Using Deep Learning to Generate Music

Chao Tang, Graduate School of Engineering, University of Tokyo

Abstract—**In this paper, I use two different models to generate music, one is the Google Brain Project-Magenta which uses LSTM-Recurrent Neural Network(RNN) model in tensorflow, another called MusicGenerator, improved the architecture of Magenta to add an encoder/decoder block to the 2 layer LSTM network. For Magenta, a dataset contains 10 midi files and 50 midi files of the classical piano music was used for training. For the second one, a dataset of 50 ragtime music was used for training. We can find the differences of two models and in hope to build a bridge in Python 3, tensorflow and midi music library.**

Magenta generated music link:
https://soundcloud.com/hrnu9kn66u8d/magenta_result1_5songs
https://soundcloud.com/hrnu9kn66u8d/magenta_result2_50songs

MusicGenerator music link:
https://soundcloud.com/hrnu9kn66u8d/musicgenerator

Key words— **deep learning, music generator, recurrent neural network**

1. INTRODUCTION

MUSIC is an art form which represents the creativity of human activities, but here I want to ask the question that can we use deep learning to do something creative, in other words, can we further our own creativity as people. For the development of modern music, there is a role that technology plays, and there is a role that artists play, we need to be carefully understand that interaction. If we look back at electric music, this is exactly how it works, from the early Germany electric band like Kraftwerk, they used synthesizers, drum machines, vocoders as their instruments. So for using deep learning in generating music, we are more like a musician who create new instruments and less like musician who has a high level skill in playing an instrument.

The project Magenta was released by Google Brain team in June, 2016 with two main goals: first it is a research project to advance the state of the art in machine intelligence for music and art generation; second, Magenta is an attempt to build a community of artists, coders and machine learning researchers. It is an open project in github which allows everyone to use this module in generating music also sketching.

The second project is called music generator by Conchy Licultor in github in Sept. 2016. He used a simple RNN model (2 LSTM layers) to formulate the prediction as an 88-binary classification problems: for each note on the keyboard, the network trying to guess if the note is pressed or released. The reason why I also want to try this model is because in Magenta, the documentation is not so friendly, so I want to find a project which like Keras just few lines of code. In magenta, there are too much codes for a simple model, from the model module it is very difficult to identify where is the RNN model and other parameters. Also in MusicGenerator, the writer used a compound architecture-RNN Encoder/Decoder to avoid compiling. The common problem that most model will occur is that when learning from note sequence the model may overfit or compile, so I'm going to talk about a little music theory here.

When we look at a music sheet, we can see that notes are actually related to each other, but a model can only recognize the difference in pitches, if the first note is C and second is B it will predict the same thing as first note is B and second note is C. So model could not recognize the relation between notes, if we inverted the notes it will still output the same prediction. To improve this, we are going to use an entirely different architecture in the Music Generator called encoder and decoder architecture.

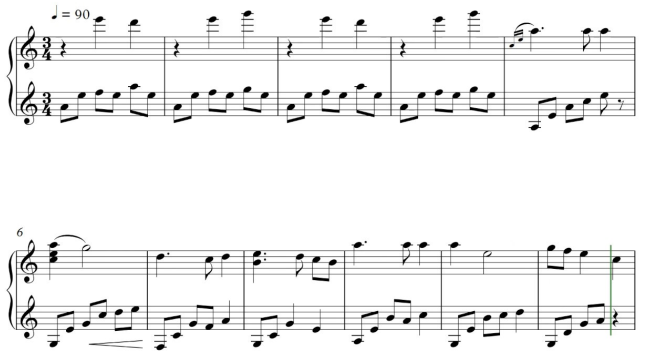

Fig.1 A music sheet

2. RELATED WORKS

The first attempt by anyone to use a computer to compose music was by two American professors at the University of Illinois Urbana-Champaign, Hiller and Isaacson [1]. They programmed the university's lilac computer to compose music and it generated pitches using random numbers. It used the past to determine the probabilities of the future. The first piece was composed in 1957 and was called the lilac suite for string quarter.

Later in 1981, the professor David Cope [2] in University of California began working with algorithmic composition system called Experiments in Musical Intelligence, or Emmy. His idea was inspired by Professor Hiller, but he made Emmy the most famous project for learning from and imitating other composers.

In 1989, after RNN was proposed in generating music beyond those exact subsequences, but RNNs were limited by their short-term coherence. In 2002, Doug Eck [3] updated this approach by using Long Short Term Memory (LSTMs) cells because we are trying to remember long-term notes. So Doug now leads the Magenta team at Google Brain applying his LSTM-based approaches in music generation.

Other project like DeepJazz by Ji-Sung Kim [4], BachBot by Feynman Liang [5] at Cambridge University and MusicGenerator by Conchy Licultor [6] also used LSTM-based model.

