The efficient generation of large character set fonts in commercially usable quality

Hongye Zhang

Abstract

The generation of large character set fonts is a domain with extensive research due to the difficulties of manually producing fonts, for they are expensive, time-consuming and susceptible to the proficiency of the artists, therefore creating the need to automate this process. Such feats are already achieved in the case of encodings such as Latin due to its simplicity and low quantity of data(256 glyphs). In large character set fonts, such as CJK Fonts (Chinese fonts, Japanese fonts, Korean fonts), however, those methods do not work for large character set fonts because the glyphs are as much as 6,000 ~ 20,000, so it is difficult to keep consistency for so many glyphs. This paper proposes a novel method implementing the StyleGAN architecture and its variants for the generation of mainly large character set fonts. The new method realizes the automatic and efficient production of high quality, commercially usable large character set font while being able to specify the desired style if needed. It is therefore a method to be able to mass produce large character set fonts such as CJK Fonts for commercial use.

1. Introduction

Text is an essential and perhaps one of the most important parts which make up human culture, taking a considerable portion of information. Thus, text design is critical to improve the quality and experience of consumers. Which is why in modern designing, the use of different fonts is widely spread. A myriad of fonts is necessary, as many try to appeal through exciting text coupled with striking design to create a longer lasting impression, which is why the current typeface market value in only China, Japan and Korea amount to 300 million US dollar. However, most of the time, one will meet the problem that the production of a new font is quite costly. Especially so in large character set fonts such as CJK (Chinese, Japanese, Korean) fonts, which include thousands of glyphs that need to be created.

For instance, a commercially usable Chinese font needs at least 6000 characters. To produce such a font, artists with rigorous training and years of experience are needed. Moreover, the production of excellent fonts may take up to 1-2 years to finish. The main reason behind the tremendous effort and the long time it takes lies within the issue of keeping the style of the font consistent. The style can be classified as the combination of both the skeleton and the details (see 3.1), and both need to be of consistent style for it to be considered a commercially usable high quality font.

2 Related works

Font generation as image-to-image (I2I) translation.

I2I translation [4] aims to translate the contents of a source to another image. Recent I2I translation methods learned to map between multiple diverse domains, i.e., multi-domain translation, that can naturally be adopted into the font generation problem. For instance, by mapping fixed source fonts to the target font with paired I2I translation [5].
Such methods are only capable to change the stroke detail of the fonts but fail to change the

skeleton, which is not sufficient for a new, different font.

Few-shot font generation.

Few-shot font generation attempts to generate new glyphs with very few numbers of existing style references without additional Fine-tuning. The mainstream of few-shot font generation attempts to disentangle content and style representations as style transfer methods [1][2] [3], but specialized to font generation tasks. For example, AGIS-Net proposes the font-specialized local texture discriminator and the local texture refinement loss. Unlike other methods, DM-Font disassembles glyphs to stylized components and reassembles them to new glyphs by utilizing strong compositionality prior, rather than disentangling content and style. Despite notable improvement over past years, previous few-shot font generation methods have significant drawbacks, such as being unable to generate complex glyph-rich scripts, failing to capture the local diverse styles, or losing the complex content structures.
Overall, Few-shot generation methods tend to lose consistency rapidly when processing a numerous glyphs, hence it is only suitable for small character set fonts.

Stroke based generation

Similar to the concept of the Few-shot generation, the method aims to be able to automatically generate components of glyphs, or extract the skeleton of components for style translation and assemble them into a glyph, e.g. by mapping the coordinates of strokes [9]. Unlike few-shot generation, this method generates its own strokes instead of extracting them from an existing font [6]. This requires large numbers of references to generate a new style, e.g. 775 glyphs.
Methods of this category are very effective to keep stroke consistency, but fail to produce a high quality skeleton and are unable to keep the skeleton consistent.

Other Chinese font generation methods.

Some other approaches make use of the compositionality to reduce search space in character space in order to produce a smaller component space. RD-GAN [11] explores the possibility of generating unseen characters in the fixed style (zero-shot generation). CalliGAN [10] encodes the styles with one-hot vectors; which needs additional fine-tuning for making unseen styles during the training. However, the aforementioned methods are also failing to keep consistency for whether skeleton or stroke, while other methods do not possess skeletons of strokes of satisfactory quality.
We can conclude that common problems in all large character set font generation include:

- Unable to handle thousands of glyphs at once
- Unable to keep the style consistent simultaneously for both skeleton and detail
- lacking quality of glyphs
- difficult data acquisition
- tolerance for missteps
- low efficiency

Therefore, resulting in previous methods failing to produce complete large character set fonts.

