Music Generation in Deep Learning

Yuege Li

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Master of Science

of

University College London.

Department of Computer Science

University College London

September 5, 2016

I, Yuege Li, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Music is a kind of art reflecting humans feelings and embodies human's emotion and intelligence. From the aspect of mathematical model, music is a complex sequencial data. With the development of technique, machine can complete a great number of tasks, such as translation, automatic speech recognition, image recognition, natural language processing etc. Whether machine could compose music like human becomes one interesting but difficult topic. The aim of this project is music generation with machine based on the knowledge about deep learning. The previous works in music generation focus on RNN, LSTM and RNN-RBM. Many improved related works are also based on these basic models. These models are introduced, compared and some of them are improved in my work. All of these previous works, no matter the model is simple or complex, music is generated by only one generative models. However, in the previous works of image generation, the image is generated by a sequence of generative models step by step, instead of only one. Inspired by these works in image, I establish a Random Filter-Non Laplacian model that generates music by a sequence of generative models forming a pyramid shape, which is introduced in this thesis in detail. To improve this pyramid model, I work with Jun's group and contribute to a paper.

Acknowledgements

I would like to thank my supervisor, Dr. Jun Wang for his useful comments, guide and support. He gives me interesting and creative ideas, which helps me think deeply and open mind. Additionally, thanks to Yixin Wu, a Ph.D student. He always communicate the details of the project and explan the difficult points for me. Finally, thanks to Ying Wen, a Master research student. Thanks for his kindly explanation to the problems in code and experiment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Music is a kind of art reflecting human's feelings and emotions. Beautiful music helps people calm down and people can enjoy it. Music composition embodies human's intelligence. Recently, teach computers learn arts and let machines know composing music becomes one interesting topic. The topic of this project is about music generation with machine based on deep learning.

Music composition means creating a melody, which is harmonious and feeling comfortable for listeners. Music is a kind of sequence information, for this kind of generation tasks, there are two main points need to be considered. Firstly, in music, whether the melody is sound or not is determined by the sequence of notes. The sequence is important. In other words, one melody composed with the same notes may sound better than another one, due to the different order of the notes. This is similar with the language sentences that is the different sequence of words will effect the whole meaning of a sentence. Another key point is the long term structure. Long term structure represents that some melody patterns repeat many times from beginning to the end. In music, this repeated music patterns are always the main melody representing the theme and motive of the whole music. Additionally, more complex music does not have only one melody line. For majority of music, there are two melody lines or one melody line with one accompaniment chord line occurring simultaneously. This kind of music called polyphonic music, which is much richer than the music with one simple melody line. Therefore, generating polyphonic music becomes one more difficult but interesting topic.

The development of composing music with machine is a developing and continue developing process with huge potential. Throughout the previous research in this area, these works are indeed try to solve the difficulties that mentioned before. For example, at the beginning, Todd [3] uses naïve RNN in order to achieve the time-series in music, next, Eck & Schmidhuber [4] use LSTM to reproduce the global structure of music. More recently, Boulanger-Lewandowski et al. [5] design a RNN-RBM model, which can generate polyphonic music better.

In this project, firstly, I review the previous works and compare some of the models from these previous works, such as RNN and LSTM. Particularly, I focus on the RNN-RBM model [5], for this is one of the most popular work in very few work regarding polyphonic music generation. Related to this mode, there are some improved work. In order to continue improving the RNN-RBM model, I combine a most recently Attention LSTM model [6] with it. Attention LSTM is an improvement work based on LSTM, which can represent long term structure of music better. I combine it with the framework of RNN-RBM model, trying to generate polyphonic music with better global structure. I also compare the result of my work with the previous ones. Beyond that, inspired by Laplacian Pyramid [1] method in dealing with image and the procedure of Laplacian Generative Adversarial Networks (LAPGAN) in sampling image [2], I establish a Random Filter-Non Laplacian (RF-Non Laplacian) model to generate music. Similar to the pyramid process of LAPGAN in generating image, beginning with a small image and after through several generative models can get the full picture, the RF-Non Laplacian generates music from a small piece of melody, through a sequence of generative models to get a long music finally. Finally, to improved this model, I work with Yixin WU and Ying WEN together and contribute for a paper. We establish two improved such pyramid models. These two improved ones, called Moving Average Model, translate note information into frequency and via low-pass filter, moving average, to deal with it instead of using the random filter.

In this thesis, I introduce some background knowledge about generating music in Chapter 2, including the musical terminologies used in this whole thesis, the

previous works in this area and the introduction about the related approaches. In Chapter 3, I introduce all of the approaches used in my work and my own models. The ideas and process about how I form it are also introduced in detail. Chapter 4 is about the whole experiment process, including the data representation, dataset used, evaluation methods and results. Finally, the limitations and future work are discussed in Chapter 5.

Chapter 2

Background

This chapter introduces all of the background about music generation, including relative musical terminologies, previous work and related approaches.

2.1 Musical Terminologies

Due to this project is about generating music, in this thesis, it involves many musical terminologies. In this part, I introduce the meaning of these terminologies, so that readers will not be confused about them.

- Pitch: simply, pitch is the height of a sound. Different pitch makes different sound. In this project, there are 88 different pitches, relating to the 88 keys on the piano respectively. For example, $Do(C)$, $Ri(D)$, $Mi(E)$, $Fa(F)$, $So(G)$, $La(A)$, $Xi(B)$ is seven familiar pitches in one octave. One octave is composed by this seven whole-step, the white keys on the piano, and another five half-step, $\#Do(\flat Ri)$, $\#Ri(\flat Mi)$, $\#Fa(\flat So)$, $\#So(\flat La)$, $\#La(\flat Xi)$, the black keys on the piano. There are 7 octaves on the piano keyboard.

- Duration: duration is the holding time of one pitch. For example, the common half note, quarter note and eighth note represent the holding time of one pitch.

- Note: one note is composed by a pitch and a duration time. In one score, a sign resembling a small tadpole is one note.

- Chord: one chord is composed by more than one notes. These notes occur simultaneously and make a harmony sound. In the piano playing, chords

always present as an accompanying part, played with left hand.

- Melody: melody is composed by a sequence of note.

- Monophonic melody: monophonic melody also means single melody. Simply, in monophonic melody, there is only one note occurring at a time.

- Polyphonic melody: polyphonic melodies are richer than monophonic ones, always have more than one notes occurring simultaneously. For example, polyphonic melody can be composed by two melody lines, or one melody line with different chords, and etc. Some instruments are common used to play polyphonic musics, such as the piano. So that a piano score is always a polyphonic score. Meanwhile, some instruments are suitable used to play monophonic melody, such as the violin. Hence, to accomplish a polyphonic music, there need to be more violins played simultaneously.

2.2 Related Work

Todd [3] was the earliest one using parallel distributed processing (PDP), also called connectionism for note-by-note music composition applications. Note-by-note music composition is using the note at time t as input to predict note at time $t+1$. The benefit of this model is that it is no requirement for the expert musical knowledge, engineers do not need to extract features from musical data. The structure of this network is a three layers' network, input, hidden and output layers, which has a recurrent connection from the output layer to the input layer. Music is a temporal process, the time represented in this network uses the spatial-position representation, which converts the problem of learning music into the problem of learning spatial patterns. The network is trained by backpropagation. For the melody representation, in this network, the melody is represented by a sequence of notes. A pair of value, pitch and duration, represent a note together. This network is able to learn the structure of existing music and then generate new pieces of monophonic melodies from these learned aspects.

Mozer [7] designed a fully connected RNN network, called CONCERT, which

can compose new pieces of monophonic melodies by note-by-note prediction. The melody described in this model is by a sequence of notes, each note represented by a pitch-duration-chord triple. The special part of this work is that the pitch represented in this network is based on a kind of psychological studies of human's perception and cognition to the music [8]. This means that notes should have similar representation if people judge them similar, which is related to the three components, pitch height, Chroma circle and circle of fifths (PHCCCF). Mozard also used a similar way to represent duration based on duration height, 1/3 beat circle and 1/4 circle.

