6-1-2017

# Chinese Font Style Transfer with Neural Network

Xue Hanyu
*Dartmouth College*

# CHINESE FONT STYLE TRANSFER WITH NEURAL NETWORK

**DARTMOUTH COMPUTER SCIENCE TECHNICAL REPORT**

**TR2017-830**

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Master of Science

in

Computer Science with a Concentration in Digital Arts

by

Hanyu Xue

DARTMOUTH COLLEGE

Hanover, New Hampshire

June 1, 2017

Examining Committee:

Qiang Liu, Chair

Lorie Loeb

Wen Xing

F. Jon Kull, Ph.D.
Dean of Graduate and Advanced Studies

# Abstract

Font design is an important area in digital art. However, designers have to design character one by one manually. At the same time, Chinese contains more than 20,000 characters. Chinese offical dataset GB 18030-2000 has 27,533 characters. ZhongHuaZiHai, an official Chinese dictionary, contains 85,568 characters. And JinXiWenZiJing, an dataset published by AINet company, includes about 160,000 chinese characters. Thus Chinese font design is a hard task. In the paper, we introduce a method to help designers finish the process faster. With the method, designers only need to design a small set of Chinese characters. Other characters will be generated automatically. Deep neural network develops fast these years and is very powerful. We tried many kinds of deep neural network with different structure and finally use the one we introduce here. The generated characters have similar style as the ones designed by designer as shown in experiment part.

# Acknowledgements

To the committee members who helped me make it work.

To my family who supported me during my master program.

To my friends who gave useful suggestions for the program.

And to those I have not acknowledged yet: if you are reading this at all, thank you!

# Contents

# List of Figures

# INTRODUCTION

Now computer fonts are used widely. Most people only read articles, papers, books and write them on electronic device. People are interested with various fonts because different fonts provide different feeling. Designers use font in their product to convey various emotion. And ordinary people prefer to use special font for personal interest. Thus a lot of companies focus on generating new font. Thousands of new fonts are generated every year. However, although number of font products increased rapidly in these decades, demand for more fonts is still increasing.

Currently, professional company rely on CAD software heavily, which need a lot of manual operations and professional experience. The most popular softwares are Fontlab, Fontographer and AsiaFont Studio. And because Chinese font designing is too complex, many companies develop their own software for more efficient designing. However, no matter how we re-design these software, the process is similar. Designers have to design characters one by one, which is time-consuming. And for most ordinary people, there is no way to design a personal font. China has the tradition of calligraphy. Chinese people start to design beautiful font with writing brush thousands of years ago. Even today, a lot of Chinese people still spend time to practice hand-writing. And many children learn how to write beautiful characters with professional teachers for several years. Thus many Chinese people are good at hand-writing

and are willing to digitize characters in their personal fonts.

Building a Chinese font library is not easy because of the large size of character set. Unlike English, which includes less than one hundred alphabet, Chinese has more than 20,000 characters. Modern simplified Chinese has 27,533 characters according to Chinese official character set GB 18030-2000. ZhongHuaZiHai is an official dictionary, which contains all Chinese characters used at that time, including the Chinese characters used outside China and characters in dialect. There are 85,568 Chinese characters in the dictionary. The largest Chinese font dataset is JinXiWenZiJing, a dataset generated by AINet company. The set contains Chinese characters appearing in different places and time, including those that Chinese people will not use any more today. The dataset size is more than 160,000. All this big numbers indicates how difficult to design a new Chinese font. A professional designer could design about 20 characters each day. So it will take one designer about 4 years to design GB 1830-2000 dataset and 32 years to design JinXiWenZiJing, which is unacceptable. Thus most companies have a big designer group and take about 1 year to design a new font.

In this paper, we introduce a method to accelerate and simplify the process. With the method, designers only need to design a small set of Chinese character and find an existing font as reference. The method will use the provided character to train the neural network. After training, the network could generate any character. With the method, designers only need to design 1% - 10% characters of original GB 18030-2000 set. And with a scanner, the method makes it possible to digitize personal font.

# RELATED WORK

Researchers and scientists have explored some methods to generate fonts automatically. Some work focus on decomposing Chinese character into different components to allow designer only design components. Then software assemble components together to get characters. [3] and [6] both belong to this kind. There are two disadvantages of these methods. The first one is that it is still a difficult mission to design components because Chinese characters are too complex. The other one is that result is not good enough because same component may different in different Chinese characters.

Another kind of method is as [10], [4] and [8]. The methods are much more efficient than last one and result is much better. Users are required to design models for each characters in training set and provide a large set of characters in different fonts. Then softwares generate all the rest characters automatically. The method is close to practical use. However, because the provided models have to be accurate and complex, it still need a lot of manual operation and professional training. [5] provides an interesting method. The generated characters are similar with ground true. However, users have to decompose training characters manually, which makes the method less efficient.

[2] project provides a good direction for Chinese font transfer. The method uses

neural network to generate characters. The method is easy to use because designers just need to provide a small character image set. Then the network generates all the rest characters. However, the training set contains 3,000 characters, which is still too large. A professional designer will take about 5 months to design 3,000 characters. And according to the result provided, the generated characters are still not good enough. [1] uses similar idea on English. The method works well because there are only 62 different characters in English, while Chinese has tens of thousands characters.

In this paper, we also use neural network to generate characters. The difference is that [2] project uses fully connected network with 3 layers, while our method uses deep fully convolutional network. There are two advantage of our method, the first one is that convolutional network is more suited to image. Convolutional network uses context information to predict result, which is important for image. And because convolutional network always has less parameters than fully connected network, the training process need less time. Another advantage is our network is deeper than the network used by [2]. Deeper network always catch features of training data better and thus has better result. However, deeper network has over-fitting problem. We avoid this problem by adding l2 regularization part in loss function. And according to the result, our method indeed generates better result. Deep learning develops fast these years. There are many kinds of deep neural network. With some experiences, we finally use FCN as our network structure.

# Method

We use neural network as the method. The input of the network is a character image in an existing font. And the output of the network is a character image in target font. To use the network, designers are expected to provide a training data set and an existing font as reference. Each data in the data set is a pair as [character image in existing font, character image in target font]. The data will be used to train network. After training, given a character image in existing font, the network will generate the character image in target font.

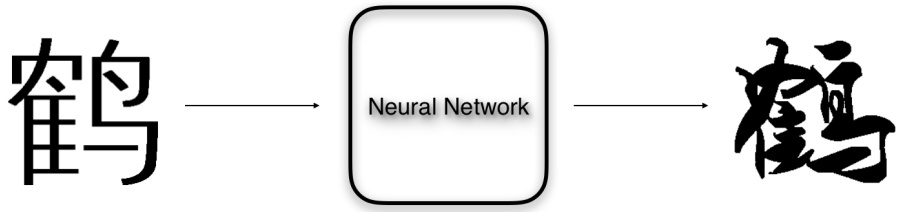


Figure 1: Method structure

## Neural Network

Neural network is a computational model used in machine learning. It is consist of many layers. Each layer has one or multi neural. The first layer of neural network

is input layer, last layer is output layer, and other layers are hidden layer. Neural in input layer are special because they get information from input directly. All other neural combine information form last year. When given a input to neural network, the information is delivered start from input layer, across hidden layers and finilay come to ouput layer. The information in output layer is the output of the neural network.

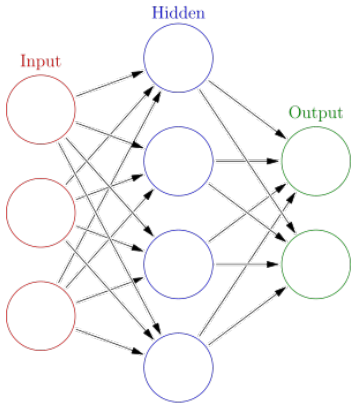


Figure 2: A one layer neural network

There are many ways for a neural to combine information from last layer. The general way is:

$$h_j^i = f(w * [h_{j0}^{i-1}, h_{j1}^{i-1}, h_{j2}^{i-1}, \cdots] + b) \tag{1}$$

$h_j^i$ is the $j_{th}$ neural in $i$ layer. $h_{j*}^{i-1}$ are neural that connected with $h_j^i$. $w$ is a matrix and $b$ is a number, parameters of this neural. $f$ (commonly referred to as the activation function) is a predefined function, such as hyperbolic tangent or sigmoid function.

