Multi-category MIDI music generation based on LSTM Generative adversarial network

Yutian Wang[a], Guochen Yu[b], Juanjuan Cai[c], Hui Wang[d,*]

Key Laboratory of Media Audio & Video, Ministry of Education, Communication University of China, Beijing, China

[a]wangyutian@cuc.edu.cn, [b]yuguochen@cuc.edu.cn, [c]caijuanjuan@cuc.edu.cn, [d]hwang@cuc.edu.cn

*Corresponding

Keywords: Music generation, GAN, RNN, midi, chords.

Abstract: Music generation by neural networks has become a central issue since deep neural networks demonstrated their ability in learning from big data collections. This paper proposes a music score generation model which employs multi-layer RNNs and GAN scheme. First of all, the midi sequences are passed to the model, which is parsed as tone lengths, frequencies, intensities, and timing, and then the music theory law is introduced, while the initial sequences are set as music chords. Consequently, the distribution of music is learned in the process of training. The experimental results show that it is a feasible network structure which can generate multi-category music with good hearing experience.

1. Introduction

It has been a long time to use computers to generate music since 1959 when the first music composite algorithm was proposed [1]. In recent years, with the prevalence of deep learning, the ability of neural networks in learning from big data has made people begin to apply DNN to generate music [2] . There has been a tremendous amount of deep neural network models for music generation, most of whom uses the RNNs and their modified architectures[3,4,5,6,7], which is presumably because the music generation is inherently belonged to sequence generation[7]. For music generation, it is mainly divided into symbolic-domain generation (generating MIDIS) and audio-domain generation (generating WAVS). Famous examples include the MelodyRNN models [5] for symbolic-domain generation and the SampleRNN model [6] for audio-domain generation. Generally, these models encode the features of music (such as pitch, frequency, etc.)to latent codes, and generate a reasonably realistic music sequence by decoding these codes. In fact, the encoder-decoder architecture has its unavoidable shortcomings. For example, the selected latent codes is possibly not an ideal indicator, while some crucial information is likely lost in the process of reducing dimension [8].

Generative adversarial networks (GANs) are designed with the aim of generating realistic data and have been widely acclaimed [9]. Using this framework can highly fit the true distribution of the data, which is not necessary to use the Markov chain for repeated sampling [10]. Besides, GAN can generate date without much inferring during the learning process and avoid the need to calculate tricky probabilities.

Recurrent neural networks (RNN) are widely used to model sequences of data [11], which have a good effect in modelling text sequences and audio sequences [12]. In 2002, ECK modeled blues songs with 25 discrete tone values [13]. In addition, there was a work combining the RNN with restricted Boltzmann machines and representing 88 distinct tones [14]. Just a 2 years ago, Daniel Johnson used bi-axis LSTM to model MIDI music [15], which generated polyphonic music with great results. Recently, a generative model called MIDINET by using CNN and GAN attracted a lot of attention and generated pleasing music [16].

Accordingly, this paper proposes a model named LSTM-GAN for Multi-category MIDI music

generation, which is trained with pop, classical and jazz MIDI files, then generates MIDI music with imposing initial chords laws. Finally, this paper uses different metrics such as scale consistency and tone span to evaluate generated music.

2. Methods

2.1 Generator and discriminator

The core idea of GAN is that the generator is trained to generate data which are indistinguishable from real data, while the discriminator is trained to identify the generated data. The generator *G c*an be implemented as MLP, CNN and RNN, and *G(z)* is a mapping function that maps the random noise *z* to the new data space. The output *D(x*) of the discriminator *D* is a scalar, indicating the probability that *x* is derived from real data rather than generated data. The objective function of GAN is as follows:

$$\min_G \max_D (D,G) = E_{x \sim pdata(x)} \log(D(x)) + E_{z \sim pz(z)} \log(1 - D(z)) \tag{1}$$

From the perspective of discriminator *D*, it is supposed to distinguish between real and generated samples, so it is expected that *D(x)* is as large as possible, while *D(G(z))* is as small as possible. From the perspective of the generator *G*, it is expected to try the best to fool the *D*, which means *D(G(z))* should be as large as possible. In other words, *D* is supposed to maximize *V(D,G)*,while *G* tries its best to minimize *V(D,G)* by contrast. In this paper, the two models use Recurrent neural networks(RNN) and finally reach the global optimum in parallel. The loss functions *LD* and *LG* are defined as follows:

$$L_G = \frac{1}{m} \sum_{i=1}^{k} \log(1 - D(G(z^{(i)}))) \tag{2}$$

$$L_D = \frac{1}{m} \sum_{i=1}^{k} [-\log(D(x^{(i)})) - \log(1 - D(G(z^{(i)})))] \tag{3}$$

2.2 LSTM-GAN model

In this model, the Long short-term memory(LSTM)[17] is used which has an internal structure of gates to help avoid the gradient vanishing and decrease the extremely large calculation when the time sequence is much long[17]. The discriminator D uses a bi-directional LSTM which can make decision in both directions with more complete context modeling and better sequential learning.

