Universal Solution to Extension and Translation of Font Across Languages

Chan Lap Yan Lennon*, Luo Steven Tin Sui*, Kong Chi Yui, Cheng Shing Chi Justin
Artificial Intelligence Project Group
St. Paul's Co-educational College
Hong Kong
sp20175121@spcc.edu.hk sp20156401@spcc.edu.hk sp20187431@spcc.edu.hk sp20175331@spcc.edu.hk

ABSTRACT

There have been numerous attempts in font generation and extension through artificial intelligence. The majority of recent works have adopted the convolutional transfer approach with probability vectors representing characteristics of the character and font styles, which comes with the drawback of a greatly limited character pool. Therefore, we propose a model, namely Universal Solution to Extension and Translation of Font Across Languages, featuring pairwise comparison methodology between two reconstructed images, one through the target itself and another taken from two images in which one bears the font characteristics of the target and another that of the character. A spectrum of experiments has also been conducted on the structure of our end-to-end model to obtain comparative results, in hope of finding the optimal model at hand for further extension across languages. The experimental results have demonstrated the universality of this model, its effectiveness and potential for further development.

* These authors contributed equally

CONTENT

1. Introduction

As typing progressively replaces handwriting in our daily lives, technology of word processing has been gaining popularity, including handwriting style recognition and reconstruction. Capable of preserving font styles, this could on one hand assist investigative and forensic forces in identifying outlaws through their handwriting, and on the other cater to the rising demand of people wanting to construct their own fonts for entertainment purposes without the toil of writing all existing characters.

Thus far, despite availability of software capable of transferring styles from one language to another, their functionalities are limited. Recent attempts of font generation and extension [11][12] have been extracting strokes from input characters, which are then selected and assembled into a new calligraphy character. Largely dependent on the extracted strokes, the effectiveness of stroke extraction technology is dubious, given the complexity of certain characters and the difficulty of mechanical stroke separation in cursive font styles. Meanwhile, a significant number of studies opt for classifying characters with convolutional neural networks (CNNs) to render probability vectors containing information regarding the character [7][13]. However, this would greatly hinder the extension of the project towards different languages due to the fact that the size of the content vector is based on the size of the character pool in question.

As a vital part of written communication, typography in computers has been increasingly shed light throughout the last decade. Given that, allowing a font to be extended regardless of language is becoming increasingly significant as designers and developers develop their projects to various languages. In order to answer these demands, this project aims at enabling users to transfer their font style across different languages. However, including every character in a language is hardly feasible due to the limited number of fonts covering all characters in certain languages such as Chinese, resulting in excessive reliance of the model on the character dataset. As a result, new fonts could not be generated if the input character is not included in the dataset should the conventional methodology be adopted. Inspired by the style extraction in [17], we extended this to the content aspect to overcome the aforementioned obstacle. We propose an end-to-end deep neural auto-encoder which is able to encode the input I and generate output G(I) based on the ground truth T treating the transformation as an image-to-image transformation.

Various methods have been experimented in boosting the auto-encoder to minimize loss

$$L = |T - G(I)| \quad (1),$$

which includes (1) the implementation of another network D aimed to discriminate whether the generated result G(I) is close enough to T, where the loss to be minimized is

$$L = |D(T) - D(G(I))| \quad (2),$$

(2) restructuring of the input I which duplicates the input and delete one of the words in the input I in each duplication, and (3) the addition of a style extractor through the correlation of strokes that is represented by a vector generated by another network E in which the loss is calculated as follows:

$$L = \alpha|T - G(I)| + \beta|E(T) - E(G(I))| \quad (3).$$

2. Related works

Extensive research and studies have been done by various entities in attempts to utilize artificial intelligence in fostering font development, which could be generally categorized into font generation, font extension, and font translation.