Before using neural network to generate output, network should be trained with training data. Training data set is a label data set, which means each data in the set is a pair of [input, output]. The output in the pair is the ground truth, the one we want network generate when given the input data. The training process is

a process to optimize $w$ and $b$ of each neural to make the output of the network as similar with ground truth as possible. To optimize $w$ and $b$, a loss function is needed. Loss function is a function of the distance between ground truth and the output of the network, which we want to decrease. $L$ is used to present loss function, $\theta$ for all parameters, $I$ for input and $T$ for ground truth. And if we consider the whole network as a function $g$, the output $O$ of the network is:

$$O = g(\theta, I) \tag{2}$$

and the loss distance $l$ between output and ground truth is:

$$l = L(O, T) \tag{3}$$

$$l = L(g(\theta, I), T) \tag{4}$$

For a given data, $I$ and $T$ are fixed and only $\theta$ is parameter. So the problem become how to update $\theta$ to decrease $l$.

$$arg \min_{\theta} L(g(\theta, I), T) \tag{5}$$

Gradient descent is used to update $\theta$. Gradient represents the slope of the tangent of the graph of the function, which means if we change $\theta$ in the gradient direction, $l$ will decrease fastest.

$$\theta \leftarrow \alpha * \nabla_{\theta} L(g(\theta, I), T) + \theta \tag{6}$$

$\alpha$ is the step size, a very small constant number. Given one data, $\theta$ is updated by a small step. Neural network may use training data many times to optimize

parameters. However, computing gradient for all parameters is time-consuming. Thus back propagation is used to compute gradients efficiently.

## Network Structure

There are many kinds of deep neural network. With some experiments, we decide to use Fully Convolutional Network based on VGG 16-layer network.

**VGG 16-layer net**: [9] introduces VGG network. The paper evaluates networks of increasing depth using an architecture with very small (3*3) convolution filters. The experiments show that the depth of 16-19 with the filter size generates good result. One main contribution of the network is that 3*3 filter is better than 5*5 filter. The reason is that two 3*3 filters have same receive field with one 5*5 filter, while two 3*3 filters have less parameters than one 5*5 filter. Less parameters results to less training time. There are 13 convolutional layers in VGG-16. Convolutional layer is useful in image problem. In image, one pixel always share information with other pixels around it. Convolutional layer takes advantage of this property to make prediction. And there are five max pooling layers. When depth of network increases, overfitting become a nonnegligible problem. Max pooling layer decreases detail information to avoid overfitting. Image 3 shows the structure of VGG network. There are six different structures. According to the paper, VGG-16 has best result, as the one in green rectangle in the image. The goal of VGG-16 is classification. Hence at the end of network, there are three fully connection layers. The three layers combine all information together to predict the class of the input image. And because there are three fully connection layers, the dimension of input image is fixed.

**Fully Convolutional Network**: VGG-16 is a classification network. To generate

| ConvNet Configuration | | | | | |
|---|---|---|---|---|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64<br>**LRN** | conv3-64<br>**conv3-64** | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128<br>**conv3-128** | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 |
| maxpool | | | | | |
| conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256<br>**conv1-256** | conv3-256<br>conv3-256<br>**conv3-256** | conv3-256<br>conv3-256<br>conv3-256<br>**conv3-256** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

Figure 3: VGG networks structure

image, [7] changes the structure of VGG-16. FCN replaces all fully connection layer with convolutional layers. Then the output will be an image in stead of a class. However, because VGG-16 uses max pooling, output image will be smaller than the input image. FCN up-samples image at the last layer, then the dimension of output image is same with the input one. Another important contribution of FCN is skip layer. The authors believe output of different layers catch information of original image at different level. Deeper layers contain more abstract information, while lower layers have more detail. To combine all this information, the paper combines some layers at different level before up-sampling. Thus, there are three different FCN, FCN-32s, FCN-16s and FCN-8S. FCN-32s has no skip layer. FCN-16s combines two layers, while FCN-8s combines three layers. The result of FCN-8s is better than the other two, and it is slower than them. We use FCN-8s for better result. Original FCN is designed for segmentation. But the idea of FCN-8s is also suitable to font

transfer. According to experiment result, FCN-8s works well.

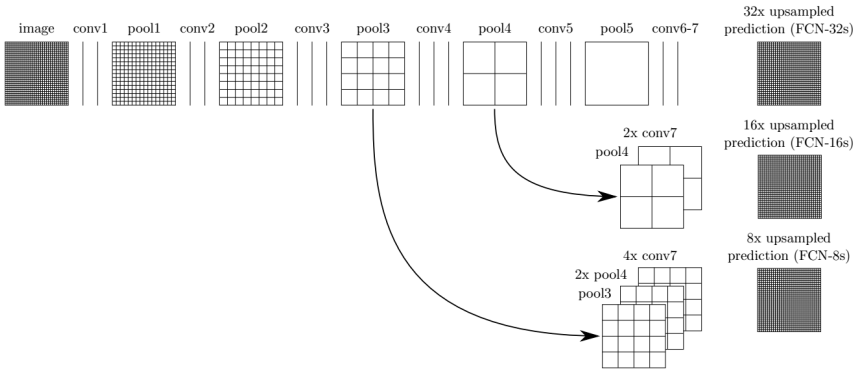


Figure 4: Fully Convolutional Network structure

# More Detail

The output of FCN is not character image directly, but the probability of each pixel to be part of character. Higher probability means the pixel is more likely to be part of character, not background. When given a character image in original font, we use the network to generate this probability matrix. If the probability is higher than 0.5, we take the pixel as foreground, otherwise the pixel will be part of background.

As introduced above, loss function is important for neural network. The method uses cross-entropy as loss function. $T$ is the ground truth image. $T_{ij}$ is the class of the pixel in image $T$ at $i_{th}$ row and $j_{th}$ column. If the pixel is part of foreground, $T_{ij}$ is 1, other wise it is 0. $P$ is the generated probability matrix. $P_{ij}$ is the probability of pixel in the matrix at $i_{th}$ row and $j_{th}$ column to be foreground. Given the ground truth image with size of $W * H$ and $P$ matrix, the loss function is:

$$L(T,P) = -\frac{1}{W*H}\sum_{i=0}^{H}\sum_{j=0}^{W}[T_{ij}logP_{ij} + (1-T_{ij})log(1-P_{ij})] + \frac{1}{2}\lambda\theta^2 \qquad (7)$$

$\frac{1}{2}\lambda\theta^2$ is the L2 regularization to avoid over-fitting.

**Algorithm 1** Training Network

Initialize parameter $\theta$ randomly
**for** each iteration **do**
    **for** each batch **do**
        $L(T,P) = -\frac{1}{W*H}\sum_{i=0}^{H}\sum_{j=0}^{W}[T_{ij}logP_{ij} + (1-T_{ij})log(1-P_{ij})] + \frac{1}{2}\lambda\theta^2$
        update $\theta$ with backpropagation as:
        $\theta \leftarrow \frac{1}{batch\_size}\sum_{i=0}^{batch\_size}(\alpha * \nabla_\theta L(T_i, P_i) + \theta)$
    **End For**
**End For**

**Algorithm 2** Generating Character

Load parameter $\theta$ from trained model
Feed character in original font $I$ as input to network
Get ouput probability matrix $P$
Generate a blank image $O$ with same size with image $I$
**for** each row $i$ **do**
    **for** each column $j$ **do**
        **if** $P_{ij} > 0.5$ **then**
            Pixel $O_{ij}$ is black
        **else**
            Pixel $O_{ij}$ is white
        **End If**
    **End For**
**End For**

# Software Framework

The code is developed in python. Python has a lot of libraries to support development of neural network.

**Tensorflow**: Tensorflow is a open-source library for machine intelligence. The library is originally developed by researchers and engineers of Google Brain Team,

and now is open to all developers. Tensorflow uses data flow graphs. Mathematic operations are presented as node in the graph, while data arrays are edges (Tensor) between the nodes. Tensorflow allows us to build neural network fast and easy to run on CPU and GPU.

**Numpy**: NumPy is a fundamental package for scientific computing with Python. The library allows us to represent image data in multi-dimension vector and compute easily. We present all our data as Numpy multi-dimension vector. The output of the network is also an vector. Python functions help us save them as images.

**Pillow**: Pillow is an open-source python imaging library developed by Alex Clark and other contributors. The library allows us to present image in python code easily. Currently most font libraries are saved as TTF file. Because the input of our network is RGB image, we need to use Pillow library to save all training characters as PNG file.

Other libraries we use are Commentjson, Scipy and Matplotlib. Commentjson helps us to write JSON file. Scipy is a basic library for Numpy. And Matplotlib is designed for generating plot.

# EXPERIMENT AND ANALYSIS[1]

We design several experiments to explore the ability of our method. Accuracy and training time are important for neural network. So we will present these two values in most of our experiments. SongTi font is a classic Chinese font and is pre-installed on most computers. So we use SongTi as our basic font. When training, the batch size is 32 and max iteration is 12,000. All experiments are processed on single GPU TITAN X with 12G memory

## Experiment 1: Different Size of Training Set

Training set size is one important factor. The motivation of the project is to help designers design fonts faster. The best situation is that designers only need to design one character, then the network generates all characters in the same style. However, this is impossible because one character is not enough to present a style. In another word, designer could not present their idea in one character. So how many characters are needed to train the network?