In order to get better generated effect, this model adds chord laws when training music. For every major tune C, D, E, F, G, A, B, there are only 3 basic triads which is shown in Table I:

TABLE I. The basic triads of 7 tones

Tones	Basic triads		
C	*135*	461	572
D	*2#46*	*572*	*6#13*
E	*3#57*	*6#13*	*7#2#4*
F	*461*	*7#2#4*	*135*
G	*572*	*135*	*2#46*
A	*6#13*	*2#46*	*3#57*
B	*7#2#4*	*3#57*	*461*

During the generation process, *Cadd9* chord is used as the initial chord, which means the C chord (*135)* plus *9*(*2* in a high octave). That is to say when the generator generates the music, it learns to generate note according to our given initial chords, which produces better pop music.

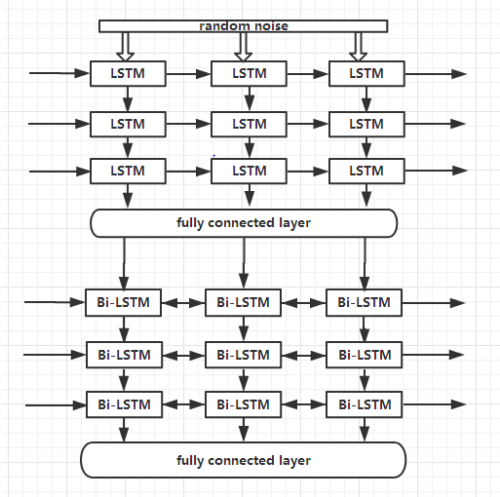

Fig.1 This model uses 3-layer LSTM [17] as generator and 3-layer bidirectional LSTM as discriminator.

3. Implementation

3.1 Dataset

The training data are MIDI music collections, including famous classical music, jazz music and some pop music. Each MIDI note is parsed and saved with its duration, tone length, intensity, and time spent since the previous tone, which can represent polyphonic chords. All data are normalized to a 440 tick for each quarter note, and each midi is a standard 4-4 beat, while each octave is divided into 12 major tones. The data includes 3, 697 midi files from 160 different classical music composers, 391 MIDI files from 74 different jazz composers and 200 MIDI files from 57 different pop music composers.

The LSTM of both *G* and *D* has 3 hidden layers, each LSTM cell has 200 hidden units. The input of *G* is the random noise and the output of the previous unit, while the output of each LSTM unit in *D* is fed into a fully connected layer whose weights are shared with time steps, and then a sigmoid output of each unit from each LSTM cell is averaged to the final decision of the sequence.

3.2 Training step

This paper uses mini-batch stochastic gradient descent with the batch-size set to 20, while using the standard end-to-end Backpropagation through time(BPTT). The learning rate is set to 0.1, and L2 regularization is applied to the weights of the generator and discriminator to prevent overfitting. This model is pre-trained for 12 Epochs with squared error loss for predicting the next event in the training sequence. As in the adversarial setup, the input to each LSTM unit is a random vector z that is connected to the output of the previous time step. z is uniformly distributed among $^{[0,1]^k}$, and k is selected as the number of features in each tone, 4. During the training process, a schema for sequence length is proposed, which train the model with short sequences at the beginning, then increase gradually.

A technique called “frozen”[18] is used in LSTM-GAN to solve the gradient disappearance and explosion which cause it difficult to achieve Nash equilibrium for training GAN. When D is trained to be overly strong and consequently leads to the gradient disappearing of G, the excessively powerful party is frozen, and vice versa. That is to say the update of D’s gradient is stopped when the discriminator training loss is less than 70%of the training loss of G. The same thing is done when G becomes too strong.

