An Automatic Music Arrangement System Using Machine Learning

A Thesis submitted to the Department of Computer Science and Communications Engineering,

the Graduate School of Fundamental Science and Engineering of Waseda University

in Partial Fulfillment of the Requirements for the Degree of Master of Engineering

Submission Date: July 18th, 2020

Tianyi GAO

(5118FG19-0)

Advisor: Prof. Hiroshi Watanabe

Research guidance: Research on Audiovisual Information Processing

Contents

1. Introduction

1.1 Abstract

Nowadays, **Machine learning (ML)** can be found everywhere in our life. There are many applications of machine learning, such as message filtering, speech processing and computer vision. Machine learning is a branch of Artificial Intelligence (AI). It is the study of computer algorithms that improve automatically learning through experience [1]. **Deep learning** is a specific kind of machine learning, which performs great on Natural Language Processing, Self-Driving Cars, Computer Vision and many other areas.

A **generative adversarial network** (**GAN**) [2] is a class of frameworks performs great on image generation, which is also a significant part of Deep Learning. In a GAN, usually Two neural networks contest with each other in a game. Given a training set, GAN learns to generate new data with the same statistics as the training set. Using GAN, researchers got plenty of achievements on different areas, especially on image generation. However, on music generation area, GAN still have a long way to go.

In most of the researches, GAN generates there target from noise. Nevertheless, some research generates pictures from picture. For example, Pix2Pix [3] generate pictures from scratch, Cycle-GAN [4] generate pictures from another group of pictures, and Star-GAN [5] generates pictures from several other groups of pictures. We notice that the picture generated by GAN usually have more details than before. Then we came up an idea: Can we use GAN to make a music arrangement system, which gives our input music more details.

Thus, in our research, we provide a system to arrange a music. In the first step of our system, we transfer the input music into **MIDI** file. Then, we train a GAN model (we use Cycle-GAN here) to

transfer between input music and arranged music. We can see the effect of the automatic music arrangement system by the output MIDI file. We also confirmed the difference by groups of hearing tests.

1.2 Motivation

Recently, more and more people are getting interested in music arrangement, aiming to make their own music. People are getting more and more interested in music. Not only listening to music, but also, sometimes, making music by themselves. For example, when a man is taking a shower in his bathroom, a rhythm comes up in his mind. He thinks it is beautiful and he recorded it right after he comes out from his bathroom. What to do next? To make the rhythm into a music, music arrangement is needed.

However, for freshman, music arrangement is too difficult to learn. Even some of the people cannot play any instrument. Most of the people stop their dream here. Maybe a brilliant music is also killed because the people who made it do not know how to arrange it.

Consequently, an automatic music arrangement system may help this kind of people to arrange their music easier. With the well-developed machine learning technique nowadays, making such a system becomes possible. In our research, we provide such a system to arrange a simple music. When a rhythm comes up in your mind, you can make a music in a second.

1.3 Problem Statement

For freshman, there are two main problems on music arrangement:

The first problem is to transfer their idea into a score, which can be considered as MIDI file here. To solve this problem, In the first step of our system, we transfer the input music into **MIDI** file. We used a model called Onset and Frame here, which will be introduced in Part 2.

The second problem is to arrange the music. We need more details in our music, but we do not want to change the main rhythm made by ourselves. So, in the second part of our job, we train a Cycle-GAN model to transfer between input music and arranged music.

To train the Cycle-GAN model, we need a dataset with 2 groups of music, including arranged music and un-arranged music. We cannot find such a dataset because the arranged music and un-arranged music we need should be in pair. As a result, we made the dataset by ourselves. This is the most time-cost problem in our research.

The last problem is the evaluation. We also cannot find an evaluation method for our arranged music. At last we confirmed the difference by groups of hearing tests, which is used in most of the research about music generation.

1.4 System Overview

To solve the problems mentioned in the previous part, the system has a series of experiments. Figure 1 shows the system in this research.

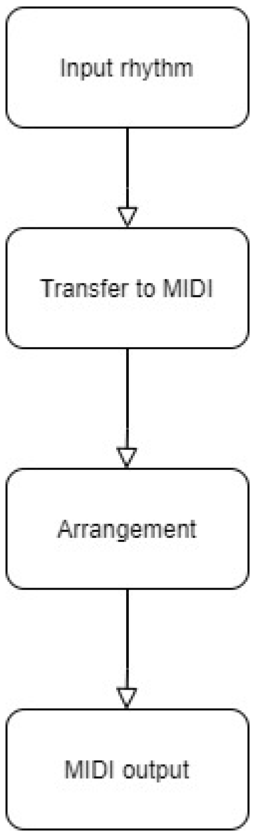

Figure 1 System Overview

For the “Transfer to MIDI” part, we used a structure called Onsets and Frames, which is the state-of-the-art model for automatic polyphonic piano music transcription.

And for the Arrangement part we trained a Cycle-GAN model. For the Generator of Cycle-GAN we use U-Net structure. And for the Discriminator we use Resnet structure.

1.5 Outline

The outline of this thesis is as follows:

Chapter 1:

An introduction of this research, including abstract, motivation, problem statement, system overview and outline.

Chapter 2:

Background of this research, including MIDI, Onsets and Frames, and Cycle-GAN. we talk about the tools and methods we use in our research in this chapter.

Chapter 3:

Proposed method of this research, including pre-processing, arrangement model and data collection. In this chapter, we talk about the proposed methods of our research, which should be used in our experiment.

Chapter 4:

Experiment of this research, including working environment, dataset, pre-processing, training, result and evaluation. In this chapter we talk about the experiment based on the method in chapter 3 and the solutions for some coming problems during the experiment.

Chapter 5:

Conclusion and future work of this research. In this chapter we have a brief conclusion about our research, and we talk about the work can be done in the future.

2. Background

To make our system, we need to do a lot of paper research before our work. Machine learning has been researched for decades since the term Machine Learning was coined in 1959 by Arthur Samuel. Neural Network also has been researched for years. For music arrangement, it has even hundreds of years history. We are standing on the shoulders of giants.

In this section, we talk about the tools and methods we use in our research, including MIDI, Onsets and Frames and Cycle-GAN.

2.1 MIDI

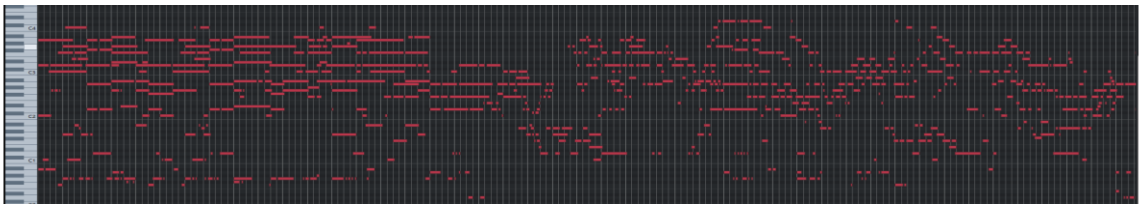

Figure 2 A MIDI file for a classic piano music

MIDI file introduces contents that indicate how do musicians work [6]. Simply speaking, MIDI file stores what instrument does the user play, when does the user play and which key or string does the user play, by which we can use simple arrays to store a complex music. A popular music saving in MIDI file usually takes under 1 megabyte, sometimes even under 100 kilobytes. For some classic piano music, it takes only around 20 kilobytes. Figure 2 shows a MIDI file for a classic piano music, it takes only 32 kilobytes. The height of each line means the key of the note, can be understood as the keys on the piano we play. The length of the lines indicates how long the note keeps, can be known as how long we played the key.