There are two factors affecting training set size. One is what characters we use for training. Different characters contain different information of a style. If the characters

[1]All fonts mentioned are from [11]. We use same name as the font name on [11]

we use are too similar, they could not fully present a style. Thus it is unreasonable to expect the network generates good characters. So we choose fonts randomly from Chinese character set.

Another factor is target font. Some fonts are difficult for the network to predict, while some are easier. The target font we use is WeiYouYuan. The font is selected because it combines printing font and handwriting font and thus presents a medium level. With the same font, we change the training data set size and test on same 100 characters. There are 7 different set size, 3000, 2000, 1000, 500, 100, 50 and 10. The training characters and 100 validation characters are selected randomly from SongTi font set. There is no overlap between training set and validation set.

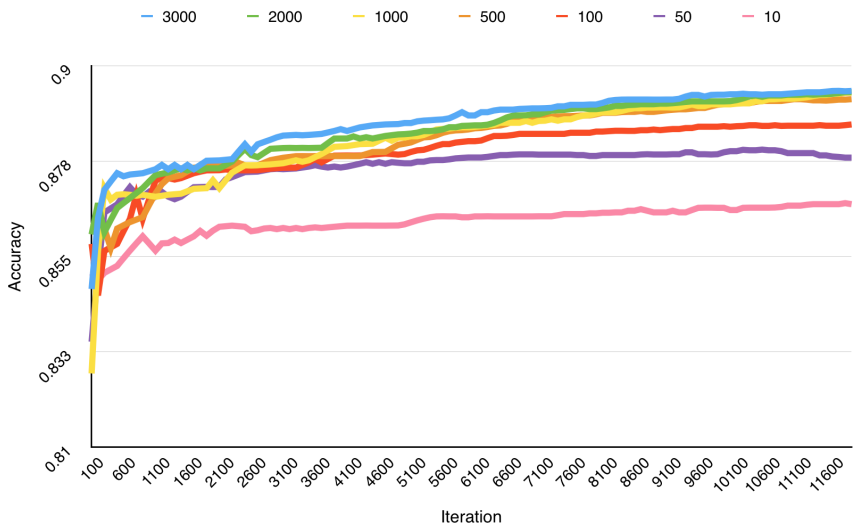


Figure 5: Accuracy chart of different training set size

To evaluate the method, we define the accuracy as:

$$Accuracy = \frac{\sum_{i=0}^{H}\sum_{j=0}^{W}\begin{cases} 1 & ,\text{if } O_{ij} = T_{ij} \\ 0 & ,\text{if } O_{ij} \neq T_{ij} \end{cases}}{H * W} \tag{8}$$

$O_{ij}$ is the pixel of output image on $i_{th}$ row and $j_{th}$ column. $T_{ij}$ is the pixel of ground truth image on $i_{th}$ row and $j_{th}$ column. $H$ and $W$ are height and width of the image respectively. The accuracy is a number between 0 and 1. 1 means the generated image is totally same with ground truth, which is a good result. And zero means the generated image is totally different with the ground truth. It is an unacceptable result. As chart shows, accuracy starts from about 0.82 and ends at about 0.89 after 12,000 iterations. For all experiments, accuracy keep increasing. Thus there is no overfitting. If we keep training the model, accuracy may keep increasing. But the result is good enough after 12,000 iterations. The chart shows that the method will have better performance if training set is bigger. However, experiments with 3000, 2000 and 1000 set size have similar accuracy result. And according to the output images, training set with 1,000 characters indeed generates good result. Thus 1,000 is an ideal size of training set. And because WeiYouYuan is pretty different with SongTi, while most other fonts are much similar, other fonts will need smaller training set. There are big accuracy gaps between 1000, 500, 100, 50 and 10. And according to output images, quality of output images decreases obviously. The results of 500, 100, and 10 have disconnections between strokes.

Training time is also very important. The training time for all these 7 experiments is about 2.0 hours. The reason is that compared with training set size, iteration number is much bigger. Thus to decrease training time, decreasing iteration is much useful than decreasing training set size. If training set size is 1,000, after 6,000 iteration, results are much similar with final results. Thus deceasing iteration number is a feasible solution.

## Experiment 2: Different Fonts

Same Chinese character in different fonts could be totally different. Some of them look similar with input font SongTi, while some look like a different character. Intuitively, the fonts that are different with SongTi will be a more difficult task for our network. Thus, the ability of the network to deal with different fonts is important.

To evaluate the ability of the network, we use five different fonts from [11]. We carefully choose the fonts to test network on different levels. There are two printing style fonts and three handwriting style fonts. Printing style fonts are similar with SongTi and thus easier for network to predict. Handwriting fonts are much difficult, because there are a lot of connections between adjacent strokes, change of stroke width and tilt, while SongTi is clear and consistent-width. The five fonts are SongTeXi, WeiHeiJian, WeiYouYuan, XingKai and TongTiJian. We use 1,000 as training set size as last experiment shows. All experiments take about 2 hours to train and could generate about 15 images each second when testing.

Same as experiment 1, we use accuracy defined as last part to evaluate the result. According to the chart, some fonts start with high accuracy because they are similar with SongTi, while others start with low accuracy value. SongTeXi, WeiHeiJian and WeiYouYuan end with high accuracy, thus they are successful. XingKai and TongTiJian start with low accuracy. The accuracy increases with iteration increasing. But the final accuracy is still too low, which means the generated characters are very different with ground truth. From the accuracy, we can predict that the first three fonts will be more successful than the other two.

As images shown in appendix B, SongTeXi, WeiHeiJian and WeiYouYuan fonts have good results. The predicted characters are similar with ground truth. Although

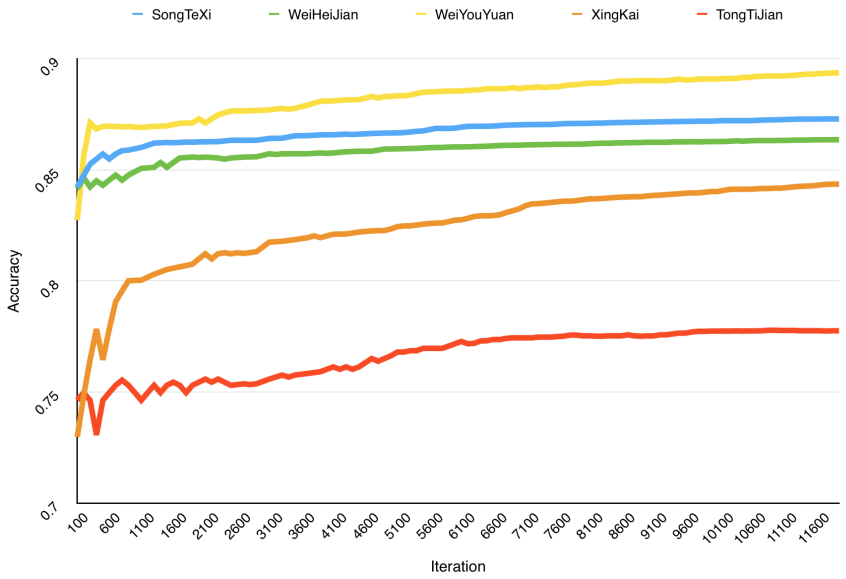


Figure 6: Accuracy chart of different fonts

they are not totally same, they have same style. Generated characters of SongTeXi have thin lines, WeiHeiJian characters are bold and WeiYouYuan characters have rounded edges. Thus the method works well on these three fonts. However, XingKai and TongTiJian are not as successful as these three fonts. We even cannot tell out what the characters are in generated images. Generated XingKai characters catch some features of the style. They have similar outlines with ground truth. But the generated TongTiJian characters almost have nothing in images.

The mainly reason of failure is that target fonts are too different with original font. Ground truth characters of XingKai and TongTiJian are totally different with SongTi fonts. Location and shape of each stroke are different. Compared with XingKai and TongTiJian, SongTeXi, WeiHeiJian and WeiYouYuan are similar with SongTi. Although stroke shape is different, location does not have significant change. This experiment reveals one limitation of the method. The method is good at catch the

shape change and is weak at location change. Thus, to get good result, a good original font is very important. An ideal original font is that it has similar stroke location with target font, while shape of each stroke is different.

## Experiment 3: Characters in Other Languages

Characters in all languages have similar components, horizonal stroke, vertical stroke, dot, left-falling stroke and right-falling stroke. Chinese characters is one of the most complex characters. It covers almost all components we need for any language. Is it possible to predict characters in other languages? This is useful because it is amazing if a font set contains characters in all language.

The last two experiments evaluate the ability of the method to generate Chinese characters. This experiment presents the transferring ability of the method between different languages. This experiment trains network on Chinese and try to predict character in other languages. The original font is still SongTi and the target font is WeiYouYuan. The training set size is 1,000 and iteration number is 12,000. As last experiment shows the model works well on Chinese characters. The other languages we use are English, Japanese and Roman numbers.

