an extra information of emotion. Another hypothesis raised by the author is that, dynamics makes the training process more complex, thus it can learn a better model.

B. *MuseGAN*

Most of the current evaluations of models in the field of research on music generation are based on Turing tests in the form of questionnaires, where listeners are asked to discern whether the music was played by a human or a computer. However, in this paper the authors give some metrics that can be used to quantitatively evaluate the goodness of computer generated music. This also makes the evaluation of the model for MuseGAN more reliable than the questionnaire.

The indicators includes:

Ratio of empty bars (in %, the lower the better) (EB)

Number of used pitch classes per bar (from 0 to 12) (UPC)

Qualified note rate (QN): the proportion of the counts of qualified notes to the total counts of notes. A lower value implies that the music is too fragmented.

Polyphonicity (PP): the ratio of time steps with more than two notes playing together to the total counts of time steps.

Tonal distance (TD): the tonality distance between the chromatic features (one per beat) of two tracks.

User Study: each participants are required to compare two sections of generated music with length of four bars and voted for the better one in four measures, completeness, harmonicity, rhythmicity and overall rating respectively.

1) *Comparison*

DeepJ model uses controlled variable method, and test the model performance based on how human will interpret the music. This is because music is very subjective fundamentally, and cannot be simply evaluated by mathematical indicators.

In MuseGAN model, the evaluation is mainly based on several indicators of the generated music, which show how complex a music piece is. This model also provides human preference test which is the indicator “user study” to make the evaluation more complete.

V. Conclusion

At the beginning of the project, Biaxial LSTM model was explored as the initial music generation experiments. This model has the ability to generate polyphonic music, but since the algorithm still does not have the ability to filter music styles in order to customize music generated, two cutting-edge researches on AI music generation were further explored, which are the DeepJ and MuseGAN models.

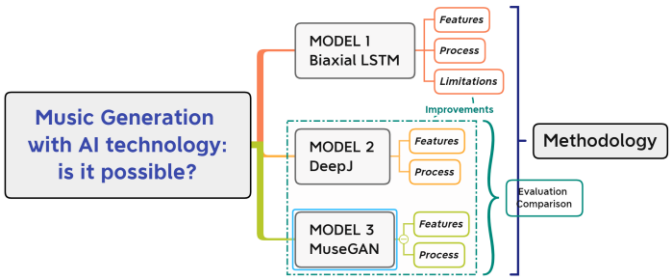

Fig. 12. Overview

The existing models DeepJ model is an optimization for the Biaxial LSTM model which is a deep learning model that can theoretically be used for various styles of music, as it incorporates considerations of musical style and dynamics. The MuseGAN model is another style-related model that is used to generate music with multiple instrument tracks, normally used to generate modern pop music. The underlying neural network models and mathematical logic used in the two algorithms are quite different, and therefore the application for these two models is also different.

In theory, the DeepJ model can be generalized to other types of music, but modifications to the model are required, so the music generated here is classified into three categories of classical music which are Baroque, Classical and Romantic periods. Compared with DeepJ model, the MuseGAN model is more suitable for generating pop music with multiple instruments and tracks, and can be used in the future for game soundtracks, etc.

Furthermore, our team compared the differences between the two algorithms in the evaluation phase. Compared to the MuseGAN model, the DeepJ model introduces human evaluation and therefore has a deeper dimensionality. Human evaluation is also a more convincing way of evaluation than pure metrics, because music is an abstract art that can be felt by humans, while data indicators maybe can prove the complexity of music, but cannot replace human feelings.

In conclusion, we studied several cutting-edge algorithms in the field of AI-generated music, analyzed, implemented and compared their strengths, weaknesses and differences. Hopefully, in the future, these algorithms can be used to their maximum value in the field of artificial intelligence music generation.

ACKNOWLEDGMENT

We would like to thank our instructor Dr. Zhang Changjiang for his careful guidance and for giving us much help in the past months. We would also like to thank the team members for the wonderful cooperation. The successful completion of this project could not have been achieved without everyone's efforts.

REFERENCES

[1] M. M. Team, "Recurrent Neural Networks – with memory," [Online]. Available: https://www.mo-data.com/mo-tech/recurrent-neural-networks-with-memory/.

[2] S. Skúli, “How to Generate Music using a LSTM Neural Network in Keras” 2017. [Online]. Available: https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5.

[3] H. H. Mao, T. Shin and G. W. Cottrell "DeepJ: Style-Specific Music Generation," arxiv.org, 3 1 2018.

[4] V. Bok, GANs in Action, 2017.

[5] H. W. Dong, W. Y. Hsiao, L. C. Yang and Y. H. Yang "MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment," 2019.

[6] E. Raffel, "Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching," 2016.

[7] Oinkina, “Understanding LSTM Networks”, 2015.

[8] B. Or, “The Exploding and Vanishing Gradients Problem in Time Series”, 2020.