3. Our method

We propose a novel method, consisting of three different sections. In the first section, the pre-processing focuses on the ability to gain control over the resulting style, as well as the task of keeping the style consistent through dataset composition and data enhancement. The second section, the process itself, applies StyleGAN to generate large character set fonts through style mixing, creating entirely new skeletons and details. The third section, the post-processing, utilizes various image manipulation and processing procedures to ensure the high quality of the details. As a consequence, we are able to create consistent and commercially usable large character set fonts of high quality.

3.1 Preparation of Data

To ensure the quality and consistency of results, the preparation of data is of paramount importance. For the input, we will first need a moderate amount of fonts, annotated as F_0, to use as the Ground Truth. These fonts will serve as the reference of the generated results by comparing them as it is designed in the StyleGAN network. StyleGAN takes bitmap pictures as input, which is why glyph images, annotated as I_0 need to be created. The dataset will consist of a range of glyphs ($[1:\infty[$), which will be annotated as $G = \{$ 的, 是 , G_3, … , $G_i \}$. I_0 will be generated for G from F_0.

Style mixing

Another important factor while assembling the dataset is the amount of different styles as well as the quantity of I_0 for each style which will be annotated as $S = \{ s_1 , s_2 , \dots , s_i \}$. The percentage will inevitably influence the bias b in the AI architecture, and it is possible to use the bias to partially control the direction and general style of the results. For instance, if the entire dataset $D = \{ s_1 \}$, obviously, the only possible outcome will be results of the style s_1. If we use the dataset of $D = \{ s_1 , s_2 , s_3 \}$ instead, we will get combinations of styles which influence each other, and thus get e.g. results similar to s_1 but with traits or features of s_2 and s_3 . Most of the changes will occur to the details, with minor, yet noticeable changes to the skeleton of glyphs.

Image 1: Differences between structural changes and detail changes. Red represents the structure/structural changes, and blue the details or (mostly) changes in the detail, shown on the character 字

Style control

Datasets often require careful assembly of data as in most cases, it is advantageous to have an unbiased dataset through careful weighting. Weighting itself, however, can also be exploited to increase the probability density of certain results appearing. According to the likelihood function, we can define $P_\theta(X = x)$ as the probability of a source X being from style/ label θ having the feature x which equals $p_\theta(x)$ with p being the probability mass function and can be summarized as $L(\theta \mid x)$. Therefore, the probabilities $P(\theta \mid x)$ are given by

$$P(\theta \mid x) = \frac{L(\theta \mid x)}{\Sigma_\vartheta L(\vartheta \mid x)}.$$

Which is the likelihood of $L(\theta \mid x)$ divided by the total number of likelihoods.

In case of two different sources, $L(0 \mid x)$ and $L(1 \mid x)$, the combined continuous likelihood of any case appearing will be $\sum_i L(\theta \mid x)$.

Henceforth, it is possible to weight $P(\theta \mid x)$ through the input of more sources of the amount of features present. Let $D = \{ s_1, s_2 \}$ with the ratio $7 : 3$ be the dataset. P will be weighted more by s_1 due to its increased presence, while s_2, although still weighting the results, will in most cases only be of secondary importance, affecting details and minor structural changes. This however, requires each s in D to be represented in percentages not too low, for it may result in complete negligence of said style.

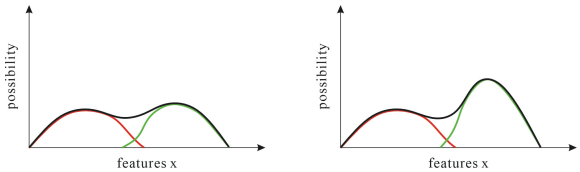

Image 2: With red and blue representing different styles with varying features, the black line represents the combined possibility of a certain feature appearing. The figure on the right shows the possibilities when the amount of $L(\theta \mid x)$ for the blue curve increases, showing higher possibilities of source X having features of the blue style.