As results shown in Appendix C, the method works well on the three languages. Generated images are very similar with the ground truth. They have same style. Without ground truth, it is difficult to tell out they are generated by network. However, compare with the ground truth, there are some problems. Same with Chinese characters, disconnection is still the main problem. For English characters, disconnection is not obvious. For Japanese characters, disconnection is obvious. But, the disconnection makes the characters more beautiful. The generated Japanese charac-

ters look like made by writing brush. However, disconnection in numbers is a serious problem. Because numbers are made of one stroke, disconnection makes them different with original numbers. For example, 8 and 6 have a lot of disconnections. It is a little difficult to tell out they are 8 and 6. Disconnection is a limitation of the method. We will discuss this in the future work part.

# CONCLUSION AND FUTURE WORK

The thesis presents a method to transfer Chinese font style. The method helps font designers design a new font faster. Designers only need to design a small set of Chinese characters manually, then other characters will be generated automatically.

However, there are some limitations of the method. The main problem is disconnection. As shown in experiment part, compared with ground truth characters, generated characters have disconnection problem. Another problem is that if target font and original font are very different, the method does not work. So a good original font is necessary for the method. Our future work will focus on these two problems. These years, generative adversarial network has a great development and solve many problems in different areas. The network may be helpful to solve the problems. We will try to use GAN to transfer Chinese font style in the future.

# APPENDIX

## Appendix A: Experiment 1

鲢 茄 灼 邹 躬 觉 娇 焉 彰 鹤 琳 沦 畔 惹 庶
毙 皖 邢 禹 渍 绷 窜 翘 淫 箪 陌 膊 鞑 咳 玫
巫 拂 蕉 澜 赎 绥 锄 囱 赌 颊 缕 寅 躁 稚 庚
苟 氦 魁 珊 蜕 蛭 酌 逗 闺 蔓 撇 豌 朕 缉 襟
镍 桅 荧 侄 卒 佃 瞿 娶 饪 耸 乍 靶 痴 靖 扛
筐 韶 嚣 崔 蓿 岔 氚 娥 剿 霖 喃 搪 雍 裳 撰
豹 骏 慷