2.2 Onsets and Frames

Onsets and Frames is a model for automatic polyphonic piano music transcription made by Google Brain Team. Using this model, we can convert raw recordings of solo piano performances into MIDI [7].

The input should be a record of solo piano music, or other instruments, even humans voice. At first, the input can be understood as a spectrogram, and the result is a MIDI file transferred from the input spectrogram. Figure 3 shows an example, the transferred MIDI file is showed in figure 4. Which has a little bit mistakes because of the noise from the original music.

This work achieves a new state of the art [11] in predict pitch onset events and then using those predictions to condition framewise pitch predictions.

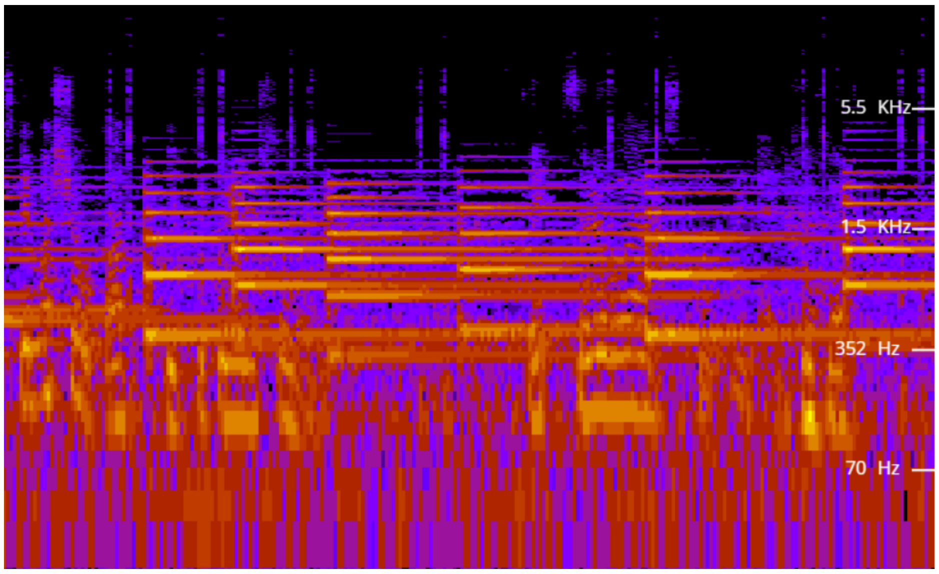

Figure 3 Piano music recorded in our lab

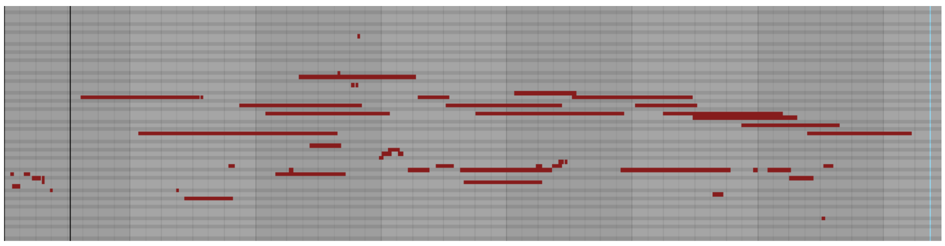

Figure 4 The MIDI file transferred from figure 3

Before Onsets and Frames, Frame-only LSTM was the previous state-of-the-art. Onsets and Frames is also a method improves from the Frame-only LSTM. Figure 5 shows the structure of Frame-only LSTM.

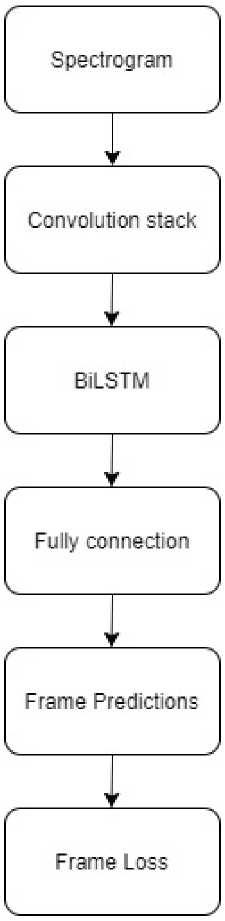

Figure 5 Frame-LSTM

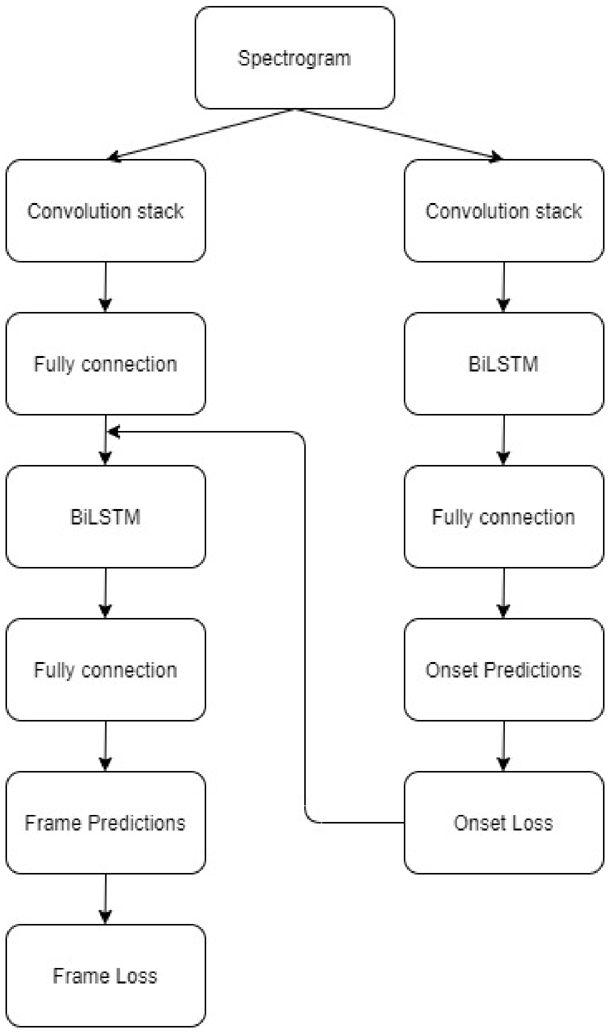

Figure 6 Onsets and Frames

The reason Onsets and Frames performs better is because they split the task into two neural networks: One network is trained for onset frames detection. Another one, same as the Frame-only LSTM, is trained for the detection of every frame.

