A Review of Automatic Music Generation Based on Performance RNN

Jiyanbo Cao [1], Jinan Fiaidhi [2], and Maolin Qi [2]

[1]Lakehead University
[2]Affiliation not available

October 30, 2023

Abstract

This paper has reviewed the deep learning techniques which used in music generation. The research was based on *Sageev Oore's* proposed LSTM based recurrent neural network (Performance RNN). We have study the history of automatic music generation, and now we are using a state of the art techniques to achieve this mission. We have conclude the process of making a MIDI file to a structure as input of Performance RNN and the network structure of it.

Keywords—deep learning, audio, music, generation, expressive timing and dynamics

Introduction

Deep learning is a trend topic in recent years. With the developing of deep learning algorithms, it starts to be used from every aspect. The most two commonly used areas are CV and NLP, but it doesn't mean researchers have not use it for other areas, audio is one aspect that they are working on. *Sageev Oore* [1] and his team proposed a Long-short term memory based recurrent neural network to generate music, this algorithm will learn from the input music representation and study the pattern of music and then compose a new musical expression. For deep learning approach that the size of input is really important, if the input size is too big then the training time and memory capacity would be an issue. In order to deal with this problem, MIDI file is used as the raw data type. MIDI is not a normal musical file like mp3, it can be described as music score. A synthesizer will generate music according what's inside a MIDI file. Other reason that using MIDI is music generation is either aiming to create music scores or directly interpreting them, but *Sageev Oore* [1] proposed that in fact jointly predicting the notes and also their expressive timing and dynamics is more valuable. That will be discuss at coming section.

The history of music generation can be call back to the 17^{th} century. People had this game called "*musical dice game* "[5], it is just a basic dice game that the only difference is the result of rolling dice is to pick a pre-composed music options and after several rounds the music options are putting together as a complete music. The first automatic music generation is a very simple game. Now through the development of machine learning, deep learning algorithms, music generation can be carried out with different ways. For our research, we focus on training a machine-learning system which is Performance RNN to generate music. They had a great success on the goal of generate music with both timing and feeling, mention that "given the current state of the art in music generation systems, it is effective to generate the expressive timing and dynamics information concurrently with the music." So, the approach is to directly generate improvised performances rather than creating or interpreting scores.

Related knowladge

Scores

Musical score is the representation of music, it is before sound, but a way to transform this representation into music into sound. Such transformation may be the conversion from digital to analog waves, or the human performance of written score. A musical score shows which notes to play and when to play them relative to each other at Fig.1. The timing in a score is a very strict concept. For example, quarter notes are the same duration as quarter note rests, twice the duration of eighth notes, and so on. But there are no specific seconds for how long a note should last. Music is creative art; musical score is only the base line for the music performance. The mapping from score to music is full of subtlety and complexity.

Fig. 1 A typical musical score

MIDI

MIDI (Musical Instrument Digital Interface) is a technical standard that describes a communication protocol for a wide variety of electronic musical instruments, computers, and related audio devices for playing, editing, and recording music.

A midi file contains two things, Header chunks and Track chunks. Header chunk describing the file format and the number of track chunks. Each track chunks have one header and can contain as many midi commands as possible. Following the header are midi events. These events are identical to the actual data sent and received by MIDI ports on a synth with one addition. A midi event is preceded by a delta-time. A delta time is the number of ticks after which the midi event is to be executed. The number of ticks per quarter note was defined previously in the file header chunk. This delta-time is a variable-length encoded value [2]. And for the MIDI event commands, there are two parts, first 4 bits contain the actual command, and the rest contain the midi channel where the command will be executed. There are 16 midi channels, and 8 midi commands. Commands like *Note off (1000xxxx), Note On (1001xxxx), and Key after-touch (1010xxxx), etc* . There are total 128 notes that can be represent from a table which corresponding the numbers to notes.

Fig. 2 A visualized MIDI file in FL Studio Piano roll. The horizontal axis represents time; the vertical axis represents pitch; each rectangle is a note; and the length of the rectangle corresponds to the duration of the note.