Eck & Schmidhuber [4] was the first one composing music with LSTM and RNN, which can compose blues music. Before this work, RNN was a straightforward way to generate music. However, it is suffered from a lack of global structure. In other word, in this network with LSTM, it can keep track of temporally distant events, so it can learn the global music structure well. Beyond that, data, instead of using the psychological representation, it is represented in a simple local form, which uses one input/target unit per note, with 1 representing on and 0 representing off. There are a range of 12 notes were possible for chords and 13 notes for melodies, and they divided chord notes from melody artificially. They trained chords and melodies in two separate experiments. In the first experiment, only standard 12-bar blues chords were learnt and reproduce a musical chord structure. In the second experiment, it learns melody and generated a melody line based on the chords before. However, the limitation of this work is that due to the fixed number of chords and notes, the output of this model is very limited. Liu & Ramakrishnan [9] designed a framework with LSTM that uses resilient propagation (RProp) to train. Replacing the standard backpropagation, the model shows a better result.

One important RNN-RBM model designed by Boulanger-Lewandowski et al. [5], which can train melody and chords together. So unlike the models before, the music generated by this model is not only with one melody line, instead, it generates polyphonic music. Polyphonic music is that notes can appear at the same time, which cannot be described conveniently by a normal RNN network architecture

designed for multi-class classification task. Because, it is expensive to enumerate all configurations of the predictions. The restricted Boltzmann machine (RBM) can express the log-likelihood of a given configuration by energy function, which enables to deal with this difficult well. In this work, music was represented in the piano-roll method, which is a binary matrix specifying precisely which notes occur at each time step. This model combines the advantage of RNN and RBM. The part of RBM enables to represent high dimensional data and the part of RNN handles the time dependency. There are four datasets, Piano-midi.de, Nottingham, MuseData and JSB chorales used in their work, which are available online now. The log-likelihood (LL) and expected frame-level accuracy (ACC) [10] two values are used to evaluate and make comparison among different models. Based on RNN-RBM model, there are many improved work occurring and shows a better result. These works use the same dataset and evaluation methods as Boulanger-Lewandowski's work, so it is convenient to compare them. The main idea of these works are all similar with the RNN-RBM model.

Based on RNN-RBM model, Goel et al. [11] improved it and designed an RNN-DBN model for polyphonic music generation. Music as a high-dimensional input data, single layer of binary features is not enough to capture the structure of the music. RBMs can be stacked and trained greedily to form Deep Belief Networks (DBN). Therefore, replacing RBM, the multi-layer DBN can represent more complicated distribution for each time step, so that produce richer and more complex melodies. However, due to RNN-type models lack of long-term memory, it cannot model the global structure of the music. LSTM does not have this limitation. Lyu et al. [12] designed a LSTM-RTRBM model, combining RTRBM represents high level data and the advantage of LSTM in capturing long-term structure together to generate polyphonic music. This model can capture the whole structure of the music well. Based on this model, Vohra et al. [13] designed a DBN-LSTM model, which is more good at hierarchical representations. This model shows the better result recently.

All of the works above are based on using current note(s) to predict next

note(s). The same task is performed for each note, and the whole music is generated directly from a random seed. However, in image generations [1][2], things are different. Stimulating human cannot finish a picture at one time, the whole picture does not need to be generated in 'one stroke'. Similarly, in most cases of human compose music, the whole music is starting from several notes, after repeating decoration and modification, and then get a dedicated music in the end. Therefore, inspired by the works in image generation and the reality of human creating music, whether machine can also achieve a similar process that through a sequence of steps to compose a music.

2.3 Related Approaches

In this part, I introduce all of the approaches used in the previous works related to the music generation, such as RNN, RBM, LSTM, Attention LSTM. Meanwhile, inspired by previous works[1][2] in generating images, I design a Pyramid model used in generating music and cooperate with Yixin WU, he improves it to be a more powerful model. So, this two inspired previous image works are also introduced in Section 2.3.5 and 2.3.6.

2.3.1 Recurrent Neural Network

Recurrent Neural Network (RNN) [14] is a common model always used in operating sequence information. The input of RNN is a sequence and the output, in most general cases, is also a sequence. For predicting a sequence, it is not only related to the input sequence, but also the ones predicted before. For example, when predicting a word in a sentence, it is better to consider the words came before it. Let's $x^{(t)}$ is the input at time step t, which is a vector representing one element in a sequence. In my project, it represents the note(s) played at that time step. $h^{(t)}$ is the hidden state and plays a role of 'memory' in the network. It is determined by the previous hidden state and the current input:

$$h^{(t)} = tanh(W_{xh}x^{(t)} + W_{hh}h^{(t-1)}) \tag{2.1}$$

$y^{(t)}$ is the predicted output in this time step, which is only related to the current hidden state:

$$y^{(t)} = Softmax(W_{hy}h^{(t)}) \tag{2.2}$$

where W_{xh}, W_{hh} and W_{hy} are three learnable parameters, sharing across all steps. This reflects the fact of recurrence that RNN performs the same task for every element in the sequence in each time step.

2.3.2 Restricted Boltzmann Machine

Restricted Boltzmann Machine (RBM) is introduced by Smolensky [15], which is used effectively in representing complicated distribution over data. RBM is a two-layer architecture model. The joint probability of the visible variables v and hidden variables u is:

$$P(v,u) = \frac{1}{Z}exp(-b_v^T v - b_u^T u - u^T W v) \tag{2.3}$$

where b_v, b_u and W are three parameters of the model. Z is known as the partition function or normalizing constant. Because RBM has a special bipartite structure, the marginalized probability of v is:

$$P(v) = \frac{1}{Z}e^{-F(v)} \tag{2.4}$$

$$F(v) = -b_v^T v - \sum_i log(1 + e^{b_u + Wv})_i \tag{2.5}$$

Given vector v, the hidden units u_i are conditionally independent of one another, for there is no link among them:

$$P(u_i = 1|v) = sigma(b_u + Wv)_i \tag{2.6}$$

Similarly, given hidden units u, visible units are also conditionally independent:

$$P(v_j = 1|u) = sigma(b_v + W^T u)_j \tag{2.7}$$

Therefore, when given v, u can be obtained by $P(u|v)$ Eqn. 2.6. When obtained u, v can be reproduced by $P(v|u)$ Eqn. 2.7. By adjusting parameters, let the visible

vector v_1 obtained by hidden vector u is equal to the original visible vector v. In this situation, hidden layer u is another representation to the visible vector v, so that hidden layer is the feature of the input data in the visible layer. The process of repeated sampling $u|v$ and $v|u$ is called Gibbs sampling. Learning the parameters, in practice, can use Contrastive Divergence (CD_k) algorithm [16]:

$$\frac{\partial(-logP(v^{(l)}))}{\partial\Theta} \simeq \frac{\partial F(v^{(l)})}{\partial\Theta} - \frac{\partial F(v^{(l)*})}{\partial\Theta} \tag{2.8}$$

where $v^{(l)*}$ obtained from a k-step Gibbs sampling from $v^{(l)}$. Θ represents three parameters b_v, b_u and W.

