Proceedings of the 2nd International Conference on Computing Innovation and Applied Physics
DOI: 10.54254/2753-8818/19/20230512

Using LSTM neural network to generate music

Yujie Zhang

College of Computer science, Sichuan university, Chengdu, Sichuan, 610065, China

imzhangyujie@163.com

Abstract. Music generation is a cutting edge and useful research filed, which is helpful for artists to compose novel melodies as well as revealing potential patterns of music. Recurrent neural network (RNN) is a member of the neural network family, which is commonly used for processing sequential data. It can deal with sequential changes in data compared to normal neural networks. Long short-term memory (LSTM) aims at improving the conventional RNN. It is designed to alleviate the deficiencies of gradient disappearance and gradient explosion that possibly happened in RNN during training. In simple terms, LSTM is superior at grasping long term information than normal RNN. It can record the information that requires to be recorded for a long time and abandon these unimportant features. Unlike RNN, which have merely one way of stacking long-term information. It's quite useful for tasks that require long range dependence. In this work the effectiveness of the LSTM is validated on the music generation task.

Keywords: LSTM, RNN, deep learning, music generation.

1. Introduction

Composition has been thought of as a specialized skill. Beautiful melodies could make audience relaxing, and exciting melodies could inspire people [1, 2]. Conventionally, musicians will take quite a long time to compose music, and some classical music will take even a life-long time to finish, which is extremely time consuming and requires years to learn the theory behand composition [3, 4]. To compose music more efficiently and give normal people the opportunities to compose their own music, some algorithm-based methods are proposed to help composition.

In recent years, machine learning is facing a rapid development [5, 6]. Models developed for human faces and videos generation is even real enough to make them indiscriminative to human eyes [7]. Machine learning can actually generate music by learning patterns in existing music [8, 9]. RNNs are widely used in natural language processing because the generation of words is successively related to each other. In fact, the same is true of music, where the features of the next note are correlated with the features of the previous notes. Therefore, RNN [10] is one of the important ways to achieve machine learning composition.

In this paper, a set of piano MIDI files is used for model training. Finally, given a sequence of notes, the model can predict the next character in the sequence based on the initial note sequence, and generate a complete MIDI file. The basic approach is divided into several main steps:

Firstly, the note sequence is preprocessed to obtain the training data set, Then the recurrent neural network is trained on dataset to achieve the neural network model, Next, the neural network model is

DOI: 10.54254/2753-8818/19/20230512

given some input to generate the entire sequence of notes, Finally, the note sequence file, the midi file, is converted to an audio format, such as mp3.

2. Introduction

2.1. Dataset

As for the dataset, the bach doodle dataset is selected for learning in this project. this dataset was launched by google in 2019, which generated 20 million midi samples. Every midi file contains the pitch, step, duration of each note.

As for the data processing, firstly, all the files in the dataset are iterated and obtain the features of each MIDI file. Then the array data of in each file is converted to array data for learning.

Then a sequence of features is constructure from this tensor data, and the label is the note at the next moment in the sequence, a window method in Tensorflow is leveraged as a sliding window, while property called sequential length is set as the length of the sequence. To extract the last pitch for learning, a function is implemented to split the last data in a window into labels, and the previous data in the sequence as features. In this work, the sequential length equals 25 is used, so the size of the data generated per sample is 25 * 3.

2.2. RNN

In traditional machine learning models, such as CNN, the previous stage cannot be stored in the model. As a result, the sequential data could not be learned properly. However, a recurrent neural network, often called RNN, could be leveraged to store the previous stages and hence suitable for learning the sequential data.

RNN have a repetition module which is designed to pass the feature from the previous level and uses corresponding output for next level. However, RNN can only maintain recent information from adjacent stages, it is not good at storing a long-term information. As a result, the generated music may not maintain a uniform style and the melody in the beginning and the end of the music may sound quite different. So, the network requires a novel mechanism to maintain long-term dependencies. Hence, the LSTM is proposed.