Onset frames is the first few frames of every note, which means the Onset network only care about the beginning of each note, but not the whole note. Usually the beginning part is the strongest. Experiments indicate that by separating out the onset detection task, this two-network structure performs much better than the previous model. The structure of Onsets and Frames is showed in figure 6.

The comparation between Frame-LSTM and Onsets and Frames:

Table 1 Inception score between Frame-LSTM and Onsets and Frames

	Inception score
Onsets and Frames	50.22
Frame-LSTM	27.89

2.3 Cycle-GAN

Before the introduction of Cycle-GAN, we need to understand several Concepts. Include GAN, CNN, and the structure we used for our Generator and Discriminator.

2.3.1 GAN

GAN [2] is a method proposed by Goodfellow et al. Two neural networks contest with each other in a game, making the generated target statistically indistinguishable from the real one. The two networks in GAN model called Generator and Discriminator. In the game, generator tries to generate a fake target which is as same as possible with the Ground truth. Discriminator tries to find out the generated target is fake or not. After training, the Generator will finally generate a target close to the real one. Figure 7 shows the structure of GAN [2, 15].

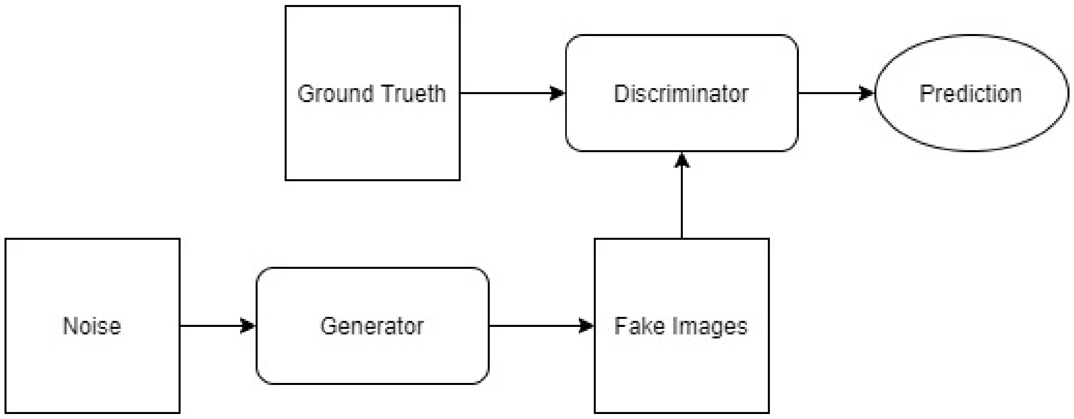

Figure 7 The structure of GAN

Generator is a neural network with input as latent variable z and output as image. On the other hand, Discriminator is a neural network whose input is image and output is the prediction of the input image (true/false). The images of the dataset and the images generated by the Generator are alternately input to the Discriminator, and the Discriminator determines whether the images are derived from the dataset or the generator (true/false). Discriminator learns to make judgment correct, and Generator learns to let Discriminator judge that the generated image as an image derived from the dataset. By successfully training the Generator and Discriminator, the Generator will be able to generate an image as same as the image of the dataset, and the Discriminator will be able to determine whether the input image is from the dataset more accurately. GAN obtains the optimal Generator and Discriminator by solving the minimax problem of the value function V (G, D) expressed by equation (2.1).

$$\min_{G} \max_{D} V(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_{z}(z)}[\log(1 - D(G(z))] \tag{2.1}$$

D(x) represents the probability that the input x of the Discriminator comes from the dataset, and G(z) represents the sample generated by the Generator from z. In addition, $p_{data}(x)$ represents the probability distribution of the data set, and $p_z(z)$ represents the probability distribution of z.

2.3.2 CNN

CNN (Convolutional Neural Network) is a class of deep neural networks which has decades of history. CNN on computer vision was developed from a research to recognize hand-written ZIP Code numbers in 1989, by Yann LeCun et al [8]. In 2004, it is proved that neural networks can be greatly accelerated on GPUs. Research of K. S. Oh and K. Jung shows that their implementation was 20 times faster than an equivalent implementation on CPU [9]. Therefore, the first GPU implementation of a CNN was described in 2006. Deep learning theory was proposed at the same year. From 2006, Convolutional Neural Networks has received attention and has been developed with the update of numerical computing equipment. Since Alex-Net comes in 2012 [10], complex convolutional neural networks supported by GPU computing clusters have repeatedly become the winning algorithm of ImageNet Large Scale Visual Recognition Challenge in several years [12].

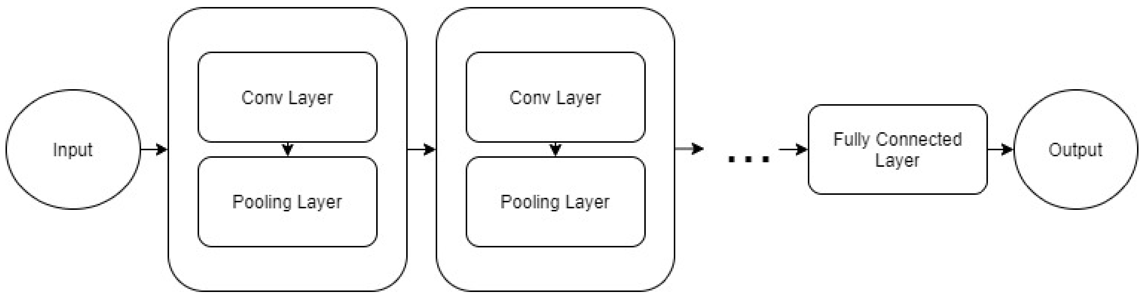

Figure 8 Structure of CNN

Figure 8 is a structure of CNN. The structure can be changed in different research, to feat different situations. There are several layers inside the structure, each of them has their own applications.

a. Convolutional Layer

Convolutional Layer (Conv Layer) consists of filters and activation functions. The general hyperparameters to be set include the number of the filters, size of the filters, step size of the filters, and whether the padding is "valid" or "same". Of course, it also includes what activation function to choose.

Figure 9 shows a working Convolutional Layer, the map in the left is the input image, and the map in the middle is the filter here. The size of the filter is 3 x 3 here. In the convolution, the filter multiplies the parts in the input image and get a result in each calculation. Step size of the filters means how long the filter goes between each calculation.

For example, when the step is 2, the size of our filter is 2 x 2, the size of our input image is 5 x 5. If we do the convolution, we cannot get the feature in the last queue. In this situation, we need to use padding. Padding means add line around the image, with “0” in each pixel. After padding, we make the size of the input image become 6 x 6 from 5 x 5, the convolution can be done well. Every pixel can be contained.