2.3.3 Long Short Term Memory

The hidden state $h^{(t)}$ is the 'memory' in the RNN, however, in practice, it cannot capture the information from too many steps before. Hence, naïve RNN is not good at long term information captured. In music, long terms structure means not only the short melody stays in key, but global structure follows a theme or chord pattern. Long Short Term Memory (LSTM) is a good way to solve this shortcoming. LSTM is a special kind of RNN, introduced by Hochreiter & Schmidhuber [17]. Let $x^{(t)}$, $h^{(t)}$ and $C^{(t)}$ represent the input, hidden state and cell state respectively, the LSTM model is defined as follow:

$$\begin{aligned}
f^{(t)} &= sigma(W_{xf}x^{(t)} + W_{hf}h^{(t-1)} + b_f) \\
i^{(t)} &= sigma(W_{xi}x^{(t)} + W_{hi}h^{(t-1)} + b_h) \\
C^{(t)} &= f^{(t)} * C^{(t-1)} + i^{(t)} * tanh(W_{xC}x^{(t)} + W_{hC}h^{(t-1)} + b_C) \\
o^{(t)} &= sigma(W_{xo}x^{(t)} + W_{hx}h^{(t-1)} + b_o) \\
h^{(t)} &= o_t * tanh(C^{(t)})
\end{aligned} \tag{2.9}$$

2.3.4 Attention LSTM

Based on LSTM, Attention LSTM [6] aims to improve the LSTM's ability to learn long term structures, producing better music and stories. The cell used in this model is a LSTM. In this model, the output for the current step is connected to the outputs from the previous n steps:

$$u_i^{(t)} = r^T tanh(W_1 h_i' + W_2 C^{(t)})$$

$$a_i^{(t)} = Softmax(u_i^{(t)})$$

$$h^{(t)} = \sum_i a_i^{(t)} h_i' \tag{2.10}$$

where r and W_1, W_2 are learnable parameters. h_i' are the outputs from the previous n steps. $C^{(t)}$ is the cell state of the current step. $u_i^{(t)}$ is an n length vector, each value corresponding to a step before, representing how much attention of each step should receive. $a_i^{(t)}$ is the attention mask in the model, which is created by a softmax to normalize $u_i^{(t)}$. The outputs from previous n steps will multiply with the attention mask and summed to get $h^{(t)}$, which is the new output for the current step.

2.3.5 Laplacian Pyramid

Laplacian pyramid [1] is a kind of image encoding technology, see in Fig. 2.1. There are two operations in dealing with images, downsampling and upsampling. The downsampling operation is getting image I_{k+1} from original image I_k through a low-pass filter. In this process, the resolution and density of the image are all decreased. To be exactly, let $d(*)$ represents downsampling operation, if I_k is a $k \times k$ images, so that $d(I_k)$ obtains a new image I_{k+1} with size $j/2 \times j/2$. Let $u(*)$ be the upsampling process, which will expand the image to be twice size, saying $u(I_{k+1})$ will get another image L_k with $j \times j$ size. After one downsampling and one upsampling operation, the final image L_k is blurry and unclear compared with the original one. So there is a difference between L_k from I_k, and this difference is h_k:

$$h_k = I_k - L_0 = I_k - u(I_{k+1}) = I_k - u(d(I_k)) \tag{2.11}$$

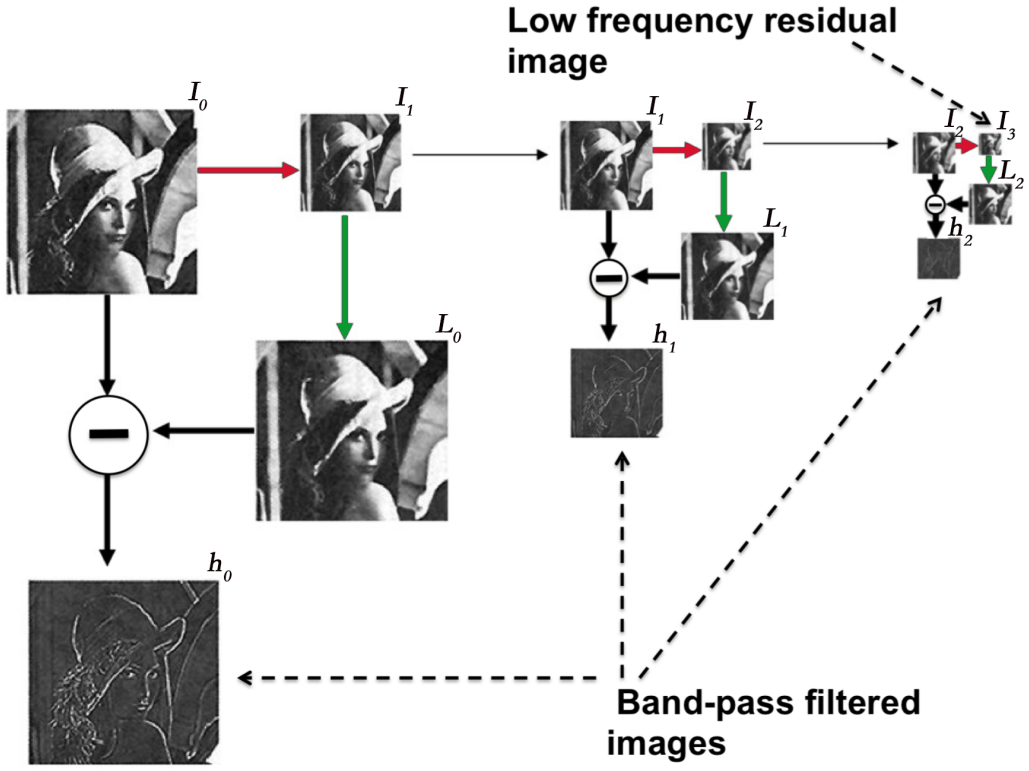

Figure 2.1: Laplacian Pyramid (source from [1])

The coefficients h_k is used to reconstruct the image from I_{k+1} to be I_k:

$$I_k = L_0 + h_k = u(I_{k+1}) + h_k \tag{2.12}$$

2.3.6 Laplacian Generative Adversarial Networks

Laplacian Generative Adversarial Networks (LAPGAN) [2] is one Generative Adversarial Networks combining with the Laplacian pyramid representation, which is a good application to the Laplacian pyramid. The main idea of LAPGAN is that human cannot draw a whole picture with one stroke, so why machine need to generate a picture at one time. Therefore, let's see the sampling process of LAPGAN, in Fig. 2.2. In this process, picture is not generated with 'one stroke', instead through a sequence of generative models $\{G_0, G_1, ..., G_k\}$, starting from a small image and obtaining a whole one. An image $\tilde{I}_{k+1}$ through upsampling operation getting I_k. Generative model G_k generates the coefficient $\tilde{h}_k$ with I_k and a noise sample z_k.

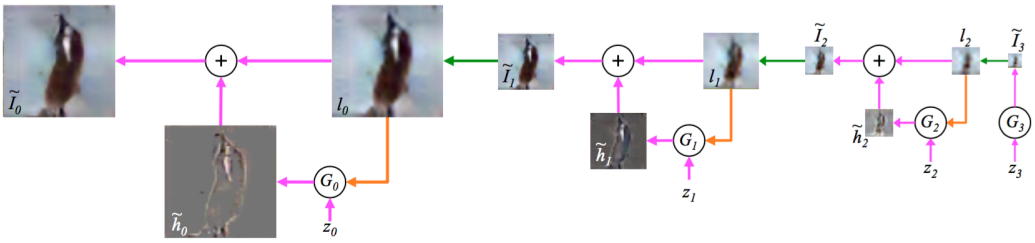

Figure 2.2: The sampling procedure of LAPGAN (source from [2])

With the value $\tilde{h}_k$, the reconstruct an image is akin to Eqn. 2.12:

$$\tilde{I}_k = I_k + \tilde{h}_k = u(\tilde{I}_{k+1}) + G_k(z_k, I_k) = u(\tilde{I}_{k+1}) + G_k(z_k, u(\tilde{I}_{k+1})) \tag{2.13}$$

The process from I_k to $\tilde{I}_k$ is to reconstruct I_k, in order words, making I_k to be clearer. The process from $\tilde{I}_{k+1}$ to I_k is to expand the image. Repeat this process, starting from a small image, through continually expand, clear, expand, clear, ..., finally, it will sample a large picture.

Chapter 3

Approach

In this chapter, I introduce the approaches used in my work. Firstly, RNN-RBM is a quite good model in generating polyphonic music recently. To improve it, I join Attention LSTM model introduced in 2.3.4, a more powerful model in capturing long term structure, with RBM. In Section 3.1, I will explain this model in detail. Additionally, making an analogy to the image works of Laplacian Pyramid and LAPGAN, introduced in 2.3.5 and 2.3.6 respectively, I established the Random Filter-Non Laplacian model which is introduced in 3.2. Based on this new model, I cooperate with Yixin WU who improves this work and make a more powerful one, which is introduced in 3.3. We work together of this part of work and contribute for a paper.