We can think of a score as a highly abstract representation of music, MIDI can be visualized as a piano roll. Figure 2 is an example of the piano roll displayed in FL Studio. Each row corresponds to one of the 128 possible MIDI pitches. Each column corresponds to a uniform time step.

Recurrent Neural Network

Recurrent neural network (RNN) is a generalization of feed-forward neural network that has an internal memory. It performs the same function for every input of data while the output of the current input depends on the past one computation [3, 7]. After producing the output, the output is copied and send back into the recurrent network. In the hidden stage, the new input will be calculated with the previous weighted output then generate new output.

Fig. 3 RNN structure

For sequence leaning, RNN can really do it work. But there are still some limitations and disadvantages. RNN architecture can capture the dependencies in only one direction of its learning target. RNN are not good at capturing long term dependencies and the problem of vanishing gradients may occur in RNN. As in every hidden stage it will calculate pervious output, as the length of the sequence grow, the training time will also grow.

Long short-term memory network

In order to solve the vanishing gradient problem of RNN, a Long Short-Term Memory networks (LSTM) was proposed. The difference between RNN and LSTM are not fundamentally, the optimization is using different functions to compute hidden stage. Memory in LSTM are called cells, it decides what to keep in memory. With more functions that LSTM can add or remove information to the cell state. Those functions are called gates. LSTM has three gates to preserve and discard information at cell state. Forget gate, Input gate, and Output gate.

The first layer is forgetting gate, it decides what information will be thrown away, new input will be computed with previous hidden stage output with a new weight, then using sigmoid function output range of [0,1]. 1 means completely keep this, 0 means get rid of this. The next gate is input gate, it decides what new information will be stored in the cell state. It consists of two layers, which are sigmoid layer using to decide what value to be update, and tanh layer is to create a vector of new candidate values. Next step is to update the previous output with the value we get from forget gate then add the new candidate. To the final step, the output gate decides what going to be output, first run the sigmoid layer to decide what parts of cell state will output then put the cell state through tanh and multiply it by the output of the sigmoid layer.

Fig. 4 LSTM structure

LSTM solve the vanishing gradient problem of RNN and can learn a better relationship between each input. The performance will be better for long sequences input.

Problems in this area

Metric Abstraction : Many compositional systems abstract rhythm in relation to an underlying grid, with metric-based units such as eighth notes and triplets. Often this is further restricted to step sizes at powers of two. Such abstraction is oblivious to many essential musical devices, including, for example, rubato and swing as described in Sect. 1.1.1. We choose a temporal representation based on absolute time intervals between events, rounded to 8 ms.

No Dynamics : Nearly every compositional system represents notes as ON or OFF. This binary representation ignores dynamics, which constitute an essential aspect of how music is perceived. Even if dynamic level were treated a global parameter applied equally to simultaneous notes, this would still defeat the ability of dynamics to differentiate between voices, or to compensate for a dense accompaniment (that is best played quietly) underneath a sparse melody. We allow each note to have its own dynamic level.

Monophony : Some systems only generate monophonic sequences. Admittedly, this is a natural starting point: the need to limit to monophonic output is in this sense entirely understandable. This can work very well for instruments such as voice and violin, where the performer also has sophisticated control beyond quantized pitch and the velocity of the note attack. The perceived quality of monophonic sequences may be inextricably tied to these other dimensions that are difficult to capture and usually absent from MIDI sequences.

Performance Rnn

Figure 5 shows how we use Performance RNN to generate music. The first stage shown in this figure is score composition, which is generating music score. The music score then goes through a synthesizer, and it will be transformed to an audio format that can hear by human listener.

Fig. 5 The process of music generation. The music starting with the composition of a score; that score gets turned into a performance (shown as a MIDI piano roll); that MIDI roll, in turn, gets rendered into sound using a synthesizer, and finally the resulting audio gets perceived as music by a human listener.

In this music generation process, performance RNN act as a composer and a performer combined. It will generate the music score

Input representation

The representation of a MIDI file is converting to a sequence of events form 413 different events.