2.1: Font Generation

Font generation refers to the rendering of completely new fonts, usually given a character set and a randomized vector as inputs to generate brand new fonts. This aspect has relatively been more developed, but most projects focused on the Latin or English alphabet. In image generation projects, font generation inclusive, generative adversarial network (GAN)[1] is frequently used, which consists of a generator to generate images and a discriminator to grade the quality of the images. A broad spectrum of GANs have been inspected, such as multi-content GAN where all 26 alphabets are generated given the style of five of them [2], GlyphGAN featuring its capability of creating legible, diverse, and consistent fonts [13], CycleGAN which specializes in cross-domain learning [10], DCGANs [9] and class discriminative variations which shows significant improvement when compared to conventional DCGANs [7]. The success of GAN attempts in English font style transfer, the aforementioned inclusive has driven that of Chinese, some by tech giants such as Alibaba, and networks are now capable of generating words embodying characteristics of both the character and the font.

2.2: Font Extension

On the other hand, font extension aims at reducing the work of font designers by creating the rest of the character set of the same language given a few samples of the font in question. This is suitable for languages with more complicated characters, such as Chinese. In this type of projects, it is of pivotal importance to ensure the ability of the network to differentiate style characteristics from that of content, where style represents the font style and content denotes the character. One prevalent approach is neural style transfer which combines the content of one image and the style of another, which includes the renowned vgg19 network [4] [5] for its gram matrices which is capability to detect intricate features thanks to its basis on the loss of specific genres, namely content, style and total variation. On the other hand, disentanglement methods may also be used, such as Kullback–Leibler divergence (KL divergence), an effective measure assessing the difference between two probability distributions. It has been proved to be apt for capturing characteristics of different font styles, even when involving languages with characters composed of complicated radicals and strokes. [17] While some projects would opt for Wasserstein distance [13], with the help of KL divergence, we measured the mismatch of the standard distribution between the images. An effective quantitative representation of the loss can therefore be obtained when feeding two sets of data, either containing the same font or the same characters, into the same extractor.

2.3: Font Translation

Font translation means the extension of fonts from one language to another, which is more of a challenge due to intrinsic differences in terms of composition and strokes between characters of different languages. This is also most

probably the reason for the scarcity of research and studies in this aspect. An attempt made by Fernandez from Stanford University [] investigated the ability of a neural network to transfer the characteristics of fonts from English to Japanese. While it successfully retains the visual characteristics of selected fonts, limitations exist in terms of its options of languages. This project therefore further looks into generating images of other languages while retaining the characteristics of old fonts. Equally noteworthy is [radical paper], which despite focusing on extension rather than translation, has introduced its methodology of intercross pairwise reconstruction, an approach that would increase the reliability of the content and style vectors and thus the consistency of font characteristics. Greatly inspired by this, we have extended the KL divergences to greater capabilities, training the network on two streams. KL divergences enable the network to extract the style from a font. We built an extractor so that we can extract both the style and content of the character, thus solving the problem of missing characters in our dataset. In the first stream an image is fed, with the objective of the network being able to reconstruct the same image, while in the second, two images are fed, and the network is trained such that the resultant image bears the font characteristics of one image and the character characteristics of the other. As a result, the network is ultimately able to identify and reconstruct characteristics of both font and character.

3. Data collection and preprocessing

Given the plethora of languages, we have attempted to select languages with characters manifesting distinct characteristics. The languages selected are Traditional and Simplified Chinese, English, Greek, Hebrew, and Japanese, both the hiragana and katakana letter systems inclusive. While for most languages we took into account the entire alphabet, the extensive scope of Chinese characters available made it hardly possible to take into account all characters. We have manually selected 250 Chinese characters as seen in Figure 3.1, holding the principle to include as many common components in Chinese characters as possible. After gathering more than 3600 fonts, the font files for each language are converted into images as a pre-processing for training. Figure 3.2 exhibits how each character in each font is processed from part of a TrueType Font file into images.