3.1 Attention LSTM-RBM

The Attention LSTM-RBM (AttLSTM-RBM), in Fig. 3.1, is an extension of the generative model RNN-RBM. The idea is that RNN is not so good at learning the whole structure of the music, so I replace the part of RNN with Attention LSTM, which is a more powerful architecture for capturing long term structure, introduced in 2.3.4 in detail. Hence, I establish this AttLSTM-RBM model. The part of Attention LSTM deal with the sequential information, and RBM handles the high dimensional data representation.

The hidden state $h^{(t)}$ is calculated by Attention LSTM (Eqn. 2.10). The biases

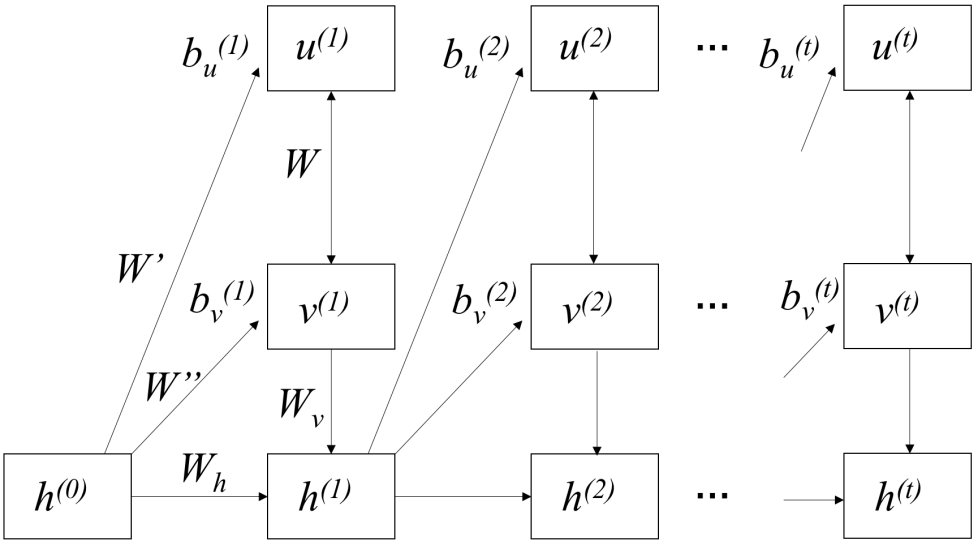

Figure 3.1: Attention LSTM-RBM

of the RBMs (Eqn. 2.3), are depended on $h^{(t-1)}$:

$$b_u^{(t)} = b_u + W' h^{(t-1)} \tag{3.1}$$

$$b_v^{(t)} = b_v + W'' h^{(t-1)} \tag{3.2}$$

where W' and W'' are two parameters.

An iteration of training this model is:

1. Propogate the current value v^t in the Attention LSTM portion, calculating $h^{(t)}$ based on Eqn. 2.10.

2. Calculate the parameters, $b_u^{(t)}$ and $b_v^{(t)}$, using Eqn. 3.1 and Eqn. 3.2, and generate $v^{(t)*}$ using k-step Gibbs sampling.

3. Estimate the log-likelihood gradient, Eqn. 2.8, using CD_k with respect to W, $b_v^{(t)}$ and $b_u^{(t)}$.

4. Propagate the log-likelihood gradient backward through time (BPTT) [18] to obtain the estimated gradient with respect to the Attention LSTM parameters.

3.2 Random Filter-Non Laplacian

Inspired by Burt [1] and Denton's [2] works in image, that image is generated through a sequence of generative models instead of only one, I have an idea formed that whether melody could be generated in a similar way, that from a small piece with several notes to be a whole music through several steps. In this part, I introduce the process of forming this idea in detail.

3.2.1 Downsampling and upsampling

Since the idea is inspired by Laplacian model [1], introduced in Section 2.3.5, so that finding the solution according to this idea, the first question need to be solved is what is the downsampling and upsampling operations in music. For a melody line, it is composed by a sequence of notes (including the rest note, means without note played at that moment). In every sample time, there will be a note played. Based on a constant time (which is always very small) to sample a melody, it will get a sequence of notes representing which note played at that moment. For example, if the sample time is t', getting the sequence of notes are $n^{(0)}$, $n^{(1)}$, $n^{(2)}$ ... $n^{(t)}$. $n^{(t)}$ is a binary vector of 88. 88 represents the 88 keys on the piano, so each element in this vector maps to a pitch in the piano. 0 and 1 represent whether this pitch is played at this time step respectively. Therefore, we can take this sequence of notes representing the melody line.

Back to the image, the result after doing the downsampling operation for an image is that the density and resolution of the image are decreased. Next, doing upsampling operation for this downsampled one, the image will become unclear compared with the original image. Make an analogy to music, Fig. 3.2 shows downsampling and upsampling operations in music, which can get a similar result with the process of image.

The *Downsampling* operation is:

$$m_1 = Downsampling(m_0) \tag{3.3}$$

which shrink the length of the music, but keep the whole musicial structure. The

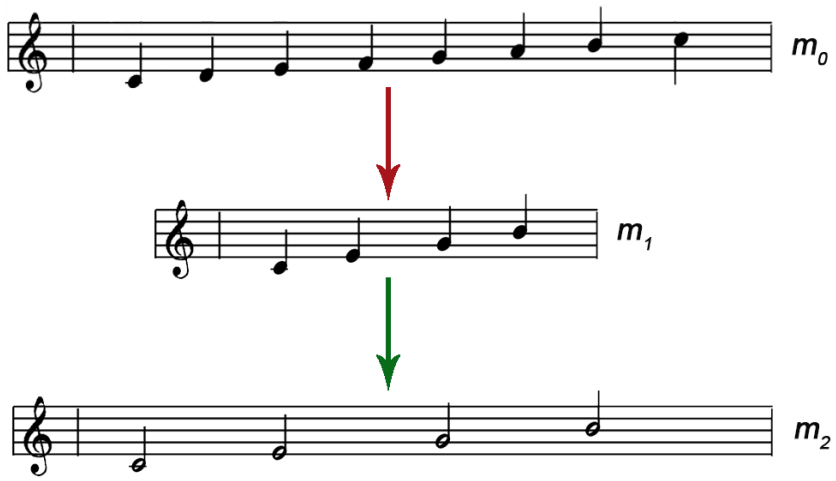

Figure 3.2: *Downsampling* and *Upsampling* in music. The red and green arrows represent *Downsampling* and *Upsampling* operations respectively.

purpose of this process is to extract the meaningful part in the music, and delete subtle decorations and tempo, showing the theme of the music more straightforward and directly. In my work, I use the easiest way to do *Downsampling* that is only keep the notes on the even position and delete the odd ones.

To expand the length of the downsampling melody m_1, recovering it to be the same size of the original melody m_0 is by *Upsampling*:

$$m_2 = Upsampling(m_1) \tag{3.4}$$

The simplest way to do it is prolonging a same holding time for each note in order to recover the length. Compared with the original melody m_0, m_2, which experienced downsampling and upsampling operations, are less richness. But, m_2 is a high level overview of the original melody m_0.