On the other hand, Boulanger-Lewandowski [7] who was a PhD student at University of Montreal, working with Professor Yoshua Bengio, now a software engineer at Google, trying to use a recurrent temporal restricted Boltzmann machine (RTRBM) to generate unconstrained polyphonic music.

3. Method

Both magenta and MusicGenerator used TensorFlow as their backend. The dependencies including Python3, Tensorflow, CUDA, Numpy, Mido (midi library), Tqdm (for the nice progression bars), OpenCv, Bazel, scipy, pandas and matplotlib. I ran the codes on the google cloud platform with a Tesla k80 GPU instance to accelerate the calculation process.

3.1 Data

The first part is to choose what to represent music and find available dataset.

3.1.1 Note Representation

For MusicGenerator, the writer chose MIDI file to represent the musical contents. Magenta uses piano roll data to represent the music.

3.1.1.1 MIDI

Musical Instrument Digital Interface (MIDI) is like the digital alphabet for music, it contains a list of messages that tell an electronic device how to generate a certain sound. So it doesn't store actual sound itself, which led to a much smaller file. We are going to extract the stream of notes for both melody and the harmony sequence from midi file, the harmony's chords accompany the melody's single notes. For example, in the Fig.1, the first row is melody's note and the second row is harmony's note. Then we will group them all together by matrix number, so each matrix has its own grouping of chords. We will vectorize these inputs by converting them into binary matrix so we can feed them into our model.

So we see the midi file like an 88-binary note, and we input the note is on or off with note velocity at time *t* to the network. The format contains a status information (channel number) and a MIDI note number (the note's pitch and a velocity (indicates the loudness of the note). For example, a note is played can be represented as <`Note on, 0, 60, 20`> which means "on channel 1, start playing a middle C with velocity of 20". A note off can be represented as <`Note off, 0, 60, 50`> which means "on channel 1, stop playing a middle C with velocity 50". Thus we can get a sequence to get what we want.

However, one of the biggest disadvantage of MIDI files is that it cannot record the multiple notes played at the same time. So if the tracks are more than one, the model will lack the ability to learn from the data.

3.1.1.2 Piano roll

Since the midi files have such drawback I mentioned before, I want to use a more complicated representation format which contains multiple tracks' information at the same time. The piano roll is inspired by automated pianos (see Fig.2). We can see there is a roll of paper with holes punched into it, each hole represents a note control information. The length of the hole indicates the duration of the note, and we can see at the same time there are multiple tracks' information. The piano roll data is the most frequent used in the music representation. The piano roll data can be generated from the source files by transposing each sequence in C major or C minor and sampling frames every eighth note following the beat information present in the MIDI file. The piano roll data can be convert to NoteSequences in building dataset.

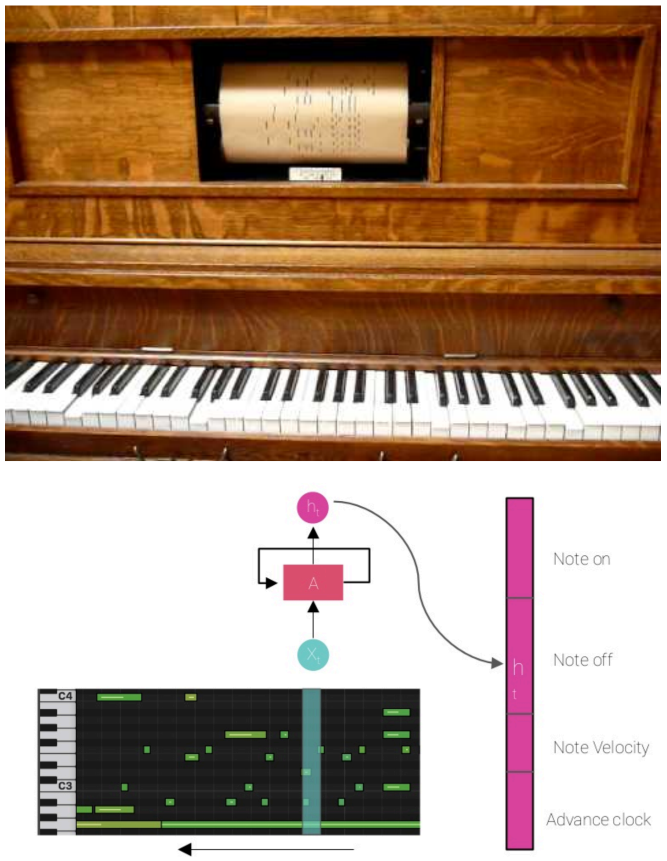

Fig.2 Piano roll data

3.1.2 Datasets

After figuring out the representation format of music, we are trying to find reliable database of MIDI files like MNIST in handwriting identification. MusicNet [8] is a collection of 330 freely-licensed classical music recordings, together with annotated labels indicating the precise time of each note every recording, the instrument that plays each note, and the note's position in the metrical structure of the composition. MusicNet is built on Numpy and Matplotlib for Python users.