世丞丹久事　亮仁伊伍似　但侯俗偏停　健僵再凋凹　功劫包卵原
及受史吹命　咒哀品哉哲　唐嘉器四垂　埃培堊堡堵　填外夭奔套
妻娶嬉存家　密寇實尖尼　屯岡岳崇崩　巫差巴希帽　平年序庖庚
度庭廟弩彗　彩待德志怖　恩悅情想愁　慶憩成我或　戲房才折拋
拜挫掃描撒　擦故斑斯施　昏易晨有栗　桃條梵棺極　楷榮模橫檜
次歌此武死　殃段毒毫氓　氣泰港渺澀　滿潮澀炮照　爵犯狗玩琴
甘申男疑疼　痰皇盤睡破　硬確福禽稱　童第簿糖統　經緩缺罩羲
翹考耐耗聚　肅肇脅腐臂　臭舂舅舍舞　艘艱色菠蓉　蕊藕虐蜀蜥
融行衡補裝　褐覆觸訊誘　諷謬譽豫賢　赤路身轉辣　逸遞酒野錯
長關隊雅霸　霾靠韋顫飯　騷骨魁魚鸞　麟麾黍黝鼠　鼾齒龍龜龠

Figure 3.1: Chinese Characters selected (Traditional variation adopted)

3.1 Text plotting

To ensure the neural networks are taking purely the characteristics of the characters and fonts into account, the characters are plotted in black on a white canvas. Instead of plotting the character on (0, 0), we plotted it on (50, 50) due to the offsets certain characters have that would otherwise render it unable to show the full character.

3.2 Boundaries trimming

One factor observed that would affect the performance of the neural network is the size of the fonts. Even when given the same "font value", different fonts may have varying actual size for the same character in terms of pixels, making it another obstacle that the neural network would have to face.
Therefore, the x-coordinates of the leftmost and rightmost black pixels, as well as y-coordinates of the uppermost and lowermost black pixels are searched across each image to determine the boundaries of the rectangle inscribing the character. Only the black pixels, rather than including the grey pixels around some parts of the borders of the characters, are considered part of the character to avoid a character with excessively thick lines that would otherwise blur the image and hinder the neural network from identifying crucial features of the images.

3.3 Size Correction

The length of the longer side of the rectangle which describes the character is then taken as the side length of a new square canvas. The trimmed character is justified onto the new canvas to render the image with a square shape. The square images are then resized to (54, 54) to retain consistency of data size. Margins of 5 pixels on all sides are also added, resulting in images of size (64, 64).

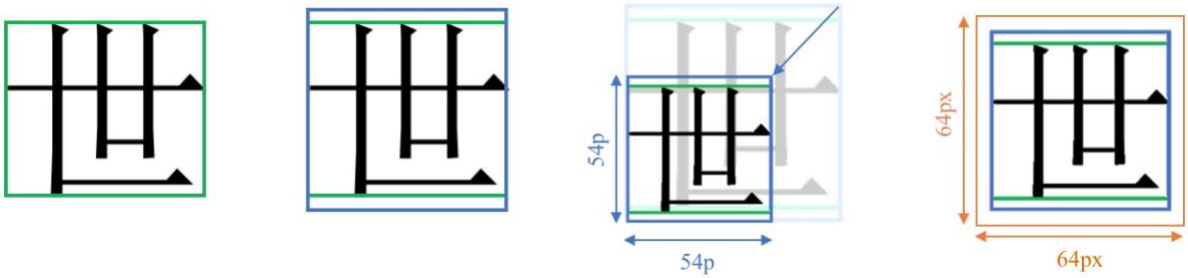

Figure 3.2: From left to right: text plotting & boundary detecting;
square shape rendering; resizing of image; margins adding

4. Methodology

4.1 Stage 1: Multi-content convolutional neural network with style vector (McGNN with style vector)

We did experiments with different models. A flaw was encountered, that the words under in the bottom of the input do not show any shape of a word, is due to the smaller size of convolution receptive fields than the image length within a reasonable number of convolutional layers, the generator fails to capture correlations among the different

independent Chinese words for those far away from the given word images, which are those in the bottom of the input. In order to tackle this, we decide to preprocess the input images for better learning. The product of the network without the discriminator is better than the product of multi-content GAN, but the words from different fonts, though are able to capture some style from the input, the difference between different generated products are not significant enough visually. To solve this problem, we applied the style disentanglement of [5] (disentanglement of content and style of an input image is used when training upon a style transfer of an image from its original style to the style of a given image) to our method, by building a style classifier to produce a style vector of the given word image.