3.2.2 Sampling Procedure

The sampling procedure of the Random Filter-Non Laplacian is akin to the sampling procedure of LAPGAN, composed by a repeated upsampling operation and generation process. The sampling procedure is shown in Fig. 3.3, starting from several notes, continually expand and rich it so that getting a more complete melody finally, forming a pyramid shape. The purpose of this sampling procedure is gen-

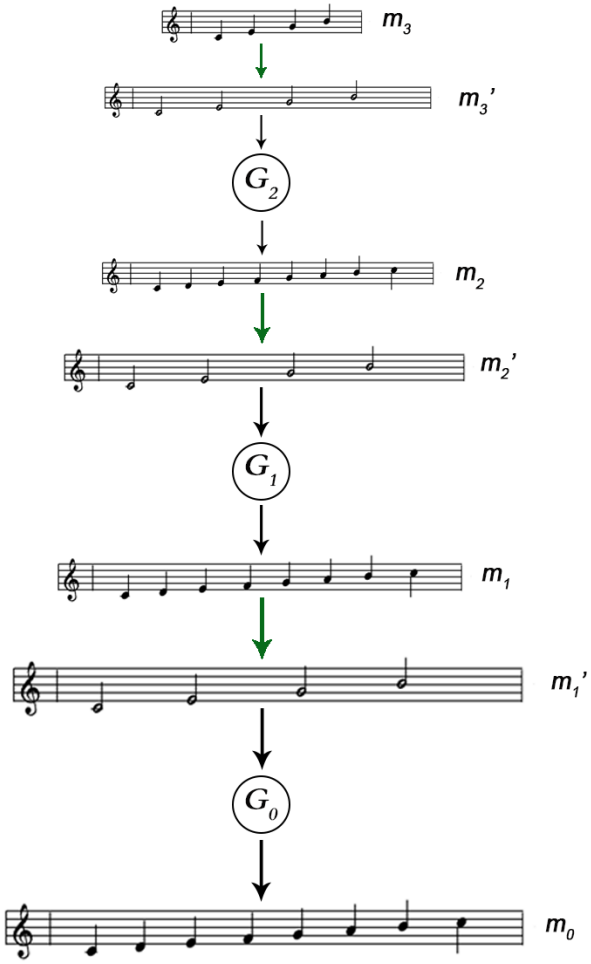

Figure 3.3: Sampling procedure of the Random Filter-Non Laplacian model

erating a full and rich music around several notes through a sequence of expand and reconstruct operations. The process from m_k to m'_{k+1} is via the $Upsampling$ operations (Eqn. 3.4). G_k is generative model.

The generative model G_k in my pyramid model is RNN, which is used to reconstruct melody. The sequence notes $\left\{n^{(0)}, n^{(1)}, ..., n^{(t)}\right\}$ represents the notes in melody m'_{k+1}. The hidden unit $h^{(t)}$ and the output of the generative model $y^{(t)}$ are:

$$h^{(t)} = tanh(W_{nh}n^{(t)} + W_{hh}h^{(t-1)} + b_h) \tag{3.5}$$

$$y^{(t)} = Softmax(W_{yh}h^{(t)} + b_y) \tag{3.6}$$

where $\left\{y^{(0)}, y^{(1)}, ..., y^{(t)}\right\}$ is melody m_k. b_h, W_{nh}, W_{hh}, W_{yh} and b_y are learnable

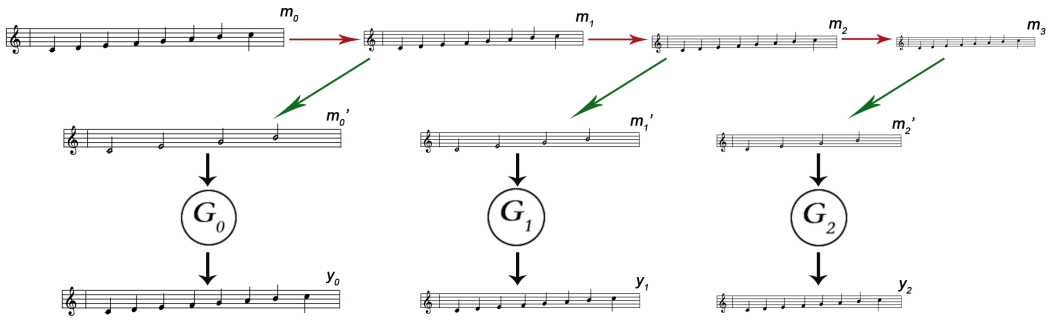

Figure 3.4: Training procedure of the Random Filter-Non Laplacian model. Red arrows represent *Downsampling*, green arrows represent *Upsampling*.

parameters.

Following training, a set of generative RNN models $\{G_0, G_1, ..., G_k\}$ is obtained. Each one can generate a more full of rhythm melody than the input one. Melody m_{k+1} through $Upsampling$ getting m'_{k+1}. Melody m_k is generated by $G_k(m'_{k+1})$, to be richer than m'_{k+1}, but still keep the theme of m_{k+1}. Following is one step sampling:

$$m_k = G_k(m'_{k+1}) = G_k(Upsampling(m_{k+1})) \tag{3.7}$$

3.2.3 Training the Model

The training procedure is shown in Fig. 3.4. Each generative model is trained separately. y_k is generated by generative model G_k based on Eqn. 3.5 and Eqn. 3.6:

$$y_k = G_k(m'_k) = G_k(Upsampling(m_{k+1})) = G_k(Upsampling(Downsampling(m_k))) \tag{3.8}$$

If y_k is equal to m_k, it can be seen that the generative model G_k reconstruct melody m_k from m_{k+1} successfully. The purpose is to minimize the cost C:

$$C = -\frac{1}{T}\sum_{t}^{T} -n^{(t)} log\tilde{n}^{(t)} - (1-n^{(t)}) log(1-\tilde{n}^{(t)}) \tag{3.9}$$

where $\left\{n^{(0)}, n^{(1)}, ..., n^{(t)}\right\}$ is m_k, $\left\{\tilde{n}^{(0)}, \tilde{n}^{(1)}, ..., \tilde{n}^{(t)}\right\}$ is y_k.

3.3 Moving Average Filter Model

The RF-Non Laplacian model in Section 3.2 is dealing with the note information and shrink or expand the length of a melody. Generate melody from several notes and expand it to be a long length melody, and then forming a pyramid shape. But sound is a kind of frequency so that Yixin WU provides an idea that whether music generation could be via dealing with frequency from rough to fine. When the note information change to be frequency, *Downsampling* can be done by low-pass filter similar with the work in image, instead of using random filter. In our work, we focus on using one of the low-pass filter called Moving Average Filter. For image, there are two kinds of methods, one is Laplacian pyramid, another one is Gaussian pyramid. The difference between these two pyramids is that Laplacian pyramid takes the difference between adjacent levels in the Gaussian pyramid. Similarly, we also construct two different Models with moving average filter, one is Moving Average Filter-Laplacian (MAF-Laplacian) introduced in 3.3.1, another one is Moving Average Filter-Non Laplacian (MAF-Non Laplacian) introduced in 3.3.2. In this part of work, I work with Jun's group together and make contribution to a paper.

3.3.1 Moving Average Filter-Laplacian (MAF-Laplacian)

In our work, the input data is in Midi format, which is a kind of data stores note information. Therefore, firstly, we need to translate these note information into frequency. In music, each pitch maps to a unique frequency. The translation is as follow:

$$f(n_i^{(t)}) = (\sqrt[12]{2})^{n_i^{(t)}-49} \times 440Hz \tag{3.10}$$

where $n_i^{(t)}$ is the index of the element with 1 in the 88 dimensional vector $n^{(t)}$, i is from 1 to the maximum number of notes played at the time step t. After translation, the original melody is shown in frequency format, which is represented as m_0.