Figure 7: Original SongTi characters

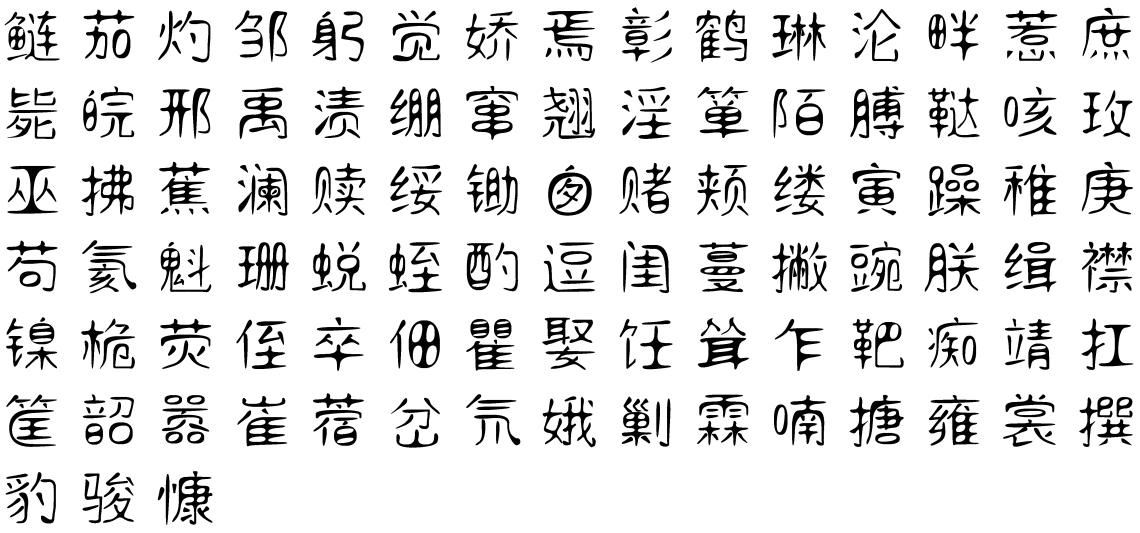

Figure 8: Ground truth characters of WeiYouYuan font

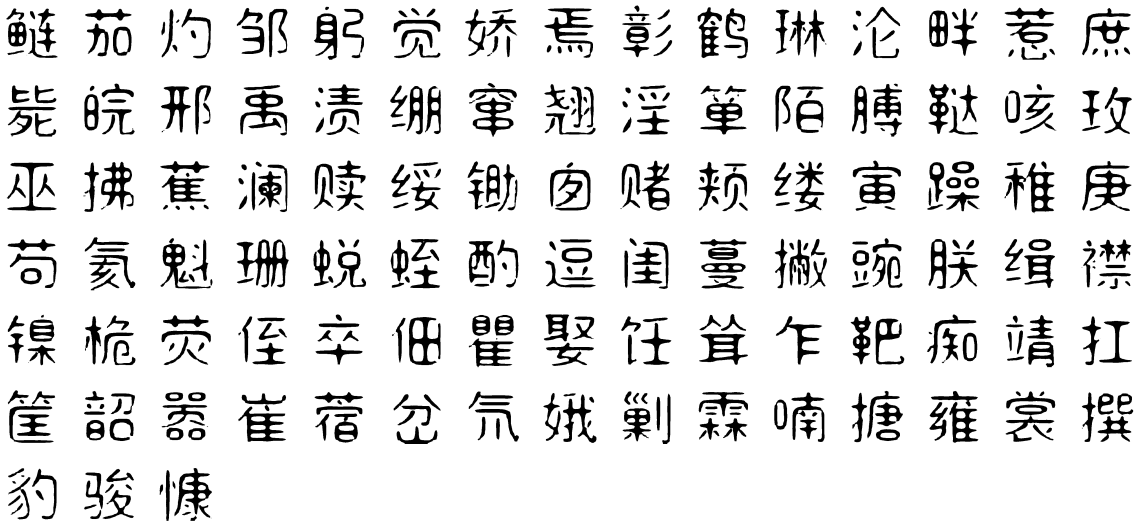

Figure 9: Generated WeiYouYuan characters with 3000 as trainging set size

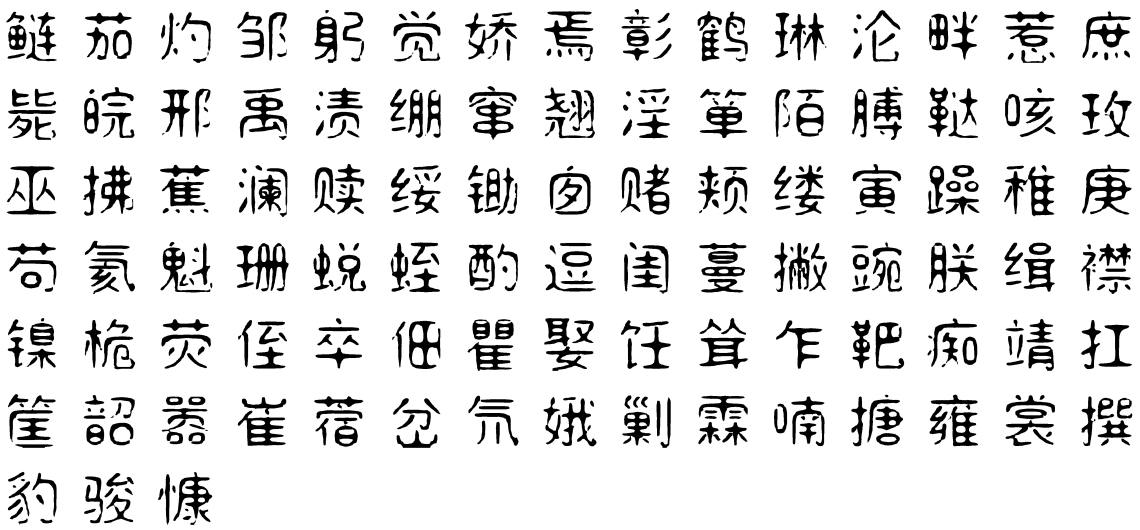

Figure 10: Generated WeiYouYuan characters with 2000 as trainging set size

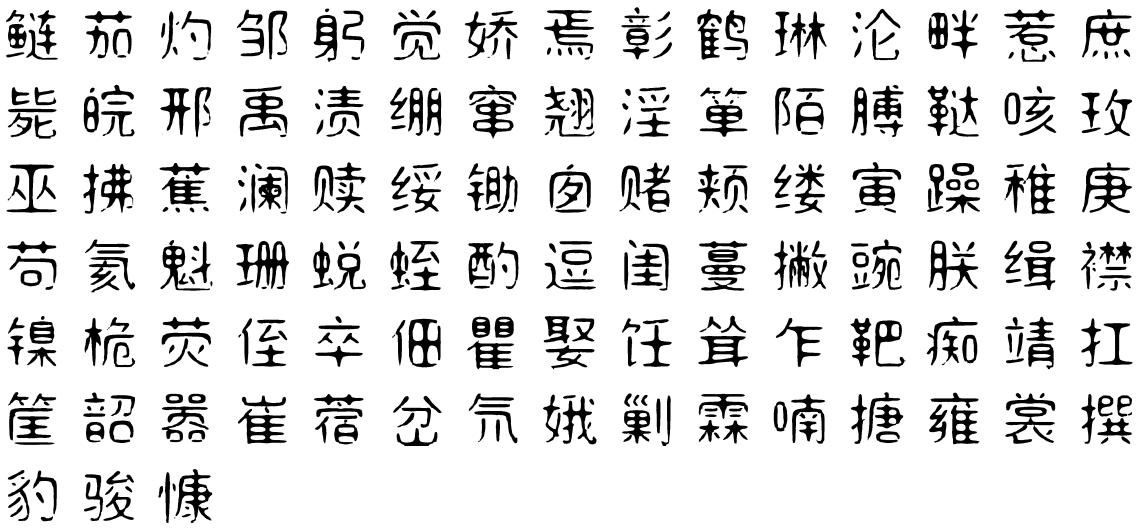

Figure 11: Generated WeiYouYuan characters with 1000 as trainging set size

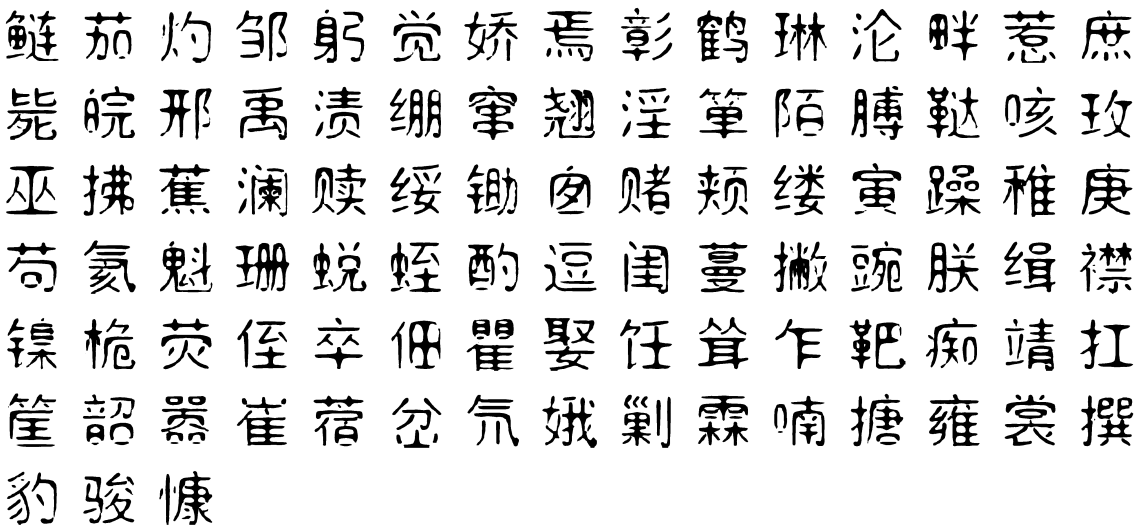

Figure 12: Generated WeiYouYuan characters with 500 as trainging set size