Style modification

When generating I_0 , to increase structural change and to enhance the creativity of results, it is possible to modify glyphs with techniques related to bitmap manipulation. This data augmentation, if practiced, can manually determine the general shape of glyphs in a font. For instance, if one modifies the shape of the glyph to be a circle, then as a result, the generated fonts will also be referencing the $Ground\,Truth$

3.2 Implementation of large picture datasets

So far, the use of StyleGAN has always centered around the input of an image with only one thing depicted, e.g. only one face, one car or only one glyph. However, if we want to achieve the creation of a font with a consistent style, it would not be possible to have only one glyph as the input. No matter how good the quality of the glyph is, the model is still separately trained, and thus making it impossible to assemble a font out of inputs of each individual model. If said method were to generate a complete font, it would first, require massively more time to train, as one would have had to train a model for each glyph, as well as the comparison of different generated glyphs to determine whether the glyphs could belong to the same font, while never having the possibility to be 100% accurate. Therefore, I propose the following method.

The most important factor in maintaining stable and consistent style across all glyphs. This method requires the assembly of an so called large data picture, which will be annotated as $I_{Grid} = \{ G_1, G_2, \ldots, G_i \}$ and is consisting of I_0 put in a grid. This will ensure that in the latent space, the entire font is treated as one, thus avoiding the problem of inconsistent style across glyphs. The order in which the glyphs are placed in the grid must remain the same to avoid the confusion of the model, leading to no disentangling of the images.

For each I_{Grid} , the shape, size and number of rows and columns (dimensions) are the same.

$I_0 \in I_{Grid}$, $dim(G_1) = dim(G_i)$.

Since StyleGan is scalable to higher resolutions, one can theoretically extend this procedure infinitely, where the only limit is the computing power required.

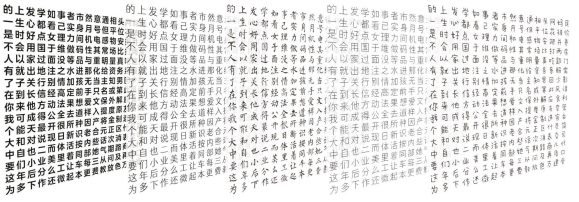

Image 3: Example of two large data images of the dimensions 16x16 glyphs with each glyph having the resolution 64x64 pixels. The order and amount of glyphs are the same, with each I_{Grid} representing a font.

The emphasis here lies within the resolution of images. As we need to put all our glyphs into one large picture, it is not feasible to use images of a resolution too high, as it would exponentially increase the size of the larger picture, and therefore the time needed to train the model would increase exponentially. For example, if one wants to generate the entire GB2312 Chinese encoding, one would have about 7000 glyphs, which, if every glyph is only 64x64 in shape, it would already require an image of more or less 5400x5400 to fit in all the glyphs, which increases tremendously if every glyph increases in size. However, without question, if computing power is not an issue, one would be free to train glyphs in higher resolutions, as it would naturally increase the image quality. But if that is not the case, it is highly recommended to keep the size in a moderately small range, e.g. 64x64 pixels per glyph, as it is shown above.

3.3 The use of StyleGAN models

The most important requirements to creating a new style out of existing ones are as followed:

1. The ability to generate high quality results.
2. The ability to be able to compute high resolution inputs
3. The ability to change/mix styles
4. To be able to maintain stability, as GAN networks tend to easily lose stability

Which is why Nvidia's style based generator StyleGAN [12] and its evolved versions StyleGAN 2 and 3 are the most suitable GAN model for this case.
The StyleGAN architecture goes as followed:

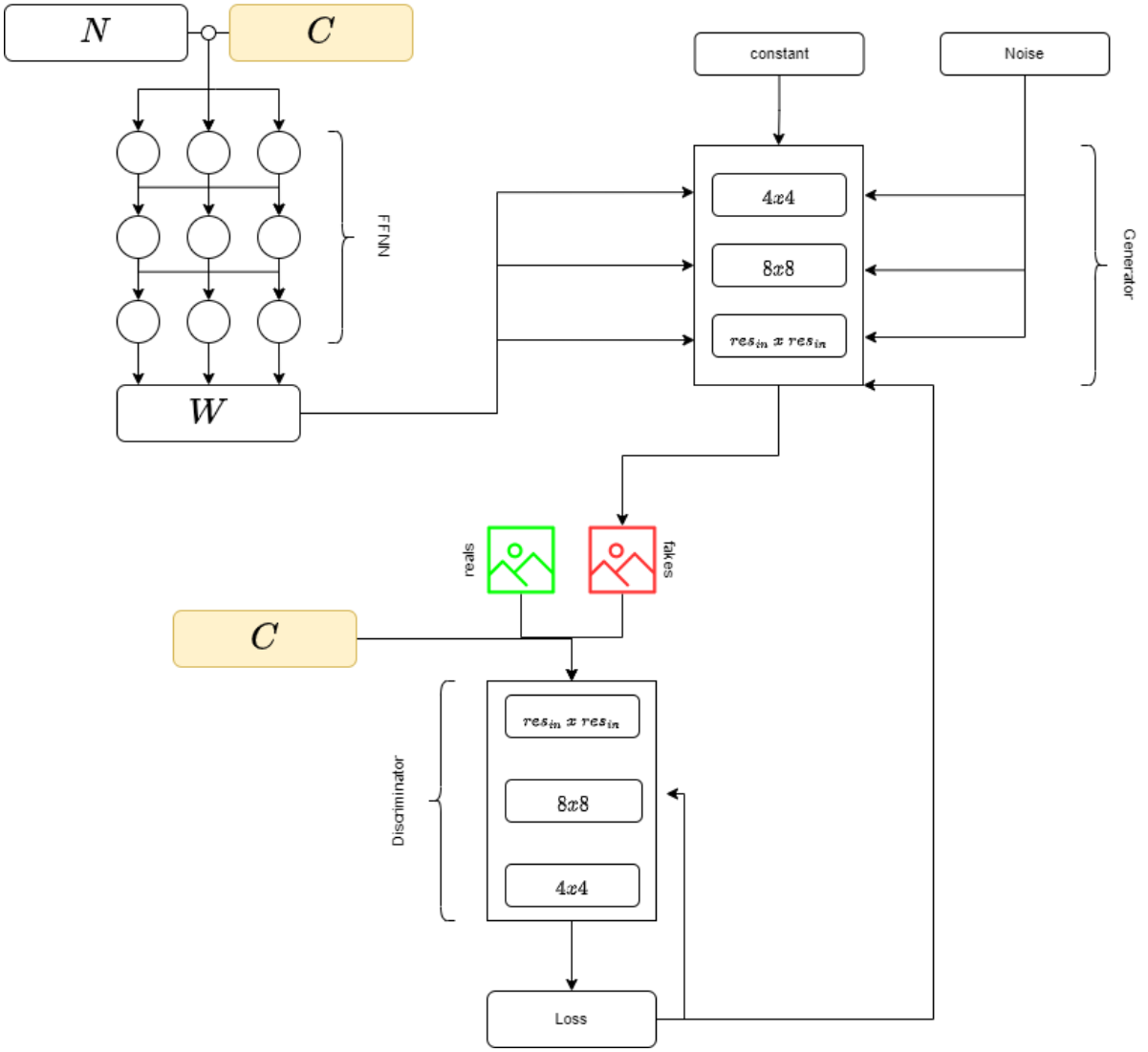

Image 4: overview of the StyleGAN architecture

A StyleGAN architecture can generally be split into two sections: The mapping network and the synthesis network while introducing the intermediate latent space W, whereas the synthesis network scales with the resolution, and is otherwise similar to the general GAN structure.

The mapping network

Every image can be translated into a vector of high dimensions with each individual value representing unique features it possesses, the image feature vector. The mapping network realizes that goal in order to be able to individually manipulate the vector N and therefore its high and low level features (which were defined as structural and detail changes, see 2.1). Instead of passing the N-space vector, which is a random vector composed by the normal distribution, directly to the generator, it will use a feed-forward neural network(FFNN), an 8 layers Multilayer Perceptron, to project N onto the intermediate latent space W in order to gain better feature disentanglement. It will allow access to each of the unique features presented in the vector within convolutional layers. If Ada-IN is applied, this will be possible for every resolution.

Adaptive instance normalization

As for the stabilization method, instead of the batch normalization which normalized all the feature vectors using the mean and the variance, instance normalization, which normalizes each individual sample, as well as its adaptive version which enables instance normalization for every resolution, was developed [13].

$$AdaIN(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sqrt{\sigma^2(x) + \epsilon}} \right) + \mu(y)$$

Where σ(*y*) is the standard deviation, σ²(*y*) the variance and μ(*y*), the mean, all computed along the spatial dimension as in Instance Norm.