Due to notes have been translated into frequency, we use low-pass filter to deal with it, more similar to the work in image. In our work, we mainly focus on a

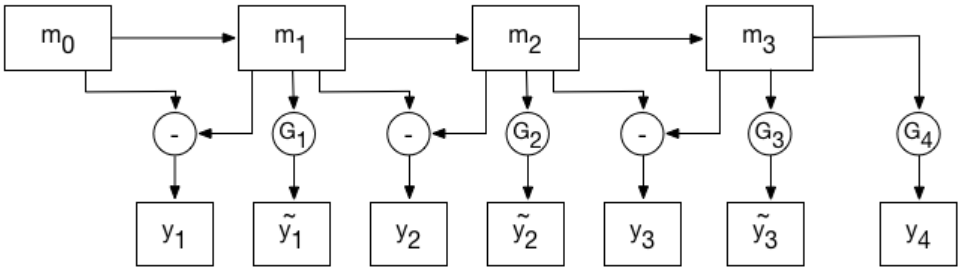

Figure 3.5: Training procedure of the Moving Average Filter-Laplacian model.

method called Moving Average Filter, one of the low-pass filter, to do this step:

$$y^{(t)} = \frac{1}{4}(x^{(t)} + 2x^{(t-1)} + x^{(t-2)}) \tag{3.11}$$

The output of this filter is related to the three successive inputs, where the centre point value is twice as heavily as another two. $x^{(t)}$ is the translated frequency of notes played at the current time t in the frequency matrix m_k. After the filter process, we get a new frequency matrix m_{k+1}:

$$m_{k+1} = MovingAverageFilter(m_k) \tag{3.12}$$

The difference between m_k and m_{k+1} is high-pass y_{k+1}:

$$y_{k+1} = m_k - m_{k+1} \tag{3.13}$$

which is used to reconstruct from m_{k+1} to m_k:

$$m_k = m_{k+1} + y_{k+1} \tag{3.14}$$

this process is called $Refinement$.

The training process of this model is shown in Fig. 3.5 The process from m_{k-1} to m_k is by moving average filter Eqn. 3.11. y_k is the difference between m_{k-1} from m_k. Except the last one, $G_k(k \in (1,2,3))$ is generative model used to generate high-pass $\tilde{y}_k$. The purpose of the generative model G_k is to fit the generated high-pass $\tilde{y}_k$

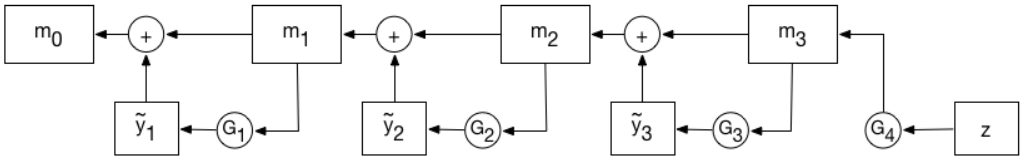

Figure 3.6: Sampling procedure of the Moving Average Filter-Laplacian model.

with the real difference y_k. If $\tilde{y}_k$ is equal to y_k, m_k could be refined to m_{k-1}. The last generative model G_4 shown in the figure is a standard RNN.

The sampling process is shown in Fig. 3.6 where except the last generative model, $G_k(k \in (1,2,3))$ is used to generate high-pass $\tilde{y}_k$, which is used to do *Refinement*. Repeat the generation and refinement process, we can get a whole melody line from rough to fine. The one sampling step is as follow:

$$m_k = m_{k+1} + \tilde{y}_{k+1} = m_{k+1} + G_k(m_{k+1}) \tag{3.15}$$

The last generative model G_4 shown in figure is a standard RNN model with a random input sample z and generates m_3 as the input of the next step. In our real sampling procedure, actually, z is used the result of the first music in the dataset after doing three times of moving average filter.

In the end of the sampling process, we get a frequency matrix m_0, and after that, it is necessary to translate the frequency information back to the notes information:

$$n_i^t = 12log_2(\frac{f_i^t}{440Hz}) + 49 \tag{3.16}$$

where n_i^t is an index in a 88 dimensional vector, and in that place the element is 1. f_i^t is the frequency value of the element in the ith row and tth column of matrix m_0.

3.3.2 Moving Average Filter-Non Laplacian (MAF-Non Laplacian)

As mentioned before, Gaussian Pyramid reduces one step compared with Laplacian Pyramid, that is calculating the difference between two adjacent levels. Similarly, compared with MAF-Laplacian in Section 3.3.1, the MAF-Non Laplacian

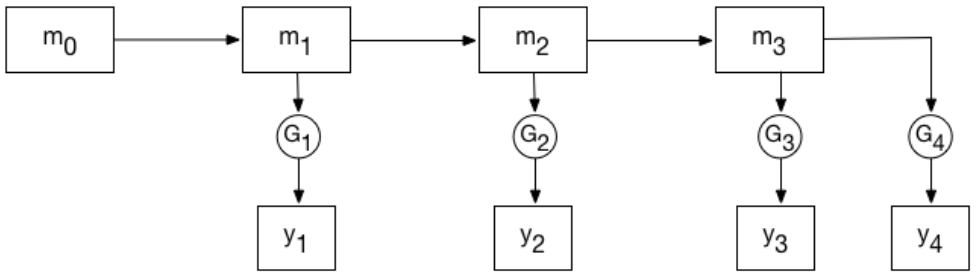

Figure 3.7: Training procedure of the Moving Average Filter-Non Laplacian model.

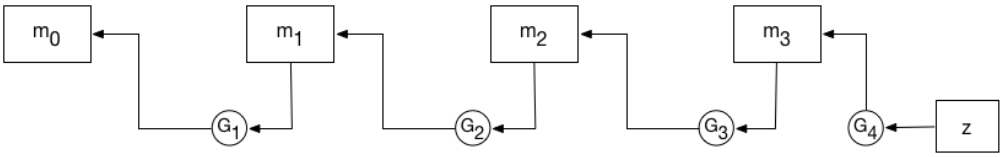

Figure 3.8: Sampling procedure of the Moving Average Filter-Non Laplacian model.

introduced in this part reduces the step of difference, which is also the important difference between two models.

Similar with the MAF-Laplacian, MAF-Non Laplacian, at the beginning, also need to translate note information to frequency with Eqn. 3.10.

The training process of this model is shwon in Fig. 3.7. The process from m_{k-1} to m_k is also done by the low-pass filter, moving-average filter Eqn. 3.11. Except the last one, $G_k(k \in (1,2,3))$ is generative model used to generate y_k. The purpose of the generative model G_k is to fit the y_k with the real one m_{k-1}. If y_k is equal to m_{k-1}, it can be said that the low-passed m_k could be recovered to be m_{k-1} through generative model G_k, which process is called $Refinement$. The last generative model G_4 shown in figure is a standard RNN.

The sampling process is shown in Fig. 3.8. Except the last generative model, $G_k(k \in (1,2,3))$ is used to generate refined m_{k-1} from m_k. Repeat the generation and refinement process, we can get a whole melody line from rough to fine. The one sampling step is as follow:

$$m_k = G_{k+1}(m_{k+1}) \tag{3.17}$$

The last generative model G_4 shown in figure is a standard RNN model with a random input sample z and generates m_3 as the input of the next step. Similar with the situation in the Moving Average Filter-Laplacian, z in this place is also used the result of the first music in the dataset after doing three times of moving average filter.

In the end of the sampling process, m_0 is a frequency matrix. Similar with the Moving Average Filter-Laplacian, it is necessary to translate the frequency information back to the notes information based on Eqn. 3.16.

3.4 Summary

The RF-Non laplacian model and two Moving Average Filter models are all via different filters to smooth the original melody and based on a sequence of generative models to generate music, and then form a pyramid shape, I call all of them Pyramid Model. Focus on these three pyramid models, I make a summary of them here.

	RF-Non Laplacian	MAF-Laplacian	MAF-Non Laplacian
Format of training data	note	frequency	frequency
Filter	random	moving average	moving average
Take difference between adjacent levels (whether it is in Laplacian framework)	no	yes	no

Chapter 4

Experiment

In this chapter, I introduce all of the related work regarding experiment. Data representation, dataset, evaluation methods and the result are all introduced in this chapter.

4.1 Data Representation

Each input and output nodes are used to represent one pitch. In my work, I use 88 visible units representing 88 pitches, which map to the 88 keys on the piano. In each time, if the note is played, the related node is 1.0, otherwise 0.0. This kind of data representation as same as the work of Boulanger-Lewandowski et al. [5].

4.2 Dataset

There are two main datasets used in my work.

The first dataset is Nottingham, which is a collection of 695 polyphonic music of folk tunes. The total duration of the whole dataset is about 8 hours. Meanwhile, I make a little modification for this dataset, that I keep only one melody line for each music in this dataset in order to establish a monophonic music dataset. I use this modified monophonic music dataset to evaluate the effect of some models for a reference.