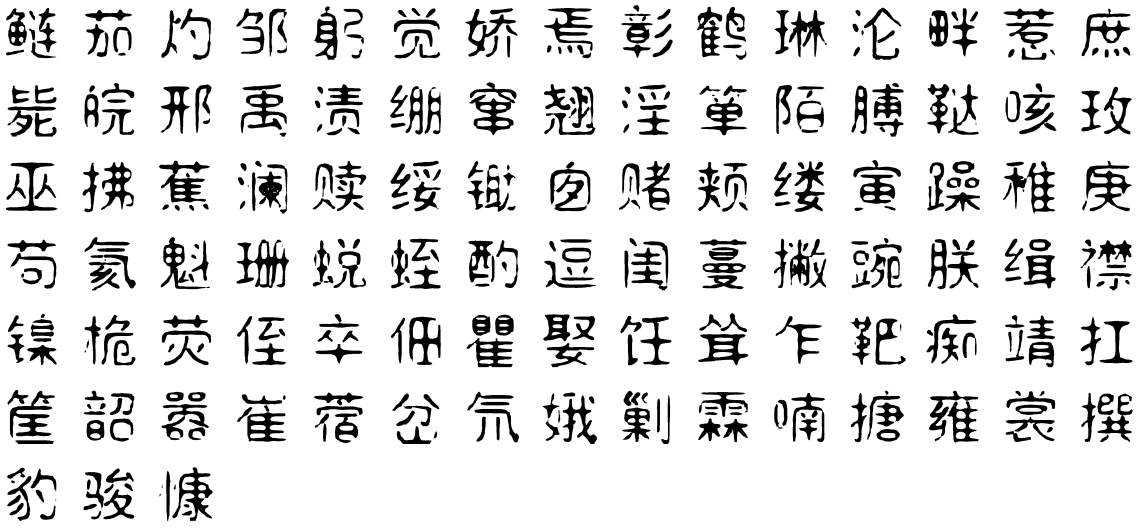

Figure 13: Generated WeiYouYuan characters with 100 as trainging set size

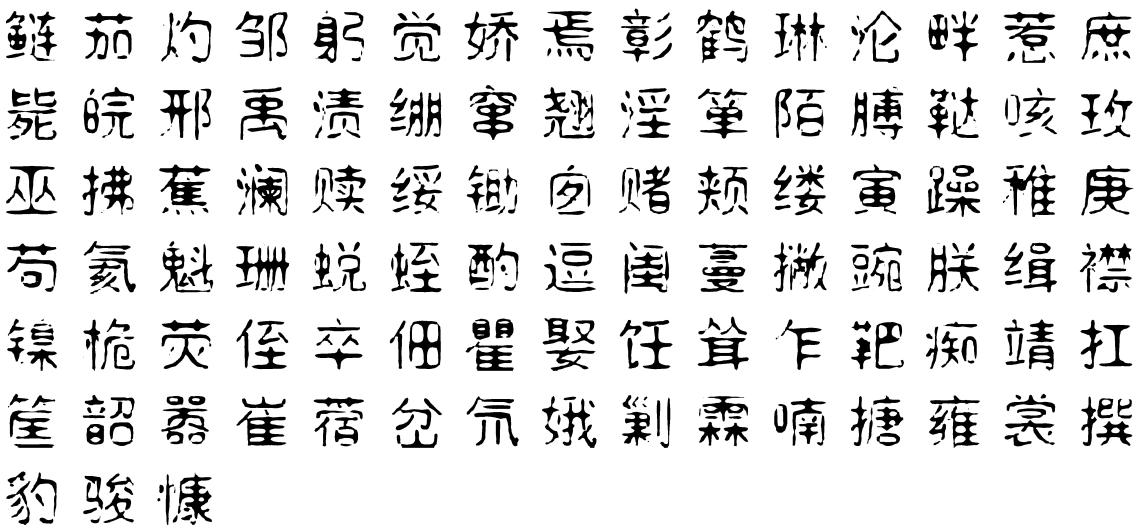

Figure 14: Generated WeiYouYuan characters with 50 as trainging set size

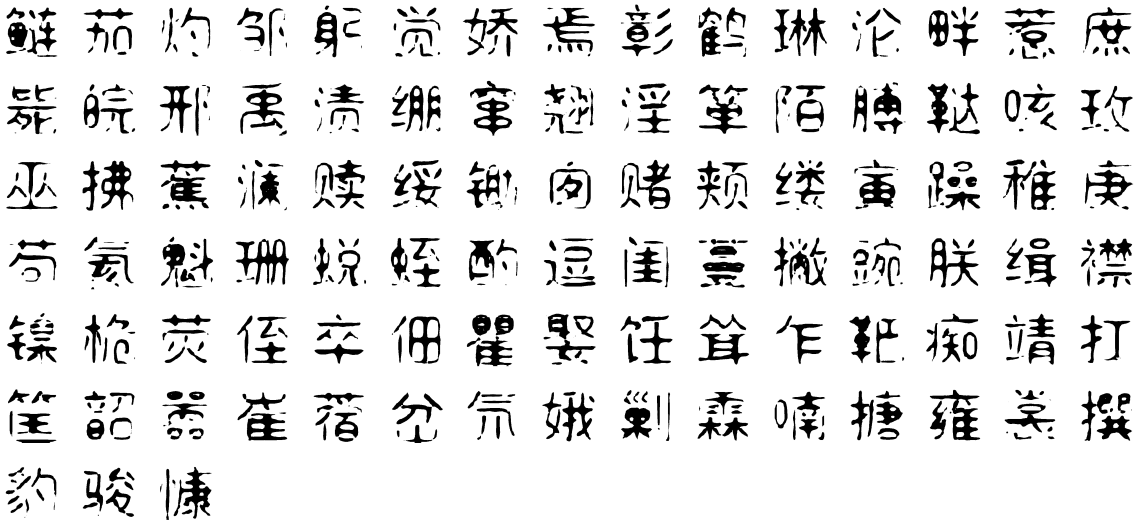

Figure 15: Generated WeiYouYuan characters with 10 as trainging set size

## Appendix B: Experiment 2

鲢 茄 灼 邹 躬 觉 娇 焉 彰 鹤 琳 沦 畔 葱 庶
毙 皖 邢 禹 渍 绷 窜 翘 淫 箪 陌 膊 鞑 咳 玫
巫 拂 蕉 澜 赎 绥 锄 囱 赌 颊 缕 寅 躁 稚 庚
苟 氦 魁 珊 蜕 蛭 酌 逗 闺 蔓 撇 豌 朕 缉 襟
镍 桅 荧 侄 卒 佃 瞿 娶 忏 耸 乍 靶 痴 靖 扛
筐 韶 嚣 崔 蓿 忿 氚 娥 剿 霖 喃 搪 雍 裳 撰
豹 骏 慷