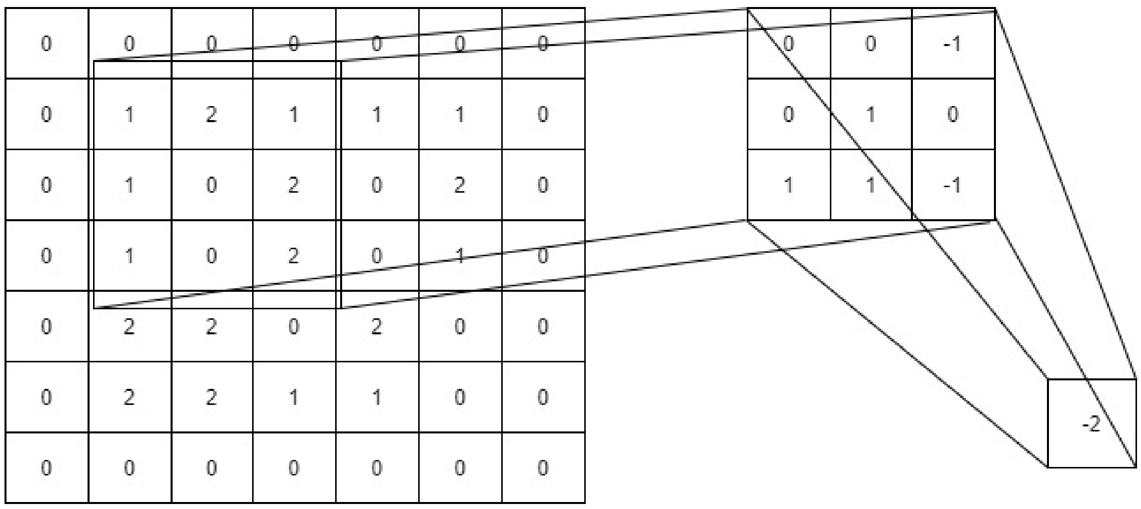

Figure 9 Progress of Convolutional Layer

b. Pooling Layer

In pooling layer there are no parameters here that we need to learn, because the parameters here are all set, either Max-pooling or Average-pooling. The hyperparameters that need to be specified include Max or average, window size and step size.

Usually, we use Max-pooling more, and generally take a filter with a size of (2,2) and a step size of 2. Therefore, after pooling, the input length and width will be reduced by 2 times, and the channels will not change.

Figure 10 shows a working pooling layer using Max-Pooling, with a 2 x 2 window size and a step size of 2. The right map is the input image of this layer. The right map is the result. In the map after pooling, each of the block means a result in the window. We are using Max-Pooling here, so each of the result indicate the Max number in the window. Take the first window as an example, there are four pixels with "1, 3, 7, 9" in them. The result should be the max number in the window, which is 9 in this situation. If we take Average-Pooling here, the result in the first window should be the average number in it, which is "5".

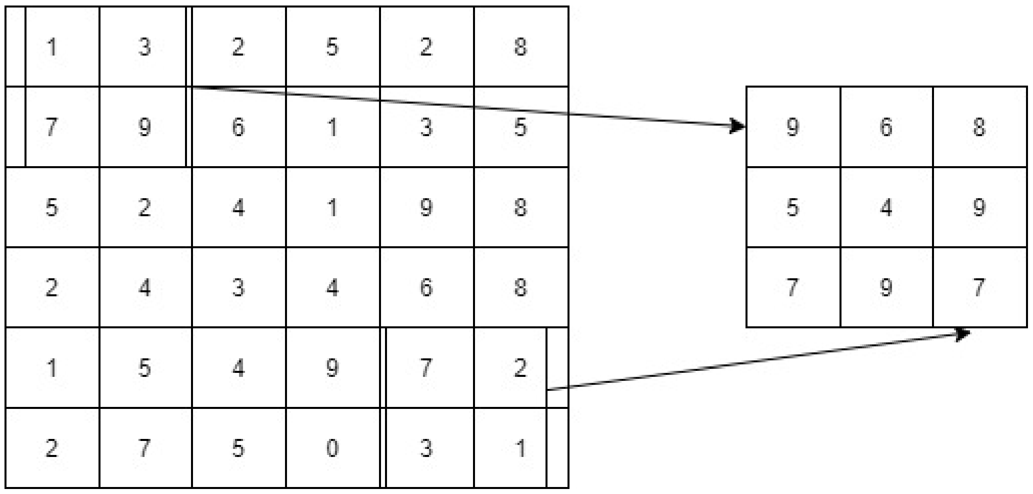

Figure 10 Progress of Pooling Layer

c. Fully Connected Layer

All neurons between the two layers have weight to connect. Usually the fully connected layer is at the tail of the convolutional neural network. It is the same as the connection method of traditional neural network neurons. The hyperparameters to be specified here are the number of neurons and the activation function.

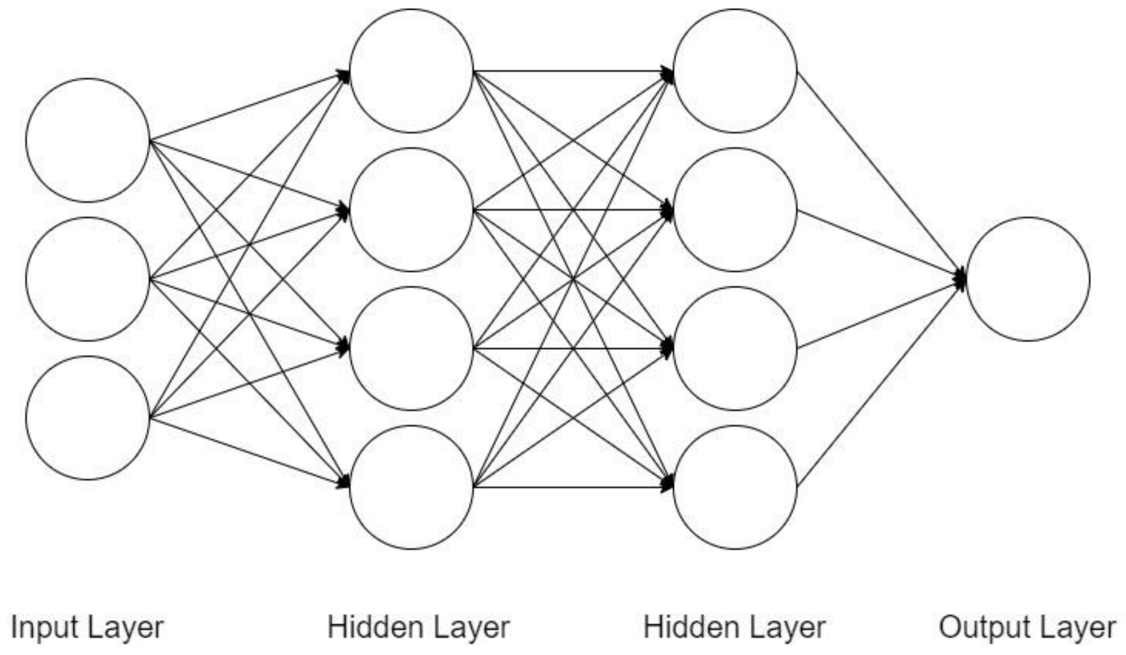

Figure 11 Progress of Fully Connected Layer

2.3.3 ResNet

We know that the deeper the network, the more information we can get, and we can also get better feature. However, according to experiments, as the network deepens, the optimization effect becomes worse, and the accuracy of test data and training data decreases. This is because the deepening of the network will cause the problem of gradient explosion and gradient disappearance.