4.1.1: Evaluation

Output 1. Target 1 Output 2 Target 2 Output 3 Target 3 Output 4 Target 4 Output 5 Target 5

(missing words due to incompleteness of data)

The network is able to reproduce detailed shapes of a word. Take ‘擦’ as an example, the lower-right component of the word has a clear structure in output 2 and 3, which is detailed enough to show a significant clarity. This shows that the network is deep enough to capture the small features of the Chinese words and well reflected on the generated results.

As for some of the missing words from the results (Output 2 and 4), it is due to the missing of the word in those specific fonts we gather. But the products with missing words still perform as good as the ones without missing words, showing the ability of the network to neglect the missing words and capture the style of a font as a whole instead of only generalizing on an estimated location of black pixels, further proving the network's deepness to capture correlations of the exact content of each Chinese words.

There are limitations for this model. We are not able to generate too many words with this model. It decreases the efficiency of this project. In order to solve this, we came up with a new architecture.

4.2 Stage 2: Final Model

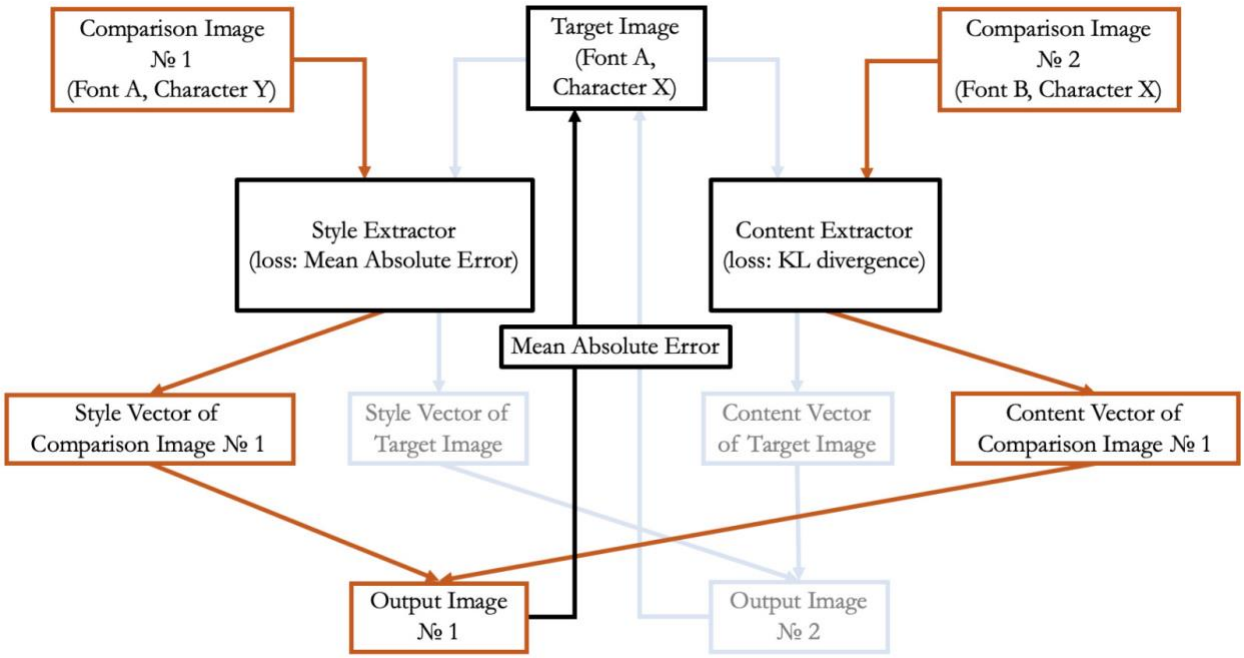

Figure 4.2.1 (Flow of final model)