When the discriminator passes a gradient to the generator, it generally causes the generated sample have a larger gap with the real sample because the G is used to deceiving the D instead of fitting the real music. To solve it, this model uses feature matching[18], a method that encourages

the samples generated by G have greater changes, and consequently avoids overfitting to the current discriminator by replacing the standard loss of generator, LG. The representations R is chosen from the last layer before the final logistic classification layer in D, and the new objective LG for G is defined as follows:

$$\hat{LG} = \frac{1}{m}\sum_{i=1}^{k}(R(x^{(i)}) - R(G(z^{(i)})))^2 \tag{4}$$

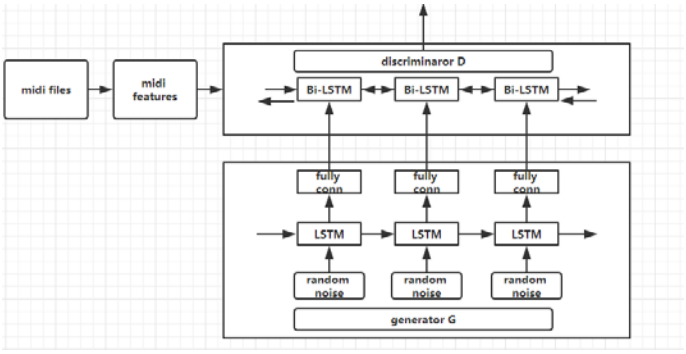

Fig.2 The training step

4. Result

4.1 Classical music evaluation

We use these following six metrics: Scale consistency, Tone span, Unique tones, Intensity span, Polyphony and 3-tone repetitions to evaluate our generated classical music.

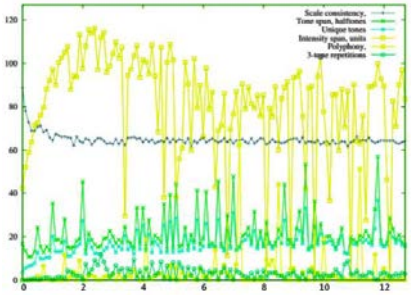

Fig.3 Generated music (with frozen and feature matching)

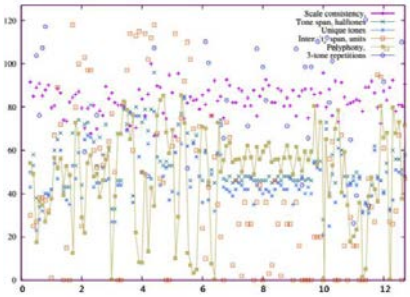

Fig.4 The real data

Compared with the real music, the scale consistency of the generated music has a certain decline, and the consistency of the scales tends to be stable with the length of the music increasing, while the range of tones and intensity are nearly same as the real ones. With the training progresses, the music produced by LSTM-GAN is becoming more and more complex. In addition, the number of unique tones increases clearly during the first 15 epochs, which means the generated music becomes more and more complicated. Meanwhile, the 3-tone repetition has an increasing trend during the first 25 epochs and then remains at a lower level.

As shown in the diagram, The 3-tone repetition rate of real music is apparently higher than generated ones. Normalization is required for comparison because of the different length of real music and generated ones. Count comparison is performed by dividing by *lr*/*lg*, where *lr* is the average length of real music and *lg* is the average length of the generated music.

4.2 Pop music evaluation

TABLE II. Result of a subjective evaluation comparing LSTM-GAN with Dense, Basic LSTM, Midinet-1D, Midinet-2D (with chord condition) and Biaxial-LSTM.

Different models	How real	How pleasing	How complicated
Dense	1.21	1.51	2.62
Basic LSTM	1.65	1.73	2.50
LSTM-GAN	3.52	3.38	3.16
Midinet-1D	3.18	3.25	3.02
Midinet-2D	3.72	3.63	3.14
Biaxial-LSTM	3.05	3.18	2.98

To evaluate the aesthetic quality of the pop music generation result, this paper conducts a subjective evaluation with more than 30 participants, and most of them understand basic music theory and have the experience of being an amateur musician.

We compared LSTM-GAN with 5 different music generation models: Dense, basic LSTM, Midinet-1D[16], Midinet-2D(with chord condition)[16] and Biaxial-LSTM[15]. All of these models used pop music for training, and generated pop music with similar style. We asked the participants to rate the generated melodies in terms of the following three metrics: how real, how pleasing, and how complicated, from 1(low) to 5(high) in a five-point Likert scale.