Ada-In takes *W*, which represents the style mathematically, as input and with its normalization process, specifies the importance of each and every feature in the form of parameters held in the convolutional layers, therefore visualizing the feature vector.

Noise

Feeding noise to the synthesis network is meant to result in stochastic variation.
Stochastic variation is usually applied by utilizing a random noise vector, formed by Gaussian distribution, subsequently feeding them to each convolutional layer. This results in variation and change in these layers, as noise is applied to each pixel.

3.5 Post processing of generated results

After generating new glyph grids, it is necessary to adopt a few methods to realize the goal of creating a high quality font as StyleGAN is unable to guarantee the quality of details, thus resulting in post-processing steps to overcome the deficit. Major steps in this case are:

- The cleansing of disturbances in the image. Due to the StyleGAN architecture which operates with the continuous addition of noise, certain disturbances and imperfections are unavoidable, which is why one needs to remove the noise, scratches and unwanted spikes. By using standardized AI methods, one can easily achieve such a feat. As for the reduction of unwanted spikes, the problem will be solved through the conversion to a vector based image when finally creating the font since Bézier curves will automatically ignore small imperfections in spots where they do not belong, as they are too small to be considered a new line.
- Increasing the resolution of generated results. After cleansing the noise, it is favorable to enlarge the image to support and ease the process of turning the bitmap images into vector based glyphs. The more pixels for reference there are, the more accurate the trace will be. Empirical studies show that since the resolution during training is low, it will cause significant problems in both skeleton and detail, if one does not increase the resolution of the image with methods such as hyper-resolution AI models
- Enhancement of details:
  The appearance of a font is greatly influenced by the looks and amount of detail. Many more artistic fonts will tend to give longer lasting impressions. Empiric studies have shown that StyleGANs processing of details is greatly limited and uncontrollable, which is why we opt to process details using transforming glyphs into vector-based strokes using bezier curves after the generation, instead of attempting to solve the problem within the latent space

3.6 Proximity filtering with image feature vectors

Although every single generated result is in itself unique and entirely new, there are still cases where the results are vastly similar to the Ground Truth.
For a font to be considered a new unique font, there must be changes in the skeleton as well as in the details. Fonts too similar to the Ground Truth do not fulfill any purpose, which results in

greater value the greater the differences are, for it will be a new artistic creation.

For the goal of generating a font having major differences compared to the Ground Truth, we propose the following method:
To implement a filtering process based on image feature vectors. Using different proximity calculation methods, the score κ varies. Calculation methods include e.g. cosine or euclidean calculation. Depending on the method used, contrary to many other papers in which a closer proximity means that the generation was more successful, the score in this case must be above a certain value to be viewed as a successfully generated font, which means that greater distance is better, as it shows that the creation is composed of many different original styles, therefore having achieved the creation of an entirely new style in the best case.