Figure 16: Original SongTi characters

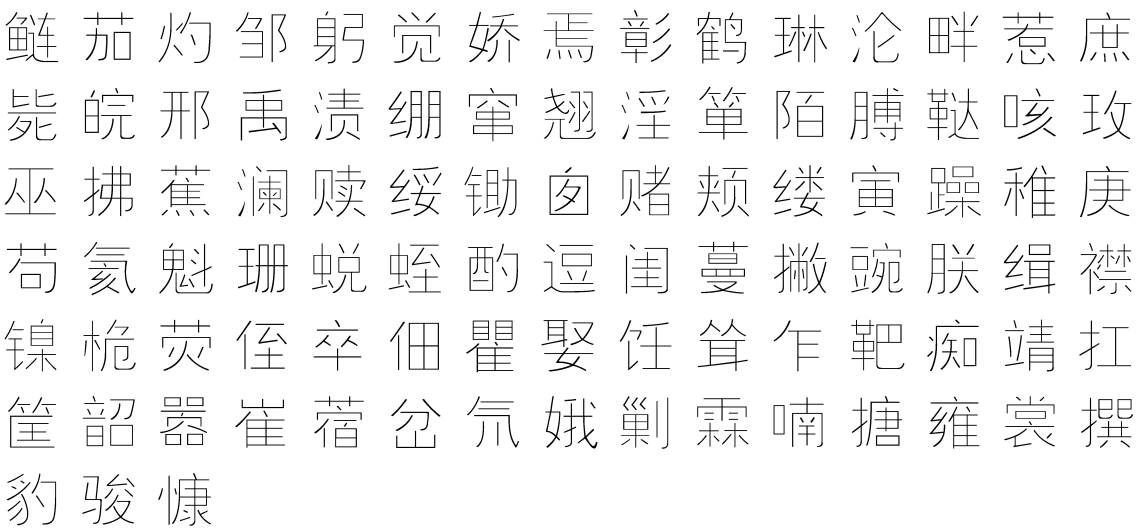

Figure 17: Ground truth SongTeXi characters

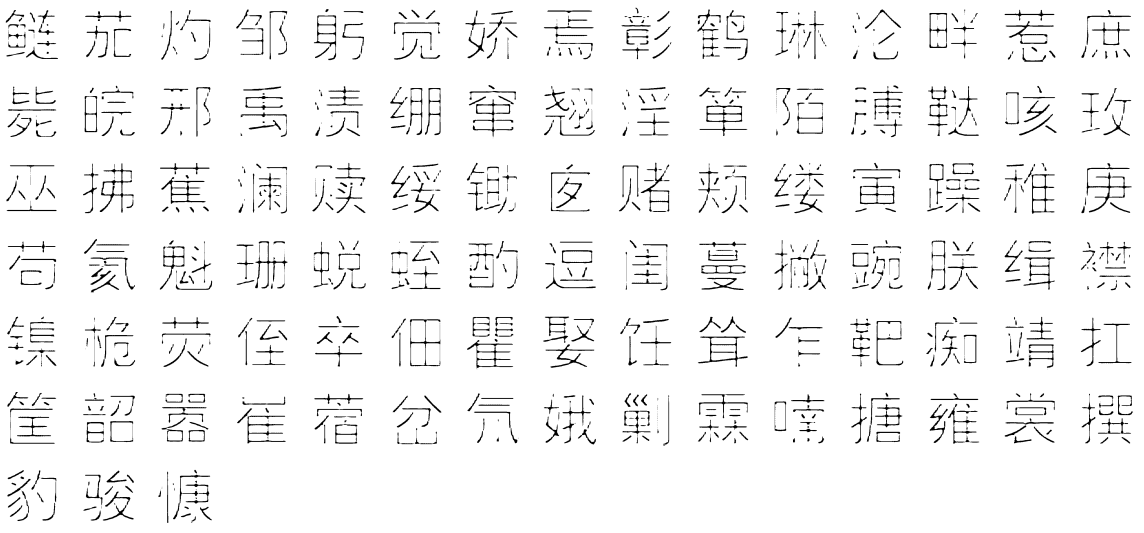

Figure 18: Generated SongTeXi characters

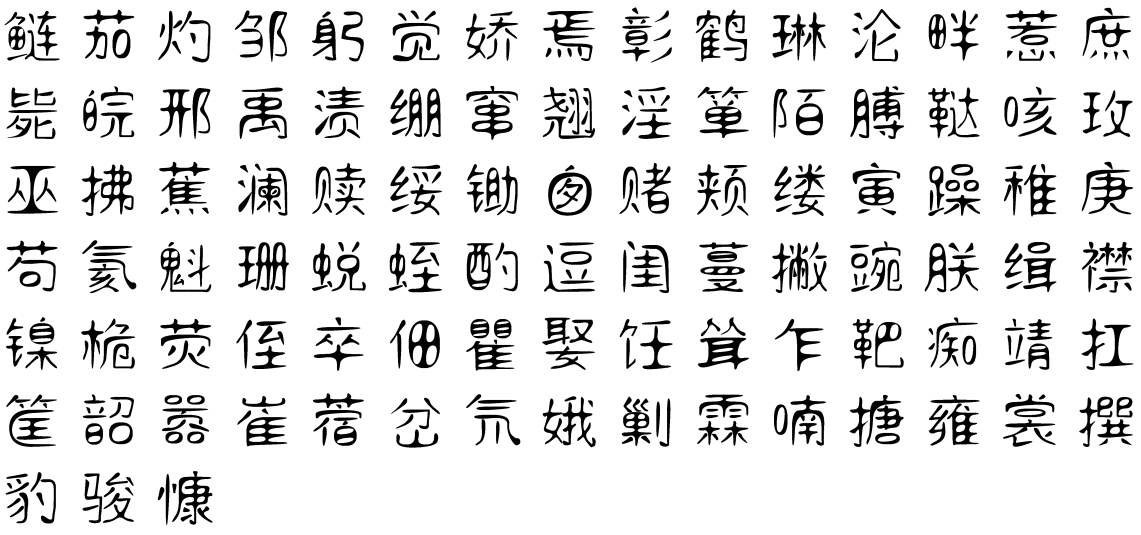

Figure 19: Ground truth WeiYouYuan characters

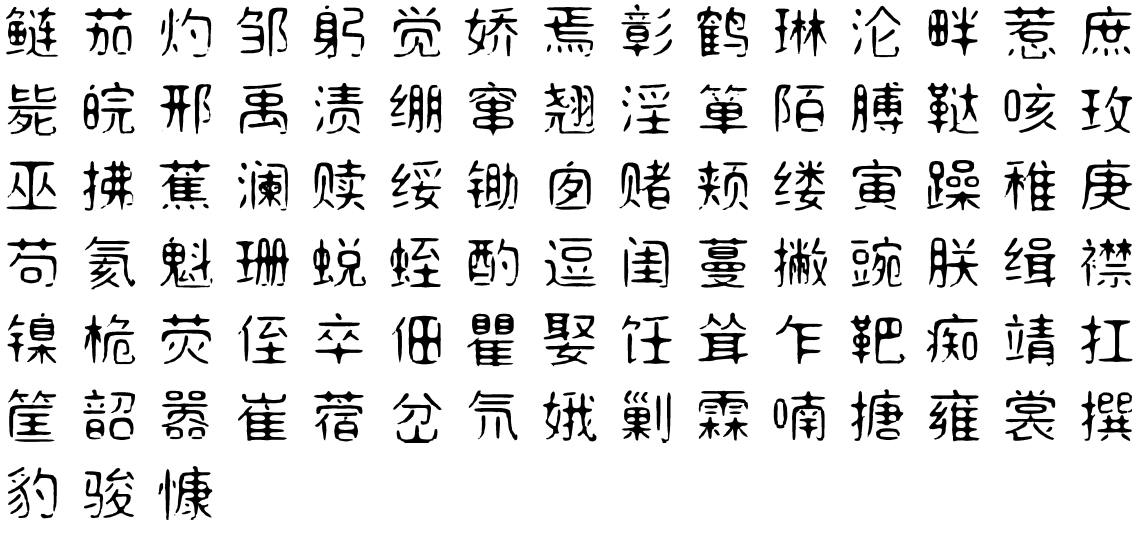

Figure 20: Generated WeiYouYuan characters

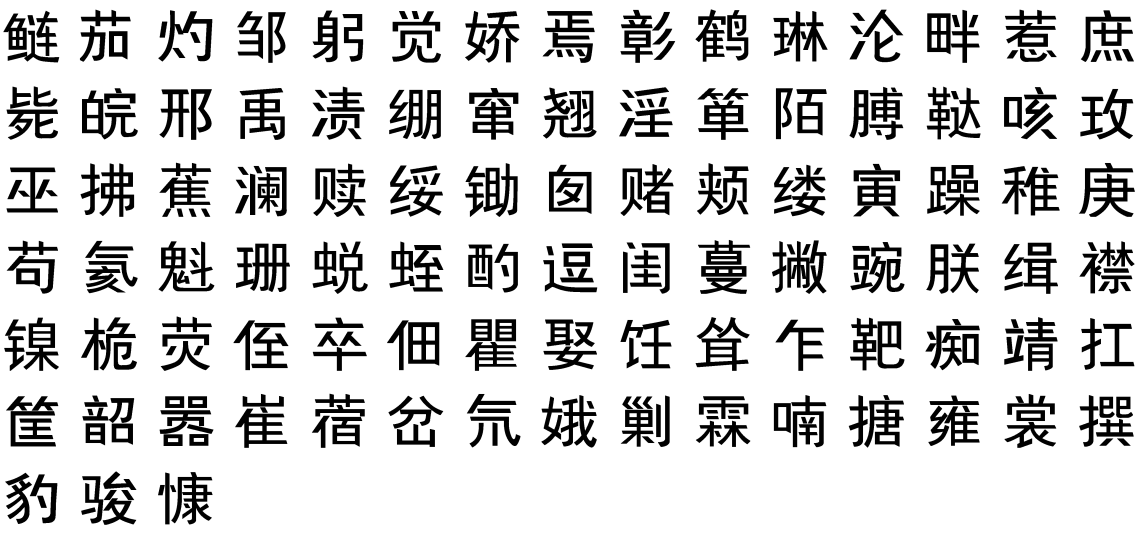

Figure 21: Ground truth WeiHeiJian characters

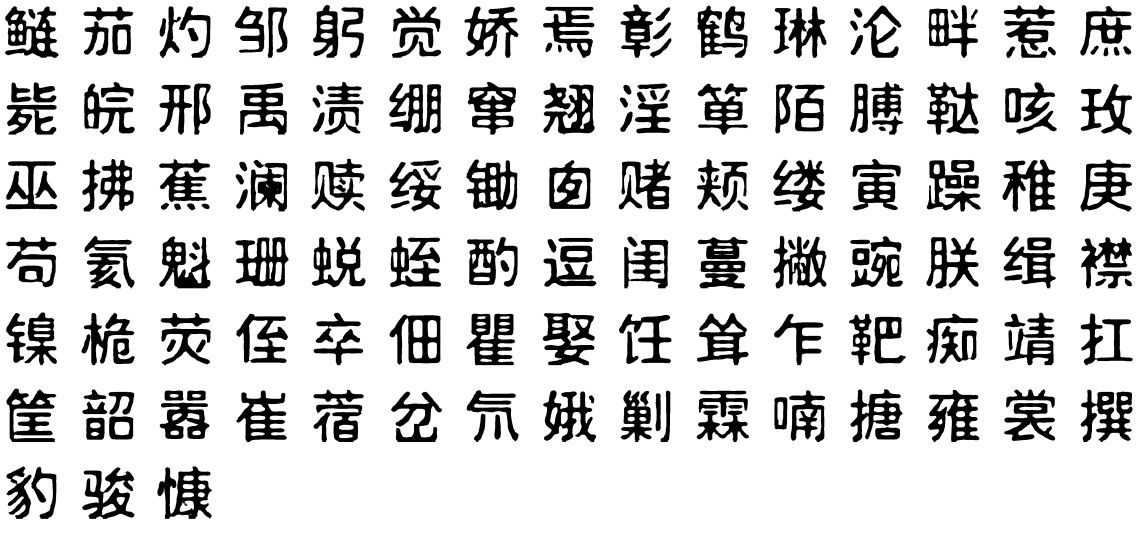

Figure 22: Generated WeiHeiJian characters

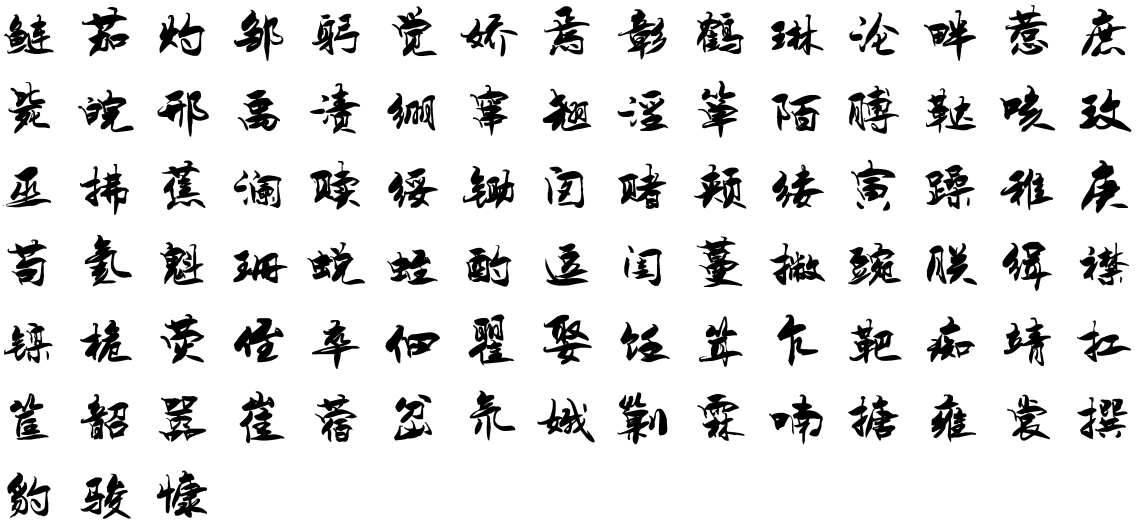

Figure 23: Ground truth XingKai characters

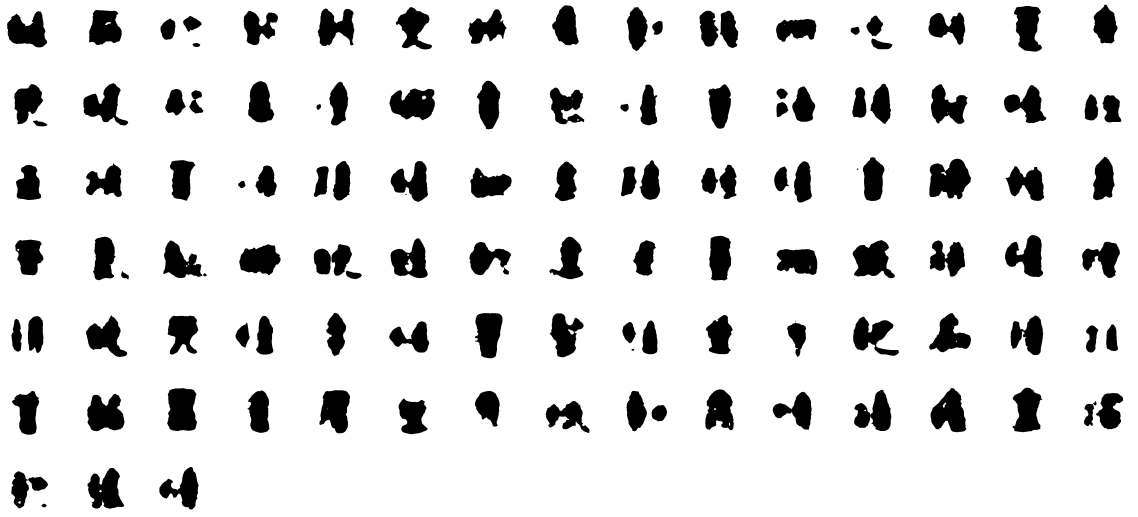

Figure 24: Generated XingKai characters

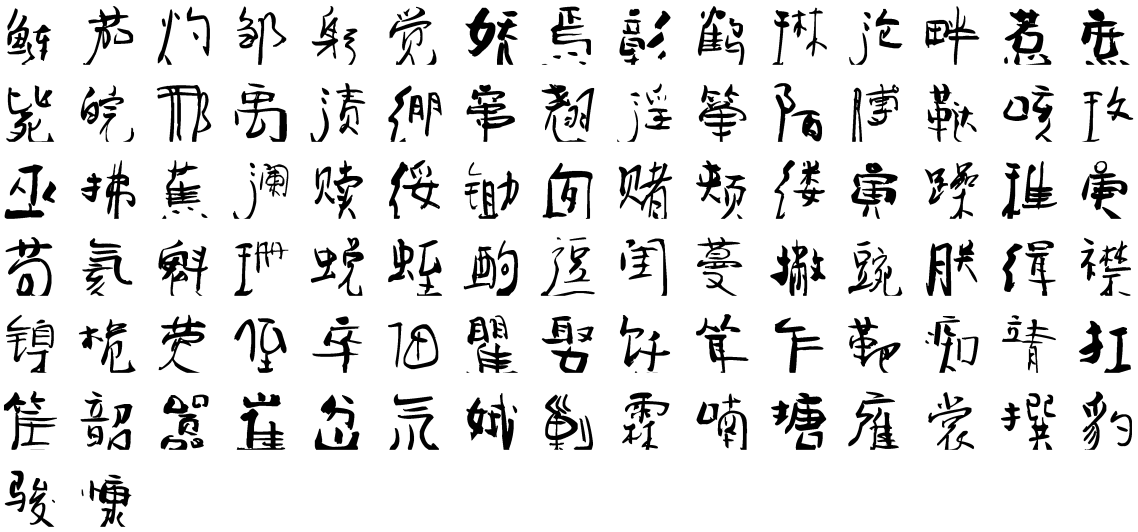

Figure 25: Ground truth TongTiJian characters

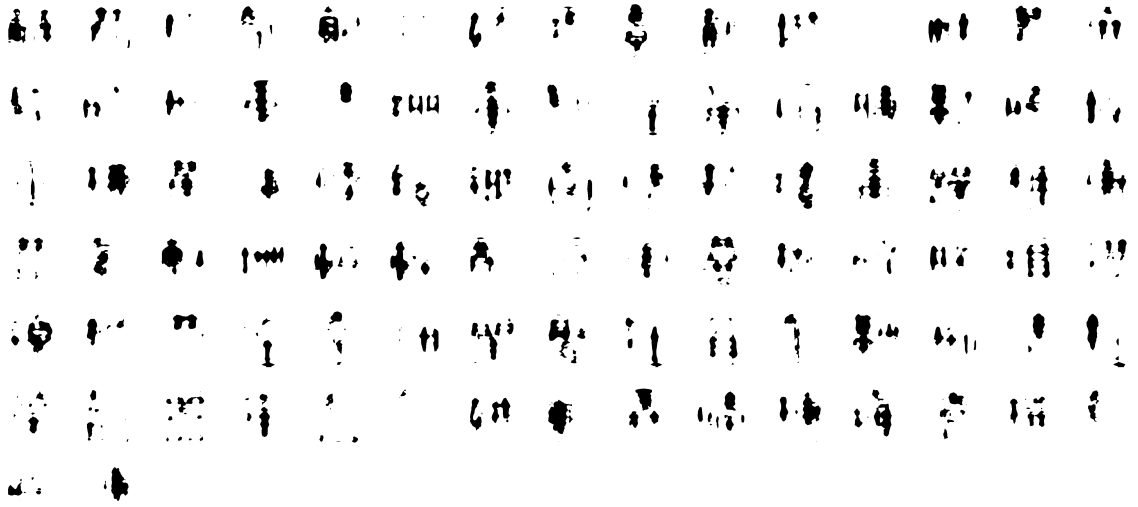

Figure 26: Generated TongTiJian characters

# Appendix C: Experiment 3

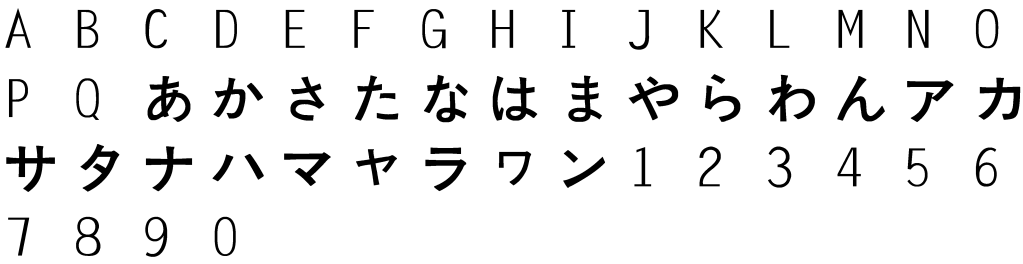


Figure 27: Non-Chinese characters in SongTi

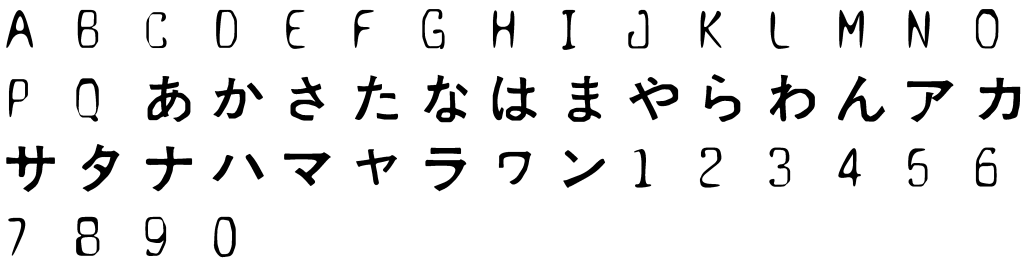


Figure 28: Ground true non-Chinese characters

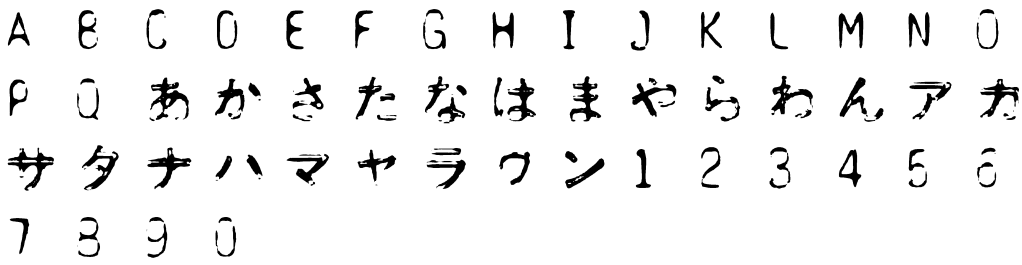


Figure 29: Generated non-Chinese characters

# Bibliography


[1] Shumeet Baluja, *Learning typographic style*, arXiv preprint arXiv:1603.04000 (2016).

[2] Kaonashi-tyc, *Neural style transfer for chinese characters*, 2016.