In view of the lacking universal representation of the previous model, where only the designated number of words (in this case 25) can be generated but no more than these 25 words, we have attempts to increase the number of words which it can generate following the original architecture, but fail due to the lack of computing power. Therefore, accompanied by the limitation that restricted language can be trained, we have developed a new architecture to tackle such problems, in which only a content vector and a style vector is needed in order to generate other words with the handwritten samples give that we can also input the other words with another style of fonts. Former auto-encoders either have no constraints on the encoders and the decoders, training them at once at the same time [3], or they assign specific tasks to the encoders, changing their nature into classifiers to classify input images into different classes. [4] As for us, as our task involves two images as input, one providing the content information, one providing the style, a simple auto-encoder cannot meet such requirements. Moreover, as our task involves generating words of multiple languages, unlike the general image generation like faces [15] or bedrooms [16], multilingual generation is also challenged by the large dictionary and rigorous structures. Our adoption of two

extractors without executing trivial classification task but instead, trained through its ability to encode the images' information and their separate ability of encoding content information and style information is largely through the comparison of the vectors generated by the extractors, where the two vectors extracted by the content extractor from the two images with same content, and the two vectors extracted by the style extractor from the two images with same style should be the same [16] . Under such kind of inter-cross pairwise training[16], the two extractors are trained to both be able to encode the input image and to perform tasks to extract content or style information. Through using the trained extractors and decoder, we can input 2 words to this network, one taking its content through the content extractor, one taking its style through the style extractor, and decode them to form an image with the content of the former and the style of the latter. (Figure 4.2.1)

Using this method of combining content and style from two words, it is expected that even if the two words come from two different language (e.g., Chinese and English), they can still be able to combine as each extractors are only responsible for extracting one feature (either style or content) of them. While the content extractor recognizes the Chinese word image to be a certain word (e.g. 一), the style extractor only recognizes that the English Word image is of a certain font(e.g. Times New Roman), thus combining these two information together, the decoder can generate a Chinese word with an English font, thus achieving cross-language word-font generation [17].

4.2.1: Model architecture

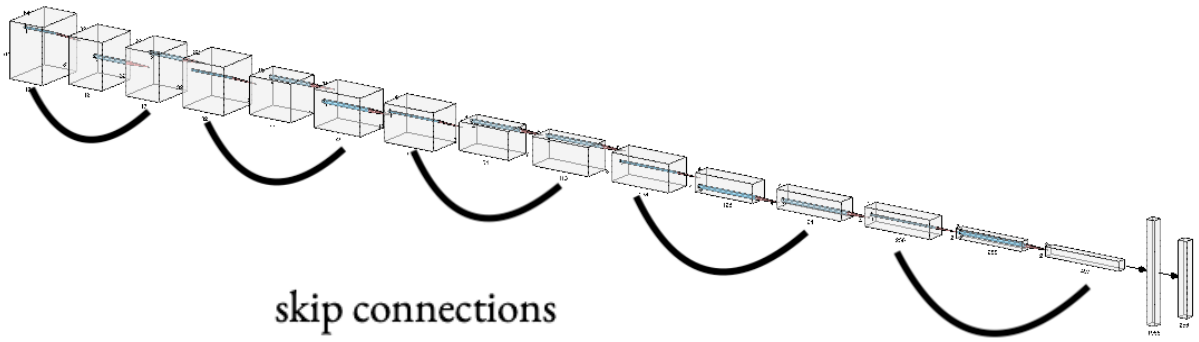

Figure 4.2.2 (model architecture of the content extractor)

The style extractor is a convolutional neural network. It extracts style information of a word through the image of the word and its content vector. The image of the word is fed into 5 resnet block which consists of a skipped layer of 2D convolution with a filter size 1x1, followed by a 2D convolution layer with a filter size of 2x2, ended with a 2x2 max pooling layer. After feeding through the resnet blocks, the output of them is flattened and connected to a fully-connected dense layer of 256 nodes, generating the style vector output, representing the data's style. Throughout the model, batch normalization [6] and leaky-relu activation [8] is used.