- 128 Note-On events: represent MIDI pitches. Each one starts a new note.
- 128 Note-off events: represent MIDI pitches. Each one releases a note.
- 125 Time-Shift events: each one moves the time step forward by increments of 8ms up to 1s.
- 32 Velocity events: each one changes the velocity applied to all subsequent notes until next velocity event.

The raw MIDI file will be converted to Note Sequence then to Sequence Examples. The Sequence Examples is a 413-dimensional one hot encoding vector as input and output as well.

Fig. 6 Example of Representation used for PerformanceRNN. The progression illustrates how a MIDI sequence (e.g. shown as a MIDI roll consisting of a short note followed by a longer note) is converted into a sequence of commands (on the right hand side) in our event vocabulary. Note that an arbitrary number of events can in principle occur between two time shifts.

Training process

The first step will be to convert a collection of MIDI or MusicXML files into NoteSequences. In this project we use MAESTRO dataset. NoteSequences are protocol buffers, which is a fast and efficient data format, and easier to work with than MIDI files. See Building your Dataset for instructions on generating a TFRecord file of NoteSequences.

We transform the row data to NoteSequences and to SequenceExamples. The first transformation is to serialize the dataset. You can define how you want your data to be structured once and then you can use special generated source code to easily write and read your structured data to and from a variety of data streams and using a variety of languages. The second transformation is to generate the input of the model.

SequenceExamples are fed into the model during training and evaluation. Each SequenceExample will contain a sequence of inputs and a sequence of labels that represent a performance. SequenceExample then will be fed in to train the model.

Fig. 7 The basic RNN architecture consists of three hidden layers of LSTMs, each layer with 512 cells. The input is a 413-dimensional one-hot vector, as is the target, and the model outputs a categorical distribution over the same dimensionality as well. For generation, the output is sampled stochastically with beam search, while teacher forcing is used for training.

Predicting Pedal

Pedal is commonly used in piano performance. It will extend the piano note when the pedal is being pressed. In the RNN model, we experimented with predicting sustain pedal. We applied Pedal On by directly extending the lengths of the notes: for any notes on during or after a Pedal On signal, we delay their corresponding Note Off events until the next Pedal Off signal. This made it a lot easier for the system to accurately predict a whole set of Note Off events all at once, as well as to predict the corresponding delay preceding this. Doing so may have also freed up resources to focus on better prediction of other events as well. Finally, as one might expect, including pedal made a significant subjective improvement in the quality of the resulting output.

Synthesizer

A synthesizer is a sample library, it has thousands of music instrument samples in it. And in the MIDI file, there is only instructions such as note-on and note-off, intensity of how to reproduce the performance, the synthesizer takes that instructions and use the sound samples in the library to reproduce the audio, turns that into sound that we can hear.

In the original paper, the author used FluidSynth as the synthesizer. FluidSynth is a real-time software synthesizer based on the SoundFont 2. FluidSynth itself does not have a graphical user interface, but due to its powerful API several applications utilize it and it has even found its way onto embedded systems and is used in some mobile apps. In our project, in order to have better visualization and editability, we are using FL Studio to visualize the MIDI files, also used it as the synthesizer.

Data

In the training process of Performance RNN, we want to get a model that can generate expressive performance, model that can generate score with dynamics and more nature timing. In order to achieve that goal, we need proper training data. In this project, we use MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization) [4] dataset, this is a dataset which include over 200 hours of virtuosic piano performances captured with fine alignment (˜3 ms) between note labels and audio waveforms.

Homogeneous

The MAESTRO dataset is homogeneous in a few different ways:

First, the MIDI files in the dataset is all classical music. This will help the outputs to be consisting.

Second, it is all solo instrumental music. The performance RNN model we want to train is to generate solo piano performance, so it makes sense that all the training data is solo piano performance. If the training data includes data that is for two or more instruments, then it no longer makes sense for this training purpose (solo piano performance), some part of the data, the instruments will depends on each other in order to sound harmony. So even if we only use the piano part of the data, it still does not fit well with the training process.