[3] Pak-Keung Lai, Dit-Yan Yeung, and Man-Chi Pong, *A heuristic search approach to chinese glyph generation using hierarchical character composition*, Computer Processing of Oriental Languages **10** (1996), no. 3, 307–323.

[4] Zhouhui Lian and Jianguo Xiao, *Automatic shape morphing for chinese characters*, SIGGRAPH Asia 2012 Technical Briefs, ACM, 2012, p. 2.

[5] Zhouhui Lian, Bo Zhao, and Jianguo Xiao, *Automatic generation of large-scale handwriting fonts via style learning*, SIGGRAPH ASIA 2016 Technical Briefs, ACM, 2016, p. 12.

[6] Jeng-Wei Lin, Chih-Yin Wang, Chao-Lung Ting, and Ray-I Chang, *Font generation of personal handwritten chinese characters*, Fifth International Conference on Graphic and Image Processing, International Society for Optics and Photonics, 2014, pp. 90691T–90691T.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell, *Fully convolutional networks for semantic segmentation*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

[8] Huy Quoc Phan, Hongbo Fu, and Antoni B Chan, *Flexyfont: Learning transferring rules for flexible typeface synthesis*, Computer Graphics Forum, vol. 34, Wiley Online Library, 2015, pp. 245–256.

[9] Karen Simonyan and Andrew Zisserman, *Very deep convolutional networks for large-scale image recognition*, arXiv preprint arXiv:1409.1556 (2014).

[10] Rapee Suveeranont and Takeo Igarashi, *Example-based automatic font generation*, International Symposium on Smart Graphics, Springer, 2010, pp. 127–138.

[11] YeGenYou, *Yegenyou chinese fonts website*, 2017.