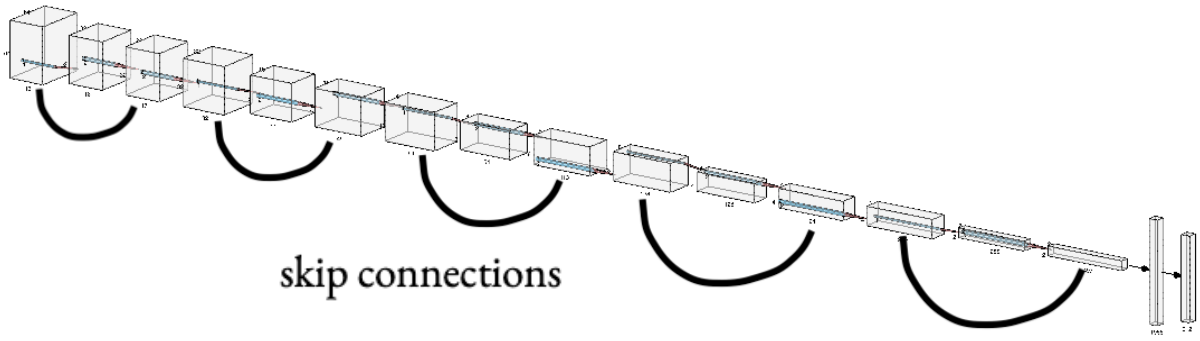

Figure 4.2.3 (model architecture of the content extractor)

The content extractor has the similar structure as the style extractor, but it extracts content information of a word through the image of the word. The image of the word is fed into 5 resnet blocks which consists of a skipped layer of 2D convolution with a filter size 1x1, followed by a 2D convolution layer with a filter size of 2x2, ended with a 2x2 max pooling layer. After feeding through the resnet blocks, the output of them is flattened and connected to a fully connected dense layer of 512 nodes, generating the content vector output which represents the data's content. Throughout the model, batch normalization [6] and leaky relu activation [8] is used.

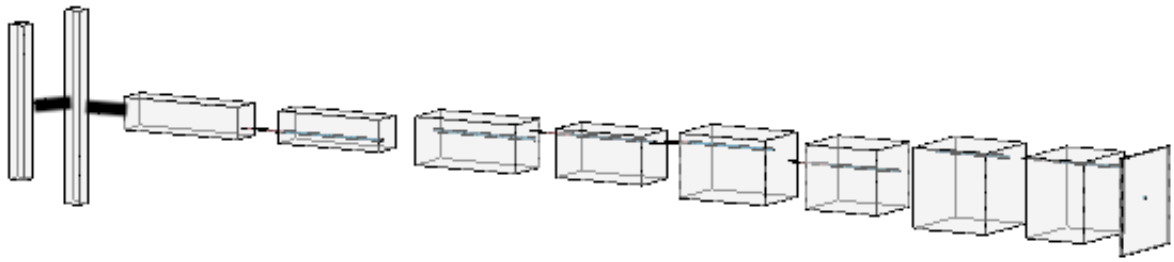

Figure 4.2.4 (The decoder)

The decoder decodes the content vector and style vector into a word with the content represented by the content vector and style represented by the style vector. The input vectors are concatenated before putting into a dense layer of 4096 nodes, which is then reshaped The image of the word is fed into 5 resnet blocks which consists of a skipped layer of 2D transposed convolution with a filter size 3x3, followed by a 2D transposed convolution layer with a filter size of 2x2. After feeding through these layers, the output has the same shape as the image input of the content extractor as well as the style extractor. Throughout the model, batch normalization [6] and leaky relu activation [8] is used.