In order to train deeper networks to better, a network structure ResNet is proposed in 2015 [13], which was the winning algorithm of ImageNet Large Scale Visual Recognition Challenge 2015. ResNet is a residual network, we can understand it as a sub-network, this sub-network can form a deep network after stacking. Figure 12 shows the structure of a ResNet block.

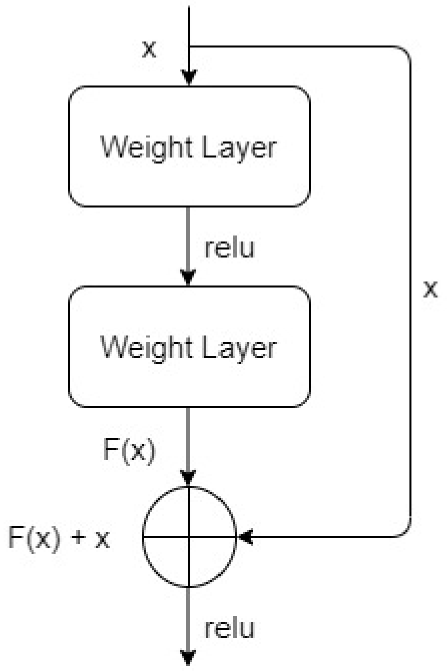

Figure 12 ResNet block

Here x is the feature before the residual block, and F(x) is the function of the two weight layers.

The output can be written as formula 2.2.

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{2.2}$$

$\mathcal{F}(\mathbf{x}, \{W_i\})$ can be written as:

$$\mathcal{F} = W_2\sigma(W_1\mathbf{x}) \tag{2.3}$$

Where W_1 is the weight of weight layer 1 and W_2 is the weight of weight layer 2. σ is the activation function, which should be Relu here.

The same as the original Cycle-GAN paper, we use ResNet for our generator structure.

2.3.4 Cycle-GAN

Traditional GAN is unidirectional, which means we can just generate from one domain to another but cannot go back. Cycle-GAN is essentially two GANs working like a mirror, forming a ring network. The two GANs share two generators, each with a discriminator, that is, a total of two discriminators and two generators [4].

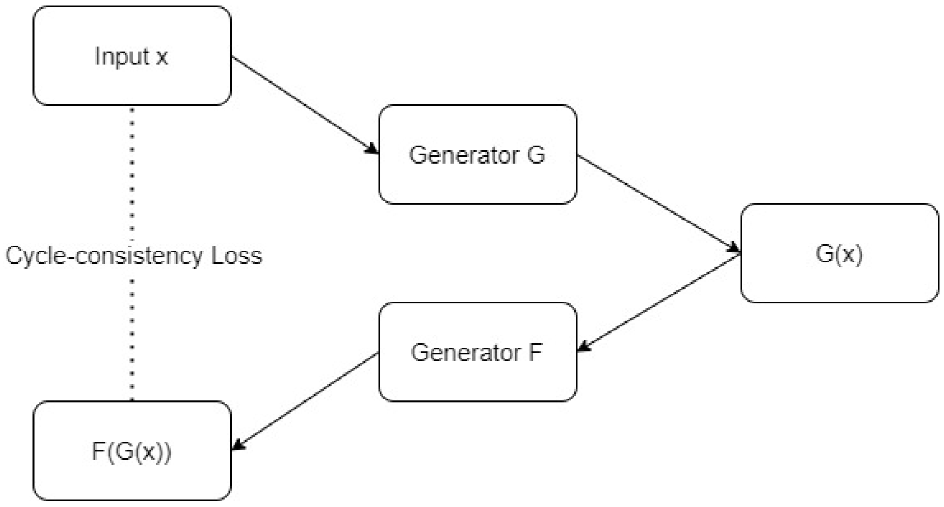

Figure 13 Structure of Cycle-GAN (1)

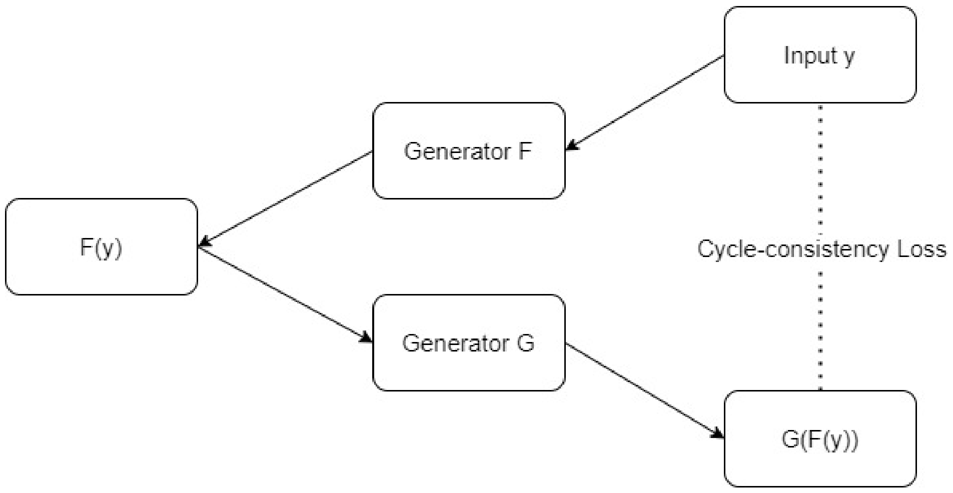

Figure 14 Structure of Cycle-GAN (2)

Figure 13 and 14 shows the structure of Cycle-GAN. In the structure, x and y are input images from two different dataset. In the first step, x is transferred to the domain of y by Generator G and y is transferred to the domain of x by Generator F. The results are shown as G(x) in figure 13 and F(y) in figure 14. At that point, there is a discriminator and we got an output just like a normal GAN. Here comes the second step, which is the most significant point in Cycle-GAN. In the second step, we take G(x) and F(y) as a new input of their Generator each other, which means G(x) is transferred to the domain of x by Generator F, and F(y) is transferred to the domain of y by Generator G. The results are shown as F(G(x)) in figure 13 and G(F(y)) in figure 14. We use Cycle-consistency Loss to evaluate whether the system works well, which indicates the difference between x and F(G(x)) and the difference between y and F(G(y)) [4, 14]. The function is like the follows:

$$\mathcal{L}_{\text{cyc}}(G,F) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\| F(G(x)) - x \|_1] + \mathbb{E}_{y\sim p_{\text{data}}(y)}[\| G(F(y)) - y \|_1] \qquad (2.4)$$

Totally, the loss function of Cycle-GAN should be:

$$\mathcal{L}(G,F,D_X,D_Y) = \mathcal{L}_{\text{GAN}}(G,D_Y,X,Y) + \mathcal{L}_{\text{GAN}}(F,D_X,Y,X) + \lambda\mathcal{L}_{\text{cyc}}(G,F) \qquad (2.5)$$

Where $\mathcal{L}_{\text{GAN}}(G,D_Y,X,Y)$ and $\mathcal{L}_{\text{GAN}}(F,D_X,Y,X)$ are two loss functions same as the normal GAN, which are used on the two Discriminators. $\mathcal{L}_{\text{cyc}}(G,F)$ is the Cycle-consistency loss and λ controls the relative importance of the two objectives.