4. Results

4.1 Raw results

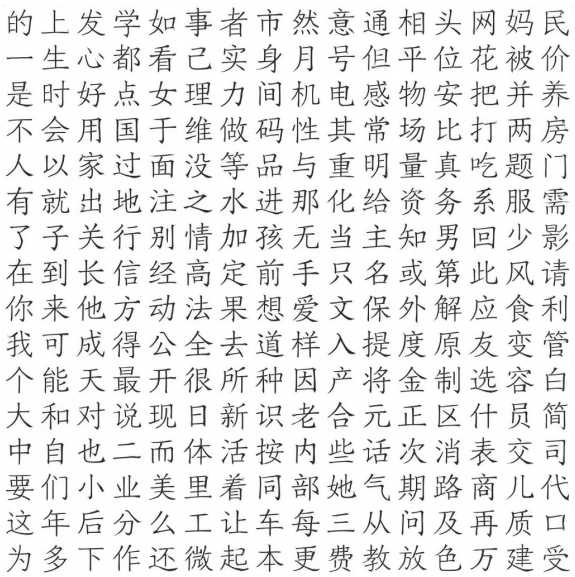

Image 5 : seed2598

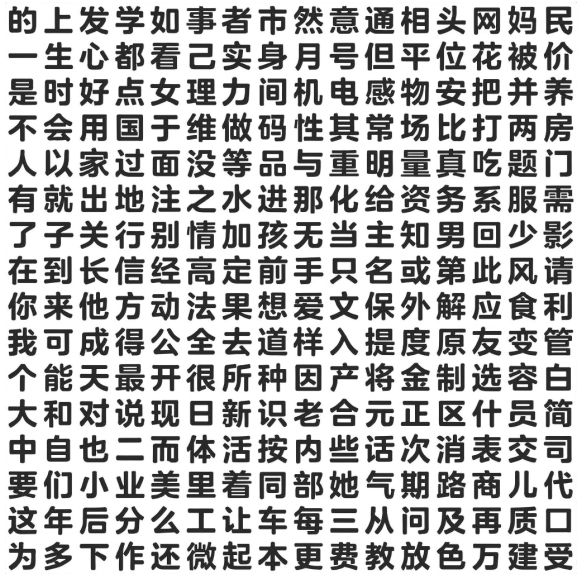

Image 6: seed2599

4.2 Quantitative evaluation

Our goal is to evaluate the results quantitatively. As of now, since no model has ever achieved the feat of fully generating a complete Chinese font or even only above a dozen glyphs at once, the quantitative evaluation is difficult in this case, especially so since this paper proposes an entirely new method. As such, we will make use of the inception score used to test GAN networks as well as the FID score. The inception score, defined as

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g} D_{KL}(p(y|\mathbf{x}) \parallel p(y)) \right),$$

which is the exponential of the Kullback-Leibler (KL)-divergence of image $p(y|x)$ where y is the set of labels and x is the image. The average score for all images will be the inception score, which can be used to determine the distinctness and variety of the results. A greater score represents a better result, as there is more variety and the quality will be higher.
The inception score is enhanced by the Fréchet Inception Distance(FID) commonly used in GAN papers and generative papers. A FID is defined as
$FID = \|\mu_r - \mu_g\|^2 + T_r(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$

which is a performance metric used to determine the quality of results. The lower the FID score is, the better are the results, since it means smaller distances between synthetic and real data distributions, whereas s score of 0 means that the image is equal to the Ground Truth. where the mean is represented by μ, and Σ symbolizes the covariance of the embedded layer of the training data r, and the generated data g.

The results are, that our FID score amounts to:

$FID = 68.7030$

A score neither too high nor too low. In this case, however, the score is proof of several points:

- A low FID score indicates how close the generated images are to the Ground Truth. While it may be better for other models, our goal is to generate new fonts, which is not achieved if the results resemble the original input too much.
- A high FID score means that the results differ from the Ground Truth, whereas a FID score too high means that the model generated results which have no resemblance with the Ground Truth. In our case, a higher FID score simply means better creativity on the model's part, as long as the glyphs are humanly recognizable, which is in fact true, as shown in the raw results (Image 5,6).

4.3 Qualitative evaluation

By subjectively evaluating the images, we will qualitatively judge the results. As such, we will use the truncation trick to evaluate. As we have pointed out earlier, the dataset will have areas of low probability density, thus affecting the generated results due to insufficient representation. To improve the results, research [14] has shown that by making use of a truncated space, which avoids low density regions. This is done by computing the center of mass through the following formula:

$$\bar{\mathbf{w}} = \mathbb{E}_{\mathbf{z} \sim P(\mathbf{z})}\left[f\left(\mathbf{z}\right)\right]$$

which has as its output the average image, the mean representing the center.

$$\mathbf{w}' = \bar{\mathbf{w}} + \psi\left(\mathbf{w} - \bar{\mathbf{w}}\right)$$

From the center, through this formula, it is possible to scale the deviation, making the style lean to one side or another. As we can see in this image, the two extremes are opposites, while the center is supposed to be the average image.

Through this image, we can analyze how well the training distribution was captured by the latent embedding and analyze the diversity of results.

Paired with the raw results shown above, it is safe to say that there are indeed many different variations of styles due to style mixing. Therefore, we conclude that the model is capable of producing new fonts in diverse styles.