Figure 4.2.5 (Main Image)

Image with the same style as the main image but with different content Image with the same style as the main image but with different content

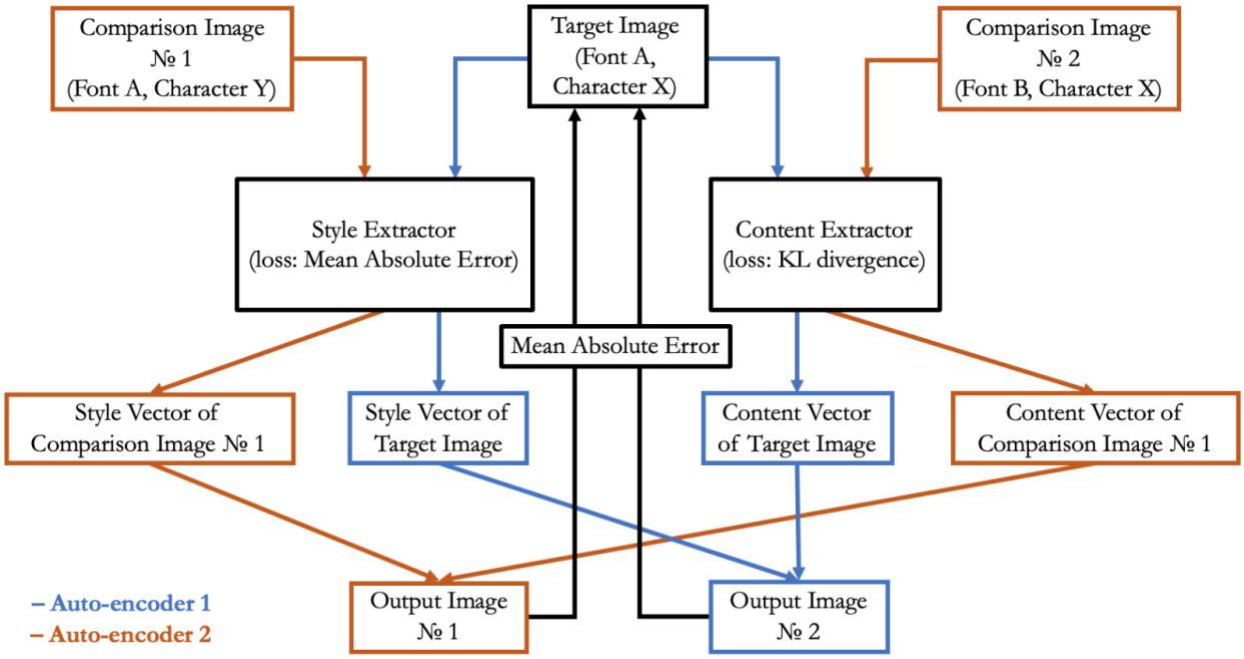

Figure 4.2.5 (Flow of auto-encoder)

In our end-to-end network, we train our style extractor, content extractor and decoder simultaneously through the auto-encoding and decoding of images. We adopt the inter-cross pairwise training method (Figure 4.5?), where we not only train our autoencoders on the main image, but also with the two comparison images as shown above. And between the two auto-encoders, as the content vectors of the main image and comparison image of the same content come have same contents and the same applies to the style extractors, they should share the same distribution of content information and the same mean of this distribution and so should the style information, therefore, we compare the content vectors and style vectors from in the middle of the two auto-encoders, training the two extractors.

First, we first feed our main image W_n (Figure 4.51) into the style extractor S, extracting style $S(W_n)$, then adding the content vector of W_n with content extractor C, $C(W_n)$, is decoded through the decoder D, generating image $D(S(W_n), C(W_n))$, which is then compared to W_n itself in a pixel-to-pixel manner, taking the mean absolute error between them as the loss to minimize, thus training the encoding ability of the extractors to encode the input image and the decoding ability of the decoder.