3. Proposed Method

In Chapter 1, we already talked about the problems we need to solve and had a brief overview of our system. In this chapter, we are going to talk about the proposed method of our research, including pre-processing, arrangement model and data collection.

3.1 Pre-processing

In Chapter 1 (Figure 1), we introduced the flowchart of our system. First, we need to get the input before we process it, which should be a record of rhythm. The recording equipment we used is AT2020 Cardioid Condenser Microphone from Audio-Technica.

Figure 15 Recording Equipment

The recorded sample is in MPEG-4 [16] format. We use the Onsets and Frames (Chapter 2.2) function to transfer the sample into MIDI file. If the sample is clear enough, we can get a nice result, which means the MIDI file is nearly totally the same as we want. However, sometimes there are some noise. In this situation, we delete some of the wrong part in the MIDI file manually, to make the MIDI file clear.

3.2 Arrangement model

We use Cycle-GAN (Chapter 2.3) for our arrangement model. When we arrange a music, we want it adds some details to the input music and keeps the main rhythm, but not totally change the music. In the music style transfer research of Sumu Zhao et al [17]., we find out that they add a loss function to check whether the transferred music can still be understood as the original music. We also add a loss here to check whether the arranged music still have the rhythm of the input music.

$$L_{D_{G,O}} = \| D_{G,O}(x) - 1 \|_2 + \| D_{G,O}(G(x)) \|_2 \tag{3.1}$$

Where $D_{G,O}$ can be understood as a new generator to find out whether the Generated music is from the Original music.

3.2.1 Generator

Although the team who makes the Cycle-GAN used ResNet for their Generator, we also want to try other structure in our research. So, we try not only ResNet, but also U-net [18]. U-net is a popular structure for generators. A lot of researchers use U-net in their research to generate image. We want to know whether it performs better in our research.

When we use ResNet as our Generator, we also try to change the numbers of the layers. In ResNet, we changed the numbers of the Residual Blocks, to see the difference of our arranged music. The structure is shown in figure 16.

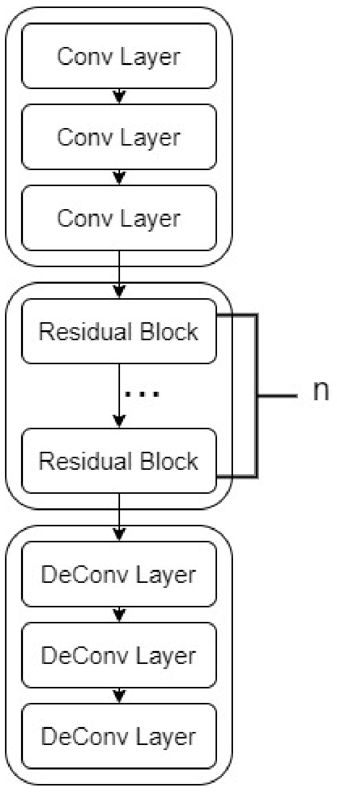

Figure 16 Generator Structure

Where n is the number of residual blocks, Conv layer is convolutional layer introduced in chapter 2 (Chapter 2.3.2). DeConv layer is Deconvolutional layer, which is a layer has the opposite function with convolutional layer.

3.2.2 Discriminator

For the Discriminator, we use the same structure with the Cycle-GAN group. It has 5 convolutional layers. Figure 17 shows the structure of the discriminator.

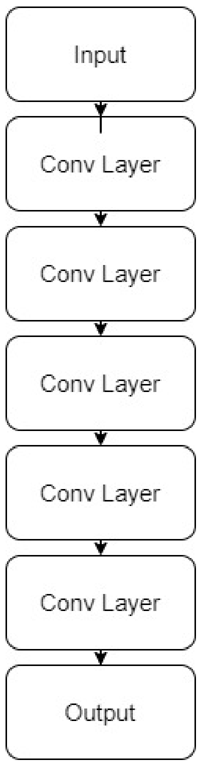

Figure 17 Discriminator

3.3 Data collection

We need a dataset with two groups of audios: The first group of audios should be music with only main rhythm, without chord and reharmonization. Another group of audios should be arranged music, with chord and reharmonization. The problem is that, we want the two groups of music to be in pairs. One rhythm should have an arranged version in another group of data, and an arranged music need to have their main rhythm in another group of data.

4. Experiment

In the previous chapter (Chapter 3), we talked about the proposed method of our research. Besides the problems we introduce in chapter 1 (Chapter 1.3), more problem is found during our experiment. In this chapter we are going to talk about the experiment based on the method in chapter 3 and the solutions for coming problems.

4.1 Working Environment

Most of our work is done on the computer in our lab. The working environment is shown in table 2.

Table 2 Working Environment

OS	Windows 10 Pro
RAM	16GB
CPU	$\text{Intel}^{\circledR}\text{Core}^{TM}\text{i7} - 8750\text{H CPU @ 2.20GHz}$
GPU	NVIDIA GeForce GTX 1080

4.2 Dataset

In chapter 3.3 we talked about the data collection method. We need a dataset with two groups of audios, and we want the two groups of audios to be in pairs. After days of paper research, we did not find such a dataset which can fit our needs. Therefore, we collect data by ourselves. We find MIDI music from opensource websites [21, 22, 23], and manually catch the main rhythm out to make a pair of data. Figure 18 and 19 show an example of our data.