The result of the user study is shown in Figure 6, where the following observations can be made. First, the LSTM-GAN model consistently outperform the Dense and the basic LSTM across the three metrics. The mean values for LSTM-GAN are around 3.4 for being pleasant and realistic, and around 3.2 for being complicated.

Second, compared with Midinet-1D, Biaxial-LSTM, LSTM-GAN also has a better performance across the three metrics, where our model has significantly scores of how real and how pleasing. When we used the Cadd9 chord as the initial chord, the generated pop music is almost as realistic, pleasing and complicated as Midinet-2D(with chord condition).

Third, in these 6 generative models, Midinet-2D[16] has the best performance, presumably because it uses the chord condition when training the model with music theory law.

5. Conclusion

In this paper, a model for music generation is proposed which uses the LSTM with adversarial training to fit the joint probability distribution of the whole real data and generate corresponding music sequences. MIDI with significantly improved effects is generated by training the sequences of different styles of the midi files and adding *cadd9* initial chord. Through the experiments, the LSTM-GAN model can learn the essential characteristics of the music better and generate a more melodic and pleasing music after using initial chord laws.

Acknowledgments

This research was supported in part by the NSFC grant (No.61501410 and No.61631016) and the Engineering Planning Project of Communication University of China (Grant No. 2018XNG1805).

* Hui Wang the corresponding author.

References

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

[2] Todd P M. A Connectionist Approach to Algorithmic Composition, Computer [J]. Computer Music Journal, 1989, 13(4):27-43.

[3] Todd P M B J. Modeling the perception of tonal structure with neural nets [J]. Computer Music J, 1989, 13(4):44-53.

[4] George Papadopoulos and Geraint Wiggins.AI methods for algorithmic composition: A survey, a critical view and future prospects.In AISB Symposium on Musical Creativity,pages 110–117,1999.

[5] Elliot Waite, Douglas Eck,Adam Roberts,and Dan Abolafia.Project Magenta:Generating long term structure in songs and stories,2016.https://magenta.tensorflow.org/blog/2016/07/15/lookback-rnn-attention-rnn/.

[6] Soroush Mehri,Kundan Kumar,Ishaan Gulrajani,Rithesh Kumar,Shubham Jain,Jose Sotelo, Aaron C. Courville, and Yoshua Bengio.SampleRNN:An unconditional end-to-end neural audio generation model. arXiv preprint arXiv:1612.07837,2016.

[7] Jean-PierrBriot,Gaëtan Hadjeres,François Pachet.Deep Learning Techniques for Music Generation-A Survey.arXiv preprint arXiv:1709.01620

[8] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate [J]. Computer Science, 2014.arXiv preprint arXiv:1409.0473

[9] IanJ.Goodfellow and Yoshua Bengio.Generative adversarial nets.In Advances in Neural Information Processing Systems,pages 2672–2680,2014.

[10] Goodfellow I . NIPS 2016 Tutorial: Generative Adversarial Networks [J].2016.arXiv preprint arXiv: 1701.00160,2017.

[11] Alex Graves.Generating sequences with recurrent neural networks.arXiv preprint arXiv: 1308.0850,2013.

[12] Tomas Mikolov, Martin Karafiát,Lukas Burget,Jan Cernocky,and Sanjeev Khudanpur. Recurrent neural network based language model.In Interspeech,volume 2,page 3,2010.

[13] Douglas Eck and Juergen Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks.In Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on,pages 747–756.IEEE,2002.

[14] Pascal Vincent Nicolas Boulanger-Lewandowski, Yoshua Bengio. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In Proceedings of the 29th International Conference on Machine Learning (ICML), page 1159–1166, 2012.

[15] Daniel Johnson. Biaxial Recurrent Neural Network for Music Composition. http://www.hexahedria.com/

[16] Li-Chia Yang,Szu-Yu Chou,Yi-Hsuan Yang.MidiNet:A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation. ISMIR(International Society of Music Information Retrieval),2017,arXiv:1703.10847v2

[17] Jürgen Schmidhuber and Sepp Hochreiter. Long short-term memory. Neural computation, 7(8):1735–1780, 1997.

[18] Tim Salimans, Ian Goodfellow, Wojciech Zaremba,Vicki Cheung, Alec Radford,and Xi Chen. Improved techniques for training Gans.In Advances in Neural Information Processing Systems, pages 2226–2234, 2016.