Third, the solo instrument is consistently piano. Different instrument has different styles of performing, therefore, classical composers generally write in a way that is very specific to whichever instrument they are writing for. The classical music scores are closely related to the timbre of that instrument, due to the different catachrestic of the instrument.

Forth, the piano performances were all done by humans. All the MIDI files in the dataset are recorded from human performer during the piano competition. In order to mimic the expressive timing that human performers have, it is much better for the system to learn from performance done by human pianists.

Fifth, all the performances were done by experts. If we wish the system to learn about human performance, that human performance must match the listener's concept of what "human performance" sounds like, which is usually performances by experts. The casual evaluator might find themselves slightly underwhelmed were they to listen to a system that has learned to play like a beginning pianist, even if the system has done so with remarkable fidelity to the dynamic and velocity patterns that occur in that situation.

Realizable

Using a dataset that the solo instrument is piano has other benefits. Synthesizing audio from MIDI can be a challenging problem for some instruments. For example, having velocities and note durations and timing of violin music would not immediately lead to good-sounding violin audio at all. The problems are even

more evident if one considers synthesizing vocals from MIDI. Here, that the piano is a percussive instrument buys us an important benefit: synthesizing piano music from MIDI can sound quite realistic (compared to synthesizing instruments that allow continuous timbral control). Thus, when we generate data, we can properly realize it in audio space and therefore have a good point of comparison. Conversely, capturing the MIDI data of piano playing provides us with a sufficiently rich set of parameters that we can later learn enough in order to be able to render audio. Note that with violin or voice, for example, we would need to capture many more parameters than those typically available in the MIDI protocol in order to get a sufficiently meaningful set of parameters for expressive performance.

Evaluation matrix

Musical expression evaluation it is not quite like basic static data evaluation. There are two different types of machine learning algorithms, for the classification and the regression problem. Simply the outputs are signal value that may means house price, temperature, or categories. But the musical expression, it is not a signal value which may represent the result. For this reason, the evaluation matrix is using log-loss.

Also, *Sageev Oore* [1] and his team used another way to evaluate the generated samples. They compared hand selected statistical features with the same features that from human music performance. They mentioned that the relationship between MIDI pitch number and MIDI velocity the best fit first three orders of polynomials are remarkably similar. That could mean the performance is good. For our research, we only share the generative performance to our class. Then let the audience to decide whether it is a good performance.

Conclution

We have seen the trends of machine learning, now is the turn of deep learning. Deep learning is used in a variety of field, and we can see every sign of it in our daily life. NPL, Computer Vision are not all, we have learned about deep learning in the audio domain which it can be used to compose music. Although, it is kind of like an entertainment. What we should see is the potential of it, if it was the start of the revolution of a new form of art–generate by machine. Our review is based on Sageev Oore, using MIDI file dataset and events that similar to MIDI events as the representation of music. The model they proposed is aiming to not only generate music but also music with expressive timing and dynamics. And the network structure is the LSTM network which is good with dealing sequences data. During our research, we have trained our own dataset also use piano performance. For the feature work, we consider using different types of music as the training set, see what kind of music would generate.

References

1. Oore, S., Simon, I., Dieleman, S. et al. This time with feeling: learning expressive musical performance. Neural Comput & Applic 32, 955–967 (2020). https://doi.org/10.1007/s00521-018-3758-9
2. File format description–(.mid) Standard MIDI File Format. http://faydoc.tripod.com/formats/mid.htm
3. Ah Chung Tsoi, Andre Back, "Discrete time recurrent neural network architectures: A unifying review", Neurocomputing, Volume 15, Issues 3–4, June 1997, Pages 183-223
4. Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang,Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset." In International Conference on Learning Representations, 2019.
5. Boehmer K (1967) Zur Theorie der offenen Form in der neuen Musik. Edition Tonos, Darmstadt
6. Simon, Ian, and Sageev Oore. "Performance rnn: Generating music with expressive timing and dynamics." JMLR: Workshop and Conference Proceedings. Vol. 80. 2017.
7. Walder, Christian, and Dongwoo Kim. "Computer assisted composition with recurrent neural networks." arXiv preprint arXiv:1612.00092 (2016).