LSTM is improved based on RNNs. It possesses a similar chain-like architecture as RNNs. Differently it has a different repetitive module. For the model part, this work use tf.keras.layers.LSTM(128). There are three main model output values, one is all output values, one is the output features of the last layer, and one is the hidden layer state value. Architecture of the whole model is demonstrated in Figure 1 and the parameters are shown Figure 2.

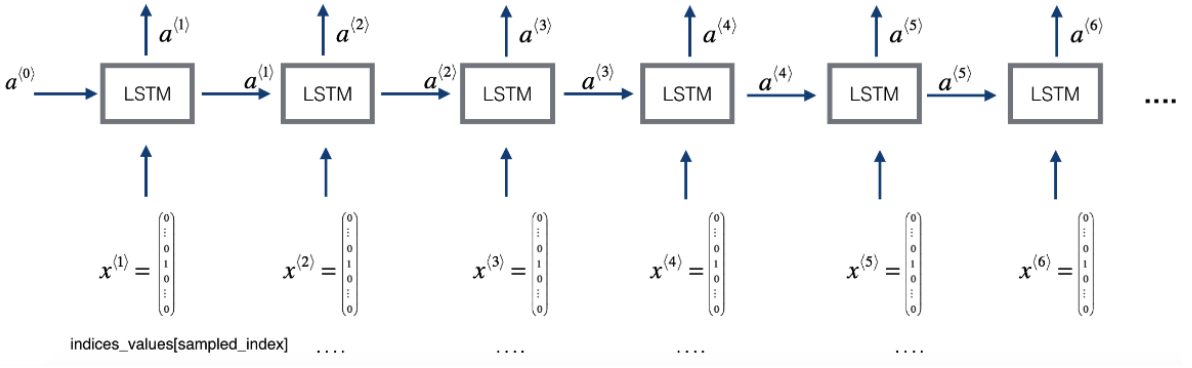

Figure 1. the architecture of the LSTM model.

DOI: 10.54254/2753-8818/19/20230512

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 25, 3)]	0	[]
lstm (LSTM)	(None, 128)	67584	['input_1[0][0]']
duration (Dense)	(None, 1)	129	['lstm[0][0]']
pitch (Dense)	(None, 128)	16512	['lstm[0][0]']
step (Dense)	(None, 1)	129	['lstm[0][0]']

Total params: 84,354
Trainable params: 84,354
Non-trainable params: 0

Figure 2. Number of parameters of the LSTM model.

2.3. Training

The training part has three main outputs, the predictions for 'pitch', 'step', and 'duration', using a mean square error-based loss function to encourage non-negative values.

The loss of step and duration use the MSE with positive pressure method as the loss function. The loss of pitch mainly uses the cross-entropy loss between the ground truth and the predicted results. This is because in the RNN model, pitch is the output of all results, while step and duration are the output of a single moment as well as the hidden layer output, the way of calculating the loss is different. And the model is mainly optimized for loss by using Adam optimizer as shown in Figure 3.

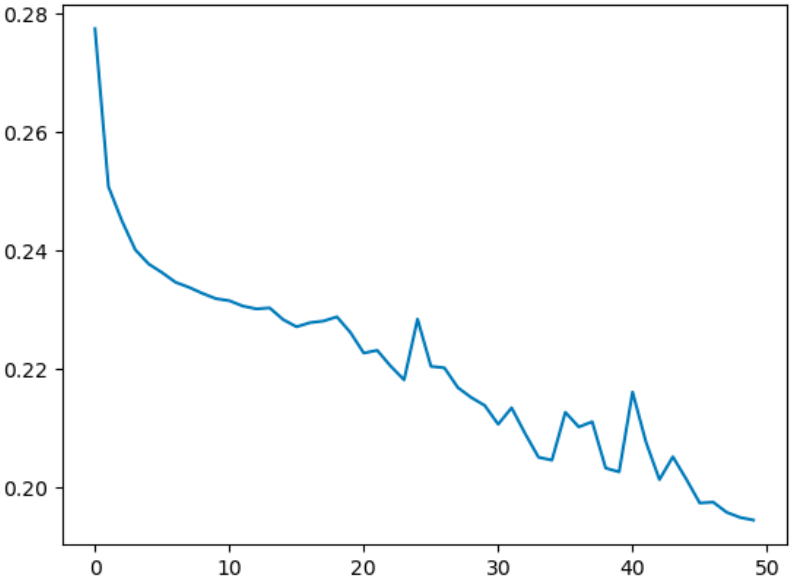

Figure 3. Loss curve during training.