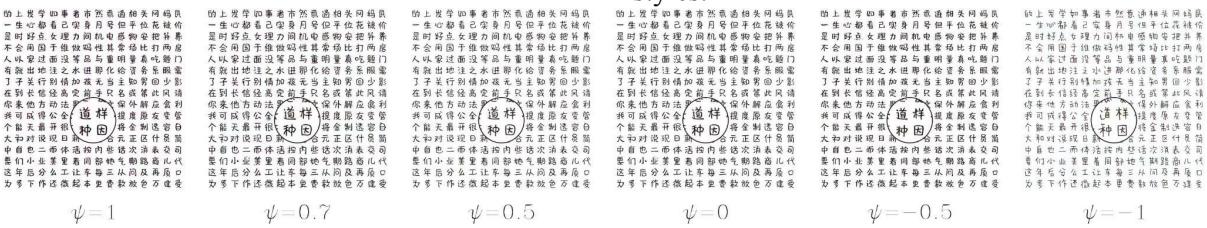

Image 7: The truncation trick showing different generated styles. Four glyphs are highlighted for better visual analysis

4.4 Analysis of creativity

Another way to analyze the generated results besides the application of the FID score is through the use of image feature vectors. As explained before, image feature vectors are in the sense of the latent space a vector which records all unique data of an image. By calculating both the feature vectors of our input and as our output, we can compare them. By using established methods (e.g. TensorFlow projector) we are able to make use of three different calculation methods to determine the proximity of the generated results. This, as seen

before, can be used to filter, but it can also be used to determine creativity, as greater distance means that the result is indeed different from the input and that there were major changes in the skeleton as well as in detail.

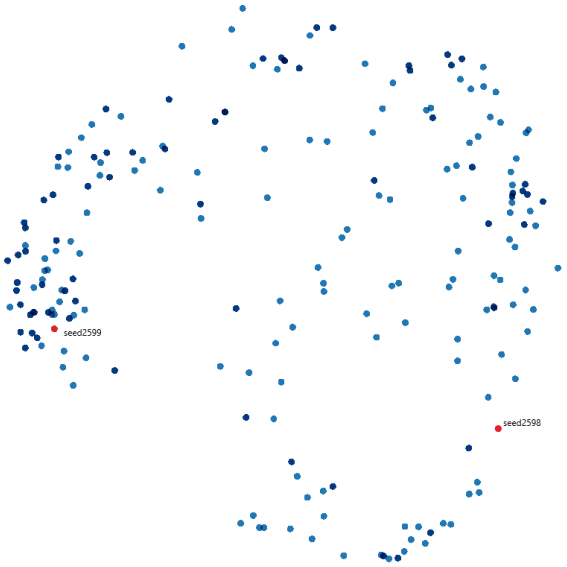

Image 8: Visualization of two generated results, where red dots represent the generated results and the blue dots represent the original input, displayed in a 2D space. The dots correspond to the generated images seen in 4.1

4.4 Use in actual design

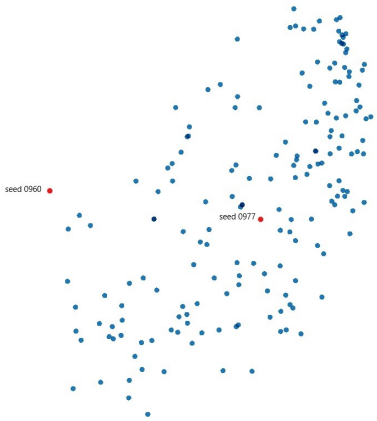

Image 9: proximity diagram for the following example

Image 10: Example Image of an advertisement using generated fonts

5.Discussion

In general, more changes in the skeleton of CJK fonts will be treated as more creativity. For example, the following ancient fonts are considered as a very creative font.

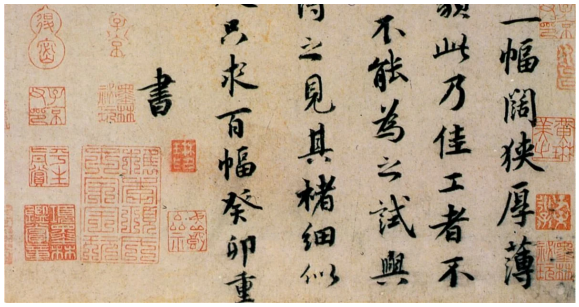

Image 11: ancient Chinese calligraphy

Although the model has already shown great results, achieving the complete generation of a large character set font in high quality for the first time, there is one obvious disadvantage:
The style had not yet reached a level where its creativity rivals that of humans. Although the results are satisfactory, they are ultimately still incapable of humanlike creation. As of now, every result is based on the style mixing of the input to bring about a combination. To improve on this point, it is highly possible that it could be accomplished with the use of the so-called Conditional StyleGAN. As it is shown in Image 4, we can see the existence of the parameter *C*, which represents the conditions.
Conditional StyleGAN is a method in which the author composes a condition, which will be fed into the StyleGAN architecture. There is no fixed model and every condition may vary greatly in form. For instance, the condition could be a parameter fed along with the input to the generator, or it could also be a certain favorable modification to the loss function, etc. In this case, there are points the condition might achieve.