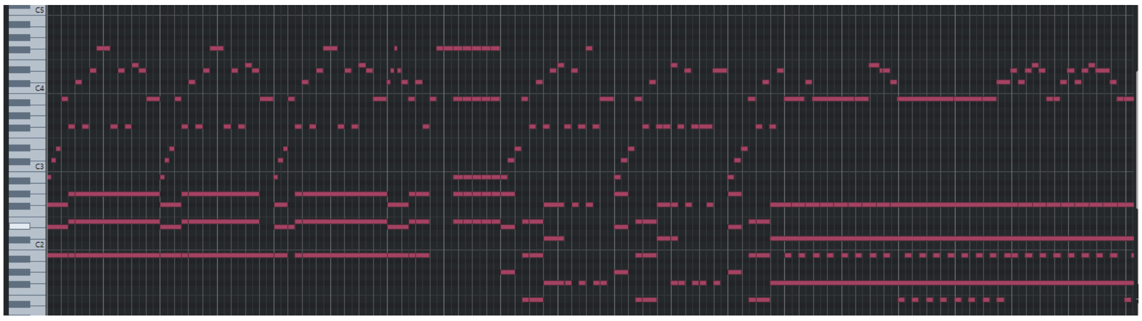

Figure 18 A data in the arranged group

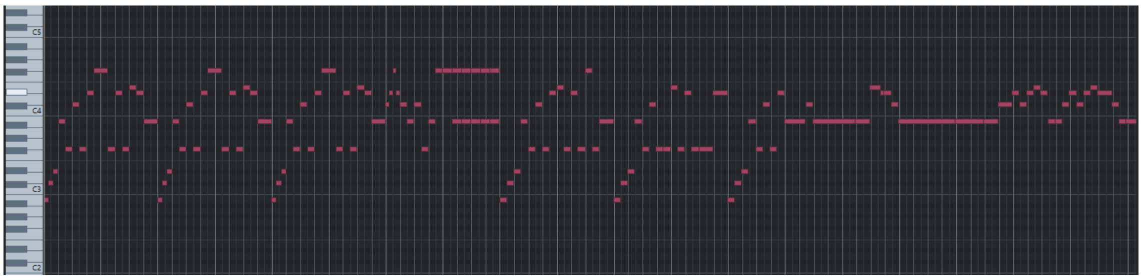

Figure 19 The data with only main rhythm

As a dataset, the data are too long and too large. Thus, we cut the data into dozens of small ones with 16 seconds for each data. As a result, we got a dataset like table 3 shows.

Table 3 Dataset

Dataset	Music	data
Arranged Music	28	454
Main Rhythm	28	454

4.3 Pre-processing

When we use Cycle-GAN to generate images, the input and output is considered as matrix. Usually we use a package in Python called Numpy [19] to deal with matrix. In image processing, we can use matrix to present the contents in an image because image is assembled by pixel, which is just like a matrix.

However, MIDI file is not assembled by pixel. Before we use the MIDI data to train our model, we need to do some pre-process on the MIDI files, aiming to transfer MIDI files into Numpy files. Here, we use a Python package called pretty-midi [20] to transfer MIDI files into matrix.

4.4 Training

In chapter 3 we mentioned that we try not only ResNet, but also U-net. We want to know which one performs better in our system. After training with different layers of ResNet and U-net, we find out that when we use U-net, the system nearly does not fit. The Cycle-consistency loss keeps high and the results was very noisy. We are not clear with the reason, but we think possibly it is because our dataset is too small for the training. However, using ResNet can get a result much better than using U-net. We tried different numbers of layers in our generator, including 8, 10 and 12. Table 4 shows the structure, the activation function is Relu [24, 25].

Table 4 Structure of Generator

Layers	Kernel size	Channel
Conv layer	7 x 7	64
Conv layer	3 x 3	128
Conv layer	3 x 3	256
ResNet	3 x 3	256
…		
ResNet	3 x 3	256
Deconv layer	3 x 3	128
Deconv layer	3 x 3	64
Deconv layer	7 x 7	1

When we use 8 layers of ResNet, the fitting rate is lower than 10 layers and 12 layers. When we use 10 layers of ResNet, the fitting rate is higher. However, when we use 12 layers of ResNet, the fitting rate is not higher than 10 layers.

After we hear the results and compared between different layers. We find out that 12 layers ResNet have better reharmonization and clearer chord (Clearer than the other two results, but still not perfect). Nevertheless, although the result is not a good music for us, 8 layers ResNet gives us a music which have more details. As a result, we choose 10 layers ResNet as our generator structure.

For the discriminator, we simply used the same structure with the original Cycle-GAN model. The structure is as follows (the activation function is Leaky Relu [24, 25]):

Table 5 Structure of Discriminator

Layers	Kernel size	Channel
Conv layer	3 x 3	64
Conv layer	3 x 3	128
Conv layer	3 x 3	256
Conv layer	3 x 3	512
Conv layer	1 x 1	1

4.5 Result

In our system, the result should have two steps:

Step 1 have a spectrogram input (Figure 20), and a MIDI output with the recorded rhythm (Figure 21).

Step 2 take the output of step 1 as an input, and an arranged music in MIDI file as an output (Figure 22).