Another dataset is Piano-midi.de, which includes 88 polyphonic classical piano music. The total duration of the whole dataset is about 7 hours.

The data is in Midi format, which stores the note information instead of sound

signal. Midi files can indicate midi equipment how to do and what to do, such as playing which notes and how much volume is.

4.3 Evaluation

There are different values used to do the evaluation. To the different models, I use different methods to evaluate it. All of these values are introduced in this part.

4.3.1 Log-likelihood

The first one is log-likelihood (LL):

$$LL = -\frac{1}{T}\sum_{t}^{T} -n^{(t)} log\tilde{n}^{(t)} - (1-n^{(t)}) log(1-\tilde{n}^{(t)}) \tag{4.1}$$

where $n^{(t)}$ is the generated note by the model, $\tilde{n}^{(t)}$ is the note in the target melody. This is a negative number, so bigger and closer to 0 is better. Except the two Moving Average Model, this value used in all rest models.

4.3.2 Accuracy

The second evaluation method is expected frame-level accuracy (ACC) [10]:

$$ACC = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FP(t) + FN(t)} \tag{4.2}$$

where $TP(t)$ is 'true positives' represents the number of notes in the generated melody that correctly correspond to the notes in the target melody in time t, $FP(t)$ is 'false positive' represents the number of notes appear in the generated model but do not exist in the target melody at time t, $FN(t)$ is 'false negatives' represents the notes in the target melody but do not be sampled in the generated melody at time t. The bigger the value is the better. This value only used in the polyphonic music generation.

4.3.3 Mean Squared Error

Mean Squared Error (MSE) is used in the two Moving Average Model:

$$MSE = \frac{1}{T}\sum_{t}^{T}(n^{(t)} - \tilde{n}^{(t)})^2 \tag{4.3}$$

where $n^{(t)}$ is the predicted note frequency, $\tilde{n}^{(t)}$ is the real note frequency in the music. This is a positive value, and much smaller is better.

4.3.4 Bleu

Another evaluation method used is Bleu, which is introduced by IBM [19]. This value originally is used to judge the machine translation. The assumption is that if the result translated by machine is closer to human translation, the quality is high. Therefore, the key point is to compare the similarity between the result translated by machine and human. Bleu is to count the number of n-gram shown in both the machine translation and human translation. The larger the value is the better.

The idea of using Bleu to evaluate music generated by machine comes from working with Jun's group together. We use this value to measure the similarity degree between the music in the dataset and the one generated by machine. The Bleu value is calculated as follow:

$$BLEU = BP * exp(\sum_{n=1}^{N} w_n \log p_n) \tag{4.4}$$

where BP is a value regarding to the length. In our project, this value is 1, for the length of generated music is longer than the shortest music in the dataset. $w_n = \frac{1}{N}$ is a weight. The value p_n is:

$$p_n = \frac{\sum C_{clip}(n-gram)}{length-(n-1)} \tag{4.5}$$

$$C_{clip}(n-gram) = min\{C(n-gram), MaxRefC(n-gram)\} \tag{4.6}$$

where $length$ is the length of the music generated by machine. $C(n-gram)$ is the number of n-gram in the generated music, $MaxRefC(n-gram)$ is the maximum

Table 4.1: Log-likelihood(LL) and Bleu in the monophonic music generation

	Nottingham	
	LL	Bleu(%)
RNN	-3.62	39.88
LSTM	-2.66	25.21
RF-Non Laplacian	-1.91	81.19

number of n-gram in one of the music in the dataset, and $C_{clip}(n-gram)$ is the less value of these two.

In my work, I use $N = 4$ to calculate Bleu. All of the Bleu values shown in Result (Section 4.5) are with $N = 10$.

4.4 Implementation

The implementation of RNN-RBM is used code based on Theano [1]. LSTM-RBM is a modification based on the previous work [2]. The model AttLSTM-RBM and Random Filter-Non Laplacian is implemented by myself based on the library Theano. The Moving Average Model is implemented by Ying WEN. The source code is on the Github: https://github.com/YuegeLI/MusicGeneration. All the code can be ran directly. The instruction of the source code is in the file 'README' on the Github, which is also in the Appendix A of this thesis.

4.5 Result

In this part, I present all of the result obtained from different experiments, including both monophonic and polyphonic music generation.

4.5.1 Monophonic Music Generation

Table 4.1 shows the results of three different models, RNN, LSTM and Random Filter-Non Laplacian (RF-Non Laplacian) on the monophonic music dataset. Monophonic music is the music with only one melody line, simply, there are only one note played at one time. The monophonic music dataset is used Nottingham, keeping only one track with one melody line. In my work, I mainly focus on doing

[1] http://deeplearning.net/tutorial/rnnrbm.html
[2] https://github.com/kratarth1203/NeuralNet

Table 4.2: Log-likelihood(LL) and Mean Squared Error(MSE) in the polyphonic music generation

	Nottingham		Piano-midi.de	
	LL	MSE	LL	MSE
RNN	-8.45	-	-11.52	-
LSTM	-8.43	-	-11.39	-
RNN-RBM	-6.78	-	-8.95	-
LSTM-RBM	-7.13	-	-14.21	-
AttLSTM-RBM	-7.22	-	-14.23	-
RF-Non Laplacian	-7.05	-	-9.21	-
GRU	-	3086.37	-	24980.83
MAF-Laplacian	-	561.3	-	9095.70
MAF-Non Laplacian	-	170.76	-	5368.14

polyphonic music generation, so for the Table 4.1 shown the monophonic result, it is only a reference for the model.

From the Table 4.1, we can see that in the situation of monophonic music generation, significantly, the RF-Non Laplacian has the best performance, showing the best result on both log-likelihood (LL) and Bleu values with -1.91 and 81.19 respectively, which are much better than the second one. So that in some degree, it can be said that this new model, RF-Non Laplacian, is more powerful than the simple model RNN and LSTM, at least in monophonic music generation.

4.5.2 Log-likelihood and Mean Squared Error in the Polyphonic Music Generation

Table 4.2 shows the log-likelihood (LL) or Mean Squared Error (MSE) for different models on two polyphonic datasets, Nottingham and Piano-midi.de. The front six models learn the note information, a binary 88 dimensional vector. They predict the probability of each notes will be played. So it can calculate LL value for the probability. However, for the back three models, the training data is frequency. They predict what frequency will be in the next time, so that MSE can be calculated between the predicted frequency and real frequency. Due to there is no probability prediction anymore, it cannot calculate LL value for these three models.

Meanwhile, there is a model GRU in Table 4.2, which is also one type of RNN,

Table 4.3: Accuracy(ACC) and Bleu in the polyphonic music generation

	Nottingham		Piano-midi.de	
	ACC	Bleu	ACC	Bleu
RNN	18.51	29.88	11.91	25.86
LSTM	18.49	12.53	13.69	16.55
RNN-RBM	41.86	17.64	22.80	20.09
LSTM-RBM	43.61	19.99	4.10	61.76
AttLSTM-RBM	42.04	31.84	4.09	59.36
RF-Non Laplacian	22.27	36.29	20.08	34.23
GRU	49.57	24.33	15.08	45.44
MAF-Laplacian	27.36	54.00	27.23	48.28
MAF-Non Laplacian	75.12	61.18	28.23	38.79

similar with LSTM. GRU is a baseline model, doing comparison with Moving Average Filter-Laplacian (MAF-Laplacian) and Moving Average Filter-Non Laplacian (MAF-Non Laplacian).

For the six models in the front of Table 4.2, RNN-RBM gets the best LL value with -6.78 and -8.95 on two dataset respectively, which is followed by RF-Non Laplacian with -7.05 and -9.21. However, it also can be seen that LSTM-RBM and AttLSTM-RBM are not bad on Nottingham, but the result of these two models are not good on Piano-midi.de, even worth than standard RNN and LSTM. From such kind of result we can get that for different models, the effect on different datasets are various. Some models work well on various dataset, such as the model RNN-RBM and RF-Non Laplacian shown in the table. There are also some models that are just suitable for a certain type of data, such as the model LSTM-RBM and AttLSTM-RBM who seems like a better fit on Nottingham this kind style of music generation.