- The Condition provides an environment where it can automatically improve on either the skeleton or the details or both on itself, therefore achieving the goal of increased creativity
- The Condition changes the architecture of StyleGAN into a more favorable and specialized environment, or to tune StyleGAN towards font generation, as StyleGAN is meant to be an image generating network, possessing many qualities a model tuned purely for font generation may ignore, such as the consideration of a background

For the first point, a method we perceive as reasonable would be to change the noise function of the StyleGAN architecture. StyleGAN uses a randomly generated pixel by pixel noise, which means that every pixel is randomly distributed step by step. In the case of fonts however, this approach easily creates some form of noise or gray areas that are not helpful in the creation of a font, since StyleGAN is meant to generate colored images. In the case of fonts however, this is not necessary, since all pixels must only be either black or white, therefore eliminating unnecessary noise, aiding the improvement of quality.

6. Conclusion

In this paper, we explored the possibility of utilizing StyleGAN for large character set font generation. We showed that through careful preparation of data, from collecting data and building datasets to putting images into an larger image as to ensure the equal style variation and mixing in the latent space, and meticulous post processing, that we were able to simultaneously solve the following issues which plagued this domain:

- The ability of generating massive amounts of glyphs
- The consistency of style, in both skeleton and detail
- The quality of images, to generate images which are clear and without blur or other noise
- The efficiency of production

Supported by our analysis of our generated results, which outperform any other method in those regards, we are now, for the first time worldwide, able to mass produce large character set fonts such as CJK fonts in qualities so high that they are commercially usable.

References

[1] Few-shot Font Generation with Localized Style Representations and Factorization Song Park, Sanghyuk Chun, Junbum Cha, Bado Lee, Hyunjung Shim,
[2] Multi-Content GAN for Few-Shot Font Style Transfer
Samaneh Azadi, Matthew Fisher, Vladimir Kim, Zhaowen Wang, Eli Shechtman, Trevor Darrell
[3] Few-shot Compositional Font Generation with Dual Memory
Junbum Cha, Sanghyuk Chun, Gayoung Lee, Bado Lee, Seonghyeon Kim, Hwalsuk Lee
[4] Image-to-Image Translation with Conditional Adversarial Networks
Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros
[5] Generating Handwritten Chinese Characters using CycleGAN
Bo Chang, Qiong Zhang, Shenyi Pan, Lili Meng
[6] FontRNN: Generating Large-scale Chinese Fonts via Recurrent
Neural Network Shusen Tang, Zeqing Xia, Zhouhui Lian, Yingmin Tang, Jianguo Xiao
[7] Learning to Write Stylized Chinese Characters by Reading a Handful of Examples Danyang Sun, Tongzheng Ren, Chongxuan Li, Hang Su, Jun Zhu
[8] Chinese Character Image Completion Using a Generative Latent Variable Model
In-su Jo , Dong-bin Choi and Young B. Park
[9] SCFont: Structure-Guided Chinese Font Generation via Deep Stacked Networks. Jiang, Yue & Lian, Zhouhui & Tang, Yingmin & Xiao, Jianguo.
[10] CalliGAN: Style and Structure-aware Chinese Calligraphy Character Generator
Shan-Jean Wu, Chih-Yuan Yang, Jane Yung-jen Hsu
[11] RD-GAN: Few/Zero-Shot Chinese Character Style Transfer via Radical Decomposition and Rendering, Yaoxiong Huang, Mengchao He, Lianwen Jin, Yongpan Wang
[12] A Style-Based Generator Architecture for Generative Adversarial Networks
Tero Karras, Samuli Laine, Timo Aila
[13] Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization
Xun Huang, Serge Belongie
[14] Large Scale GAN Training for High Fidelity Natural Image Synthesis
Andrew Brock, Jeff Donahue, Karen Simonyan