Each feature has a loss, and the total loss is a weighted sum of different losses. Since the loss of pitch is much larger than the loss of step and duration, in order to make the overall loss more uniform from the sum of three different losses, the weight is adjusted for different losses. The results of adjusting

DOI: 10.54254/2753-8818/19/20230512

weights are as follows: the weight of pitch loss is 0.05, the weight of step loss is 1, the weight of duration loss is 1.

As for inference, music generation is mainly based on the starting sequence of provided notes, and the model can generate a note from a series of notes. In order to prevent some notes with high probability in the generated input sequence from being repeatedly selected many times, this work selects them according to the softmax probability distribution.

3. Result

The result of generated music will be demonstrated in this chapter. Given the input sequence, the model makes a prediction to generate the follow notes as demonstrated in Figure 4. It is the first 10 generated notes.

```
generated_notes.head(10)
```

✓ 0.2s

	pitch	step	duration	start	end
0	75	0.070458	0.177557	0.070458	0.248015
1	98	0.373858	0.000000	0.444316	0.444316
2	99	0.341195	0.000000	0.785511	0.785511
3	87	0.342429	0.000000	1.127939	1.127939
4	87	0.314626	0.000000	1.442566	1.442566
5	87	0.316611	0.000000	1.759177	1.759177
6	84	0.316691	0.000000	2.075868	2.075868
7	87	0.305125	0.000000	2.380993	2.380993
8	71	0.316404	0.000000	2.697398	2.697398
9	87	0.246487	0.000000	2.943885	2.943885

Figure 4. Example of generated notes.

Figure 5 demonstrates the entire track of generated music, as could be observed that there are some repeat patterns in the music, which sounds harmonious and smooth.

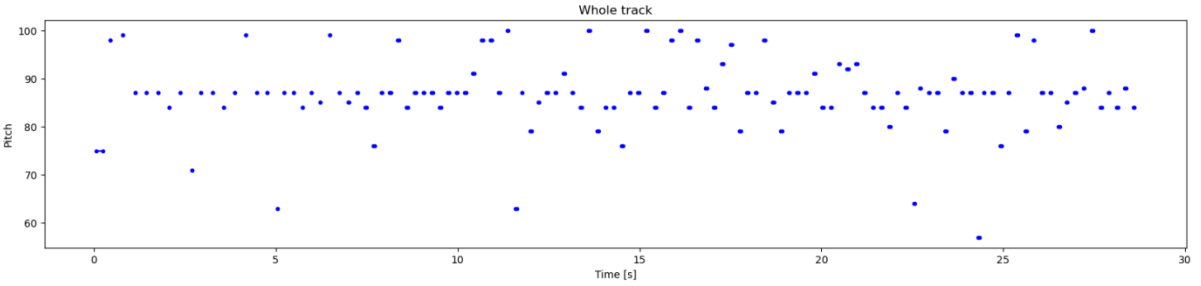

Figure 5. Whole track of generated music.

To further reveal the features of the generated music, the distribution of the three features pitch, step, and duration is shown in Figure 6.

As it could be observed, due to the feedback loop between the output and the input of the model, it tends to generate similar output sequences to reduce the loss, so the features are concentrated in some places. So, the randomness of the generated notes should be adjusted appropriately.

For pitch, the randomness could be increased by increasing temperature in predict next note. temperature is a hyperparameter of LSTMs used to control the randomness of predictions by scaling the logits before applying SoftMax. When using code to adjust the default temperature of 1.0 to 2.0 in next notes, the randomness of the pitch increased.