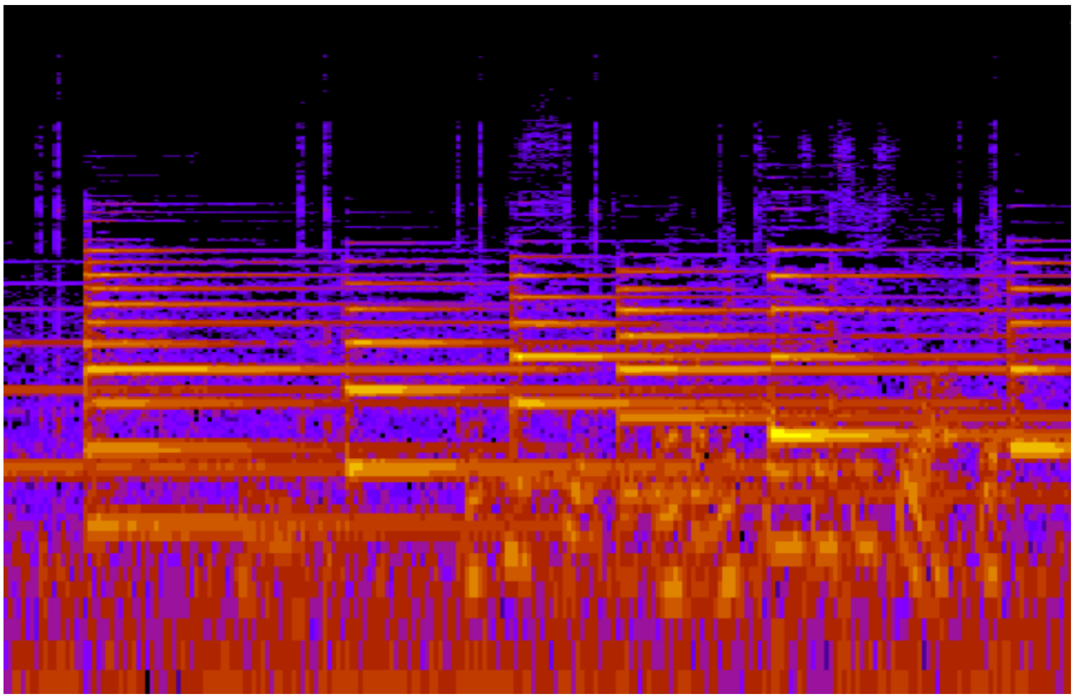
Figure 20 The input spectrogram

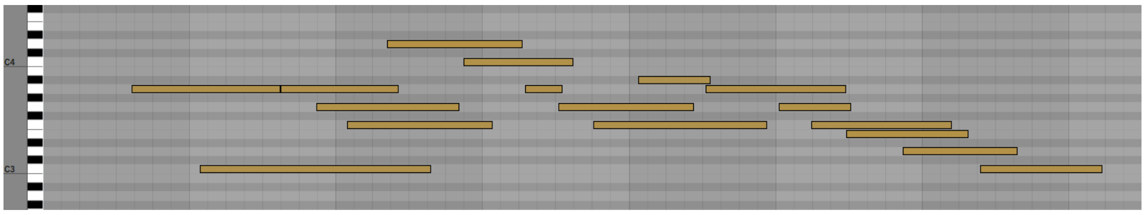

Figure 21 The MIDI file transferred from the input spectrogram

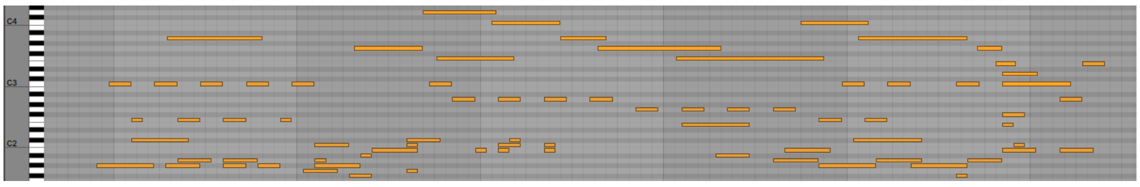

Figure 22 The MIDI file of the music arranged from 19

1.6 Evaluation

Up to now, we did not find a good evaluation method for the generated music besides hearing test. Therefore, we found 20 people to join our hearing test, including different ages and sexual. We ask them to listen to our arranged music and give a rank between 1 to 10, the result is shown in the coming table:

Table 6 Evaluation

Samples	Average Rank
Music Sample 1	4.2
Music Sample 2	5.0
Music Sample 3	3.9

From the result, we know that our arranged music still has a long distance from human reality. We still have a long way to go.

5. Conclusion and Future Work

5.1 Conclusion

In this research, we made an attempt to build a music arrangement system based on Machine Learning. The system solves two main problem for music arrangement. The first problem is to transfer idea of the users into a score, which can be considered as MIDI file. We used a model called Onset and Frame to transfer the input music from spectrogram into MIDI file. This is the first part of our system.

The second problem is to arrange the music. Using Cycle-GAN model, we add more details in our music, and we keep the main rhythm at the same time. During the experiment, we tried different structures and different layers in our structure. At last, we get our result. This is the second part of our system.

We also have two more problems during we are working on our research. The first one is collecting our dataset. We need a dataset with 2 groups of music, including arranged music and un-arranged music. However, we did not find a dataset which can fit our research well. As a result, we collect and process the data by ourselves, which take us plenty of time. However, our dataset is still small. A larger dataset may give us a better result.

The last problem is the evaluation. We cannot find an evaluation method for our arranged music. Therefore, we confirmed the results by groups of hearing tests. Although the rank is not high, but the music is being arranged and the result is not bad.

As a conclusion, our music arrangement system works well, but still have a long way to go. Some of the parts can be improved, which will mention in the coming section.

5.2 Future Work

For the future work of our research, we can briefly summary as 2 parts:

First and the most important, we need to continue the work of data collection. Although we already took plenty of time on data collection, the dataset is still too small. In machine learning, usually a larger dataset means a better result.

Second, we can try some other structure besides Cycle-GAN. In fact, we are trying Pix2Pix [26] in our system, but the work is not done yet. Using machine learning on music arrangement has too much possibility. We still have a long way to go.

6. Appendix

6.1 Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Professor Hiroshi Watanabe of Department of Computer Science and Communications Engineering, Graduate School of Fundamental Science and Engineering, Waseda University, for his kind and enthusiastic guidance in this research, and for providing me with an experimental and comfortable research environment.

I would like to thank Professor Hiroyuki Kasai for giving us accurate and valuable opinions for participating in the summer camp and the minority team seminar in 2019.

I would like to thank everyone in the laboratory for their comments, advice, and technology sharing at the seminar. Particularly, I would like to thank the members in audio team of Watanabe Lab. The advice from Hangyu Song and Ahmad Moussa inspires me a lot about my research and my life.

Lastly, I would like to express my gratitude to the family for their great support in my study and my whole life.

6.2 Bibliography

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville "Deep Learning" www.deeplearningbook.org (Accessed on July 15, 2020)

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014.

[3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros "Image-to-Image Translation with Conditional Adversarial Networks" arXiv preprint arXiv:1611.07004(2017)

[4] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, pp. 2242–2251.

[5] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, Jaegul Choo "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation" arXiv preprint arXiv:1711.09020

[6] Swift, Andrew. (May 1997), "A brief Introduction to MIDI", https://web.archive.org/web/20120830211425/http:/www.doc.ic.ac.uk/~nd/surprise_97/journal/vol1/aps2/ （Accessed on July 15, 2020）

[7] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, Douglas Eck "Onsets and Frames: Dual-Objective Piano Transcription" arXiv preprint arXiv:1710.11153

[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition"; AT&T Bell Laboratories

[9] Oh, KS; Jung, K (2004). "GPU implementation of neural networks". Pattern Recognition. 37 (6): 1311–1314.

[10] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", ILSVRC2012, ImageNet, 2012

[11] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11):2673–2681, 1997.

[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2818–2826, 2016.

[13] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition." In CVPR, 2016.

[14] J. Johnson, A. Alahi, and L. Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." In ECCV, 2016

[15] J. Zhao, M. Mathieu, and Y. LeCun. "Energy-based generative adversarial network." In ICLR, 2017.

[16] Ebrahimi, Touradj; Pereira, Fernando (2002). "The MPEG-4 Book." Prentice Hall Professional. ISBN 9780130616210.

[17] Gino Brunner, Yuyi Wang, Roger Wattenhofer, Sumu Zhao "Symbolic Music Genre Transfer with Cycle-GAN", 2018, arXiv preprint, arxiv:1809.07575

[18] Ronneberger, Olaf; Fischer, Philipp; Brox, Thomas (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". arXiv preprint**,** arXiv:1505.04597

[19] "Releases – numpy/numpy". Retrieved 4 June 2020 – via GitHub.

[20] M. Data, "Intuitive analysis, creation and manipulation of midi data with pretty midi," 2014

[21] "Listen to your favorite MIDI files on BitMidi", https://bitmidi.com/ (Accessed in 2019 and 2020)

[22] "Carlo's MIDI" https://www.cprato.com/ (Accessed in 2019 and 2020)

[23] "Midiworld.com" https://www.midiworld.com/files/, (Accessed in 2019 and 2020)

[24] Hahnloser, R.; Sarpeshkar, R.; Mahowald, M. A.; Douglas, R. J.; Seung, H. S. (2000). "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit". Nature. 405 (6789): 947–951. Bibcode:2000 Natur. 405..947H. doi:10.1038/35016072. PMID 10879535.

[25] Hahnloser, R.; Seung, H. S. (2001). Permitted and Forbidden Sets in Symmetric Threshold-Linear Networks. NIPS 2001.

[26] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros "Image-to-Image Translation with Conditional Adversarial Networks" arXiv preprint, arXiv:1611.07004

6.3 List of Figures

6.4 List of Tables

6.5 List of Academic Achievements

[1] Tianyi GAO, Hangyu SONG, Ahmad MOUSSA, Hiroshi WATANABE: "An Automatic Music Arrangement System Using Machine Learning" IEICE General Conference D-14-3, Mar. 2020