For another three models in the back of Table 4.2, which are dealing with the frequency, it can be seen that these two Moving Average Models are quite good. The value MSE are significantly better than the baseline model GRU on both datasets. MAP-Non Laplacian gets the best MSE with 170.76 and 5368.14 on two datasets respectively, these two values are only 3086.37 and 24980.83 for the model GRU.

4.5.3 Accuracy and Bleu in the Polyphonic Music Generation

Table 4.3 shows the accuracy (ACC) and Bleu value of all nine models training on two polyphonic datasets. Fortunately, these two values can be calculated on all nine models, so we can compare the effect of these nine models together in this table.

From the table we can see that MAF-Non Laplacian gets the both best ACC and Bleu value on Nottingham dataset with 75.12 and 61.18 respectively, which are significantly better than the second one. Meanwhile, on the dataset of Piano-midi.de, it also gets the best ACC value with 28.23. However, the best Bleu value belongs to LSTM-RBM with 61.76.

Comparing the ACC and Bleu value among different models on different dataset, it can be seen the characteristics of these models. I analyze the interesting part of these models in following word.

LSTM-RBM and AttLSTM-RBM shows an interesting result from the table. On the dataset of Nottingham, their ACC value are not bad with 43.61 and 42.04 respectively, however, these value are extremely low on Piano-midi.de. But also in this dataset, Bleu values of these two models are the best one with 61.75 and 59.36 respectively. Their performance on the dataset Piano-midi.de is that they are hard to predict notes will be played at the next time correctly, but they are quite good at generating the music with existing small fragment. However, for another dataset Nottingham, they perform a different result. Hence, from such kind of result and connect their similar performance shown in Table 4.2, it can be seen that LSTM-RBM and AttLSTM-RBM are more sensitive to the dataset. For different dataset they shows totally different result. So the performance of these two models are hard to be predicted. They cannot show a stable result on different dataset.

Additionally, LSTM-RBM and AttLSTM-RBM are both the improved models based on RNN-RBM, and the architecture of these two models are more complex. However, there are not obvious improvement in the results, and even worse. From this result it can be seen that increase the complexity of the model blindly does not necessarily get a good result.

Different from LSTM-RBM and AttLSTM-RBM, three new pyramid models,

RF-Non Laplacian, MAF-Laplacian and MAF-Non Laplacian shows a quite similar result on the two datasets. They do not change significantly due to various dataset, in some degree, showing a stable result. Meanwhile, they have a good balance between two values ACC and Bleu, less to see the situation of one value is good enough but another one is extremely bad than the overall level. So that although the ACC or Bleu of these three models are not always the best one, their result are still keep on the top. More importantly, they show a good stable status on different datasets and the balance between ACC and Bleu compared with the rest models. So this kind of music generation model that generate one song with a sequence of steps are worth to be explored deeply.

Chapter 5

Discussion

In this project, I read many existing works in music generation and get ideas of myself, cooperate with Jun's group to improved my idea. All of these works get some result and can generate new music. In this process of doing the project, there are some limitations that could be improved. If I had more time, there were some points that could be tried in the future. I discuss all of these in this part.

In my work, the music is in Midi format, which only stores the note information instead of the sound signal. Although, in the Improved Pyramid Model we deal with the frequency, it is via a translation step to translate note information into frequency. For music, instead of dealing with Midi file with note information, dealing with sound information directly, such as MP3 format, may also be an interesting work. However, there are much more information in a MP3 format data, such as different timbre played simultaneously, various sound effects, noise, etc., and all of these factors need to be considered carefully which make training process more difficult. For all that it cannot be denied, for music generation, it is worth to try.

The evaluation methods used in my work include log-likelihood, accuracy, mean squared error and Bleu. All of these measurements are compared generated music with the original music in the dataset. Similarity degree with real music in the dataset can judge the quality of the predicted music in some degree. However, obviously, similar with the exiting music is not enough for music composition. Creativity are also important. Therefore, besides making quality judgement for a music generation model from the angle of similarity with the exiting music, it is necessary

to measure it from the angle of music itself. For example, evaluate whether the chord is harmonious, whether the melody line is beautiful, etc. If this evaluation could be shown in numerical form, it will be a great push for music evaluation. However, for this kind of work, rich music knowledge cannot be ignored.

Due to there is not enough time, I only focus on two datasets, with the same style music in one dataset, to compare different models. However, it is undeniable that different dataset will have different effects with different models. If I had more time, I would continue test the models on more different datasets, try to generate music in different style and evaluate the effects.

In the Moving Average Filter Model introduced in 3.3 which is cooperate with Jun's group, there is one important step that smooth the melody with low-pass filter. Through learning the melodies before and after passing filter to achieve generating music from rough to fine. In our work, we only focus on the method called moving-average filter. However, there are various low-pass filters dealing with frequency. Filters are closely related to the learning effect of the model, so that in the future, it is necessary to use different low-pass filter and evaluate the effects of them.

Additionally, for such kind of pyramid model doing in my work, it generates music with a sequence of generative model instead of one generative model. For these generative model, we use the same type of model, such a sequence of RNN or a sequence of LSTM. However, focus on different step of generation, it is possible to use different type of model in different step. It is also possible to use different training methods and different learning rate to train different step. All of these work can be tried in the future.

For music generation with machine, there already have many existing works, and people still continue working hard in this area. I consider that there are many space of improvement in this area. Different models, methods, ideas, etc. waiting people to explore. The music can be more beautiful composed by machine. Music composition by machine is an interesting area, more importantly it is related to whether machine can learn to make art creation as human.

Appendix A

Instruction of Running the Source Code

The source code of this project is on the Github: https://github.com/YuegeLI/MusicGeneration. All the source code can be ran directly.

The fold 'data' includes all the data used in my project.

The fold 'model' includes the model RNN, LSTM, RNN-RBM, LSTM-RBM, AttLSTM-RBM and RF-Non Laplacian. Just run them directly. The generated music will be in the 'sample' fold.

The model GRU, MAF-Laplacian and MAF-Non Laplacian are in the fold 'MVA'. Use python notebook to run the code directly.

The code in the fold 'bleu' are used to calculate the bleu value. Just copy the generated music in this fold and run the code 'bleu.ipynb' with python notebook directly.

Bibliography

[1] Peter Burt and Edward Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on communications*, 31(4):532–540, 1983.

[2] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[3] Peter M Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43, 1989.

[4] Douglas Eck and Juergen Schmidhuber. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 103, 2002.

[5] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

[6] Elliot Waite. Generating long-term structure in songs and stories. `https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn/`.

[7] Michael C Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 6(2-3):247–280, 1994.

[8] Roger N Shepard. Geometrical approximations to the structure of musical pitch. *Psychological review*, 89(4):305, 1982.

[9] I Liu, Bhiksha Ramakrishnan, et al. Bach in 2014: Music composition with recurrent neural network. *arXiv preprint arXiv:1412.3191*, 2014.

[10] Mert Bay, Andreas F Ehmann, and J Stephen Downie. Evaluation of multiple-f0 estimation and tracking systems. In *ISMIR*, pages 315–320, 2009.

[11] Kratarth Goel, Raunaq Vohra, and JK Sahoo. Polyphonic music generation by modeling temporal dependencies using a rnn-dbn. In *International Conference on Artificial Neural Networks*, pages 217–224. Springer, 2014.

[12] Qi Lyu, Zhiyong Wu, and Jun Zhu. Polyphonic music modelling with lstm-rtrbm. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 991–994. ACM, 2015.

[13] Raunaq Vohra, Kratarth Goel, and JK Sahoo. Modeling temporal dependencies in data using a dbn-lstm. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–4. IEEE, 2015.

[14] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[15] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document, 1986.

[16] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[19] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.