The piano roll data can be accessed in the [9] which written by Boulanger-Lewandowski, I choose the Piano-midi.de database. Alternatively, pickled piano-rolls for use with the Python language are also provided.

3.2 Architecture

Magenta encourages coders and artists to involve in this program, in the model module, there are many models such as melody_rnn, sketch_rnn, performance_rnn and so on for different people to do something creative. I chose the basic_rnn model to generate music in Magenta. While MusicGenerator used a compound architecture to add an encoder/decoder to an LSTM network to generate music.

3.2.1 Recurrent Neural Network(RNN)

Recurrent Neural Networks are good at learning sequences and therefore widely used in unsegmented, connected handwriting recognition or speech recognition. The recurrent neural network usually has identical input layer and output layer, as a recurrent network predicts next note, which will also used in next input in an iterative way in order to produce sequences.

In a Machine learning sense, the RNN model for music scores would overfit, which means they would start to memorize these musical scores very quickly, this performance data. A big problem of the RNN model for music scores is that the model would overfit, which means they would start to memorize these musical scores very quickly, also RNN cannot remember long-term memory.

3.2.2 Long Short Term Memory (LSTM)

LSTMs are a special kind of RNN, capable of learning long-term dependencies, were proposed in 1997 by Sepp Hochreiter and Jürgen Schmidhuber [10]. The idea behind LSTM is to secure information in protected memory cells, protected from the standard data flow of the recurrent neural network. It has input gate, forget gate and output gate to decide when to read, forget and write. Therefore, gates are modulated by a weight, thus differentiate from back-propagation and other learning process.

3.2.3 Encoder-Decoder RNN

The idea of encapsulating two identical LSTMs into an autoencoder is a technique to encode a variable length sequence learnt by a recurrent network into another variable length sequence produced by another recurrent network. Because one of the most important thing in music is the special relationship between different notes, so the model should be able to capture the distance.

The LSTM Encoder/ Decoder takes two recurrent networks, the green set of squares and yellow set of squares are two LSTM different recurrent networks, each serves in different purposes. The role of encoder is to count the distance between the notes and compress that information into a single state vector. We are trying to take the first hidden state which is a set of feature vectors that it learned into the decoder. Finally, the decoder will output the next note in the sequence.

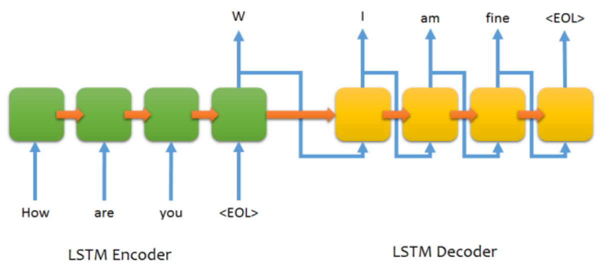

Fig. 3 Encoder/Decoder RNN

4. Experiments and Results

Different datasets were used in two projects.

4.1 Magenta

The midi files were converted into NoteSequences build dataset. NoteSequences are protocol buffers, which is a fast and efficient data format, the dataset was output to `‘/tmp/training_melodies.tfrecord’.` Train the model is run the command as magenta’s github shows. The training loop will run until terminated manually, epoch was set to 20000. After training, performances can be generated using the checkpoint file of trained model.

4.2 MusicGenerator

After setting the environment and preparing the dataset, to train the model, simply run `main.py`. Once trained, we can generate the results with `main.py --test --sample_length 500.`

5. Discussion and Future Works

In this paper, I talked about using deep learning in generating music. These two project can both generate music with very simple method.

The future works may include as below:

(1) Apply Generative Adversarial Networks (GAN) to music generation. GAN was introduced in 2014 by Goodfellow, the conceptual idea is to train simultaneously two networks. GAN

uses unsupervised learning and has been applied in generating a synthetic photograph of a cat that fools the discriminator into accepting it as an actual photograph.

(2) For note representation, maybe try something closer to the musical theory or something closer to the physics (frequency, relative distance between notes).

REFERENCES

[1] Lejaren A. Hiller and Leonard M. Isaacson, Experimental Music: Composition With an Electronic Computer, second edition (New York: McGraw-Hill, 1959): 5–7. Reprinted, Westport, Conn.: Greenwood Press, 1979. ISBN 978-0-313-22158-3.

[2] Cope, D., & Mayer, M. J. (1996). *Experiments in musical intelligence* (Vol. 12). Madison, WI: AR editions.

[3] Eck, D., & Schmidhuber, J. (2002). A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, *103*.

[4] https://deepjazz.io/

[5] http://bachbot.com/

[6] https://github.com/Conchylicultor/MusicGenerator

[7] Boulanger-Lewandowski, N., Bengio, Y., & Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*.

[8] https://homes.cs.washington.edu/~thickstn/musicnet.html

[9] http://www-etud.iro.umontreal.ca/~boulanni/icml2012