DOI: 10.54254/2753-8818/19/20230512

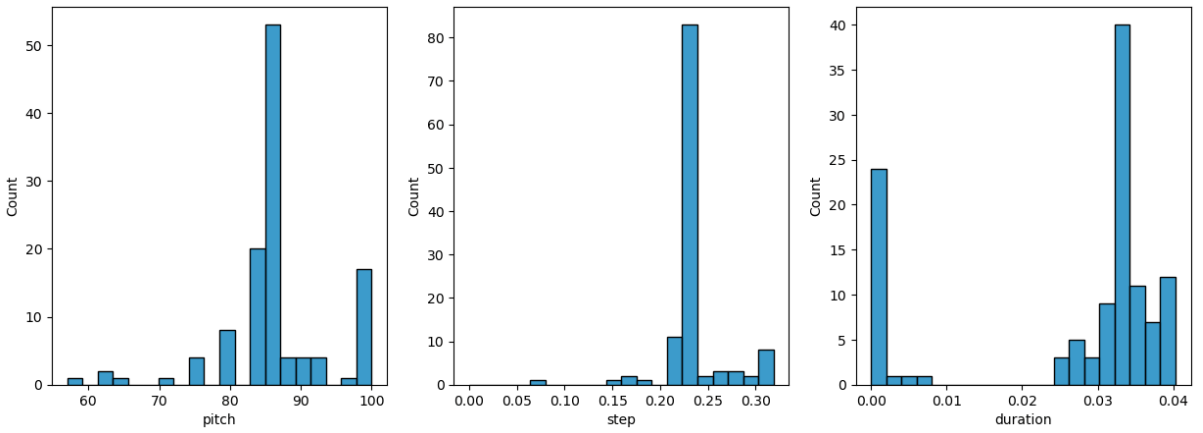

Figure 6. Pitch, step and duration distributions of the generated music.

4. Conclusion

In summary, based on the existing midi files, this project uses RNN to analyze and learn the inherent laws in them, and generates midi files by repeatedly calling the model while the total loss is acceptable. In the future, it would be an interesting topic to explore whether a model could be expanded to support multiple instruments. There is still a lot that can be improved. For example, the dataset used is piano scores, which have multiple tracks. However, the generated result is that the sounds of multiple tracks are clustered in a single track, which is one of the limitations of the current project. Also, the music generation at this time is random, there are no certain rules. The author will continue to explore the field of music generation with these two points and see if the network model can be further adjusted. The current approach is based on the historical sequence and predicts the note at the next moment. In addition, the seqGAN could be used, which can treat the process of sequence data generation as a sequence decision process. As for other ways of generating music, so far, the state-of-the-art model is designed by magenta, an open source project, whose latest method is called music transformer, which is an attention-based neural network that can generate music with long-term coherence.

References

[1] Lundqvist, L. O., Carlsson, F., Hilmersson, P., & Juslin, P. N. (2009). Emotional responses to music: Experience, expression, and physiology. Psychology of music, 37(1), 61-90.

[2] Ritter, S. M., & Ferguson, S. (2017). Happy creativity: Listening to happy music facilitates divergent thinking. PloS one, 12(9), e0182210.

[3] McCormack, J. (1996). Grammar based music composition. Complex systems, 96, 321-336.

[4] Collins, D. (2005). A synthesis process model of creative thinking in music composition. Psychology of music, 33(2), 193-216.

[5] Wang, H., Ma, C., & Zhou, L. (2009, December). A brief review of machine learning and its application. In 2009 international conference on information engineering and computer science, 1-4.

[6] Libbrecht, M. W., & Noble, W. S. (2015). Machine learning applications in genetics and genomics. Nature Reviews Genetics, 16(6), 321-332.

[7] Hsu, C. C., Zhuang, Y. X., & Lee, C. Y. (2020). Deep fake image detection based on pairwise learning. Applied Sciences, 10(1), 370.

[8] Briot, J. P., Hadjeres, G., & Pachet, F. D. (2017). Deep learning techniques for music generation--a survey. arXiv preprint arXiv:1709.01620.

[9] Mao, H. H., Shin, T., & Cottrell, G. (2018). DeepJ: Style-specific music generation. In 2018 IEEE 12th International Conference on Semantic Computing, 377-382.

[10] Medsker, L. R., & Jain, L. C. (2001). Recurrent neural networks. Design and Applications, 5, 64-67.