Similarly, we also feed a comparison image that is of different content but the same style, V_n (Figure 4.53) into the style extractor S, extracting style $S(V_n)$, then adding the content vector of another comparison image that is of same content of different content which is $C(U_n)$ (Figure 4.52) from that image U_n with content extractor C, is decoded through the decoder D, generating image $D(S(V_n), C(U_n))$, which is then compared to W_n in a pixel-to-pixel manner as $S(V_n) = S(W_n)$ and $C(U_n) = C(W_n)$, taking the mean absolute error as the loss to minimize.

On the other hand, we also compare style vectors $S(W_n)$ to $S(V_n)$ using Kullback–Leibler divergence to take the loss of the style extractor and train it by minimizing the loss, the same applies to the content vectors, where we compare content vectors $C(W_n)$ to $C(U_n)$ using Kullback–Leibler divergence to take the loss of the content extractor and train it by minimizing the loss.

4.2.2: Training

We use SGD optimizer [14] to optimize our model across a computing cluster. We use batches of 64 examples, initialized with learning rates of 0.0001 with a momentum of 0.9. SGD, whereas it will be halved every 5 epochs In our end-to-end network, our loss weight for the 2 auto-encoder mean absolute loss has a weighting of 1.0, as it is of crucial importance that the training of the two extractors based on the pairwise comparison between the content vector and style vector by the main word images and the comparison word images do not outweigh that of its ability to encode the word for the decoder to decode. While the pairwise comparison have a lower weighting than the auto-encoder itself, the weight of loss of the pairwise comparison of style vector should be higher than that of content vector as style vector is more character-invariant, and have more information associated with it than content vector. Thus, the weight of pairwise loss of the style vector is larger, which is 0.5, whereas the weight of pairwise loss of the content vector is 0.2.

There are 4 loss function in which we are training the model upon, the first one is the auto encoding and decoding of the main images, and is compared with the main images itself pixel to pixel and take the mean absolute error:

$$L(D(S(W_n, C(W_n)), C(W_n)), W_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} |z_{ij} - x_{ij}| \quad (4),$$

where W_n and z is the ground truth, $D(S(W_n, C(W_n)), C(W_n))$ and x is the output of the decoder, n is the length and width of the output and ground truth.

The second one is the auto encoding of the content vector of images with the same content but different style with the main images accompanied by the style vector of images with the same style but different content with the main images and decoding of the content vector of images with the same content but different style with the main images (and also the other way round), and is compared with an image of the content of the former and style of the latter. itself pixel to pixel and take the mean absolute error:

$$L(D(S(W_n, C(W_n)), C(W_n)), W_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} |z_{ij} - x_{ij}| \quad (5),$$

where W_n and z is the ground truth, $D(S(W_n, C(W_n)), C(W_n))$ and x is the output of the decoder, n is the length and width of the output and ground truth.

The remaining two loss functions are to intercompare the content vectors and style vectors generated by the main images and comparison images.

The third one is the loss between content vectors of the main images and the content vectors of the images with the same content as the main images but of different style, which is compared using the Kullback-Leibler Divergence

[16] between the two batches of content vectors. As the two batches of content vectors come from images with the same contents, they should share the same distribution of content information and the same mean of this distribution. Therefore, using Kullback-Leibler Divergence can help to train the content extractor to be able to only attract content-associated features from the input images but not style-associated features. The loss is expressed as follow:

$$L(C(W_n), C(U_n)) = zlog(z / x) + xlog(x/z) \qquad (6),$$

where $C(W_n)$ and z are the main images' content vectors, $C(U_n)$ and x is the content vector of the image with the same content and different style.

Similarly, the style extractor is also trained in such a way to ensure that it is only the style-associated features that are extracted from the input images. Therefore, Kullback-Leibler Divergence is used to compare the style vector of the main images and the style vector of the image with the same style but different content as the main images. The loss is expressed as follow:

$$L(S(W_n), S(V_n)) = zlog(z / x) + xlog(x/z) \qquad (7),$$

where $S(W_n)$ and z are the main images' style vectors, $S(V_n)$ and x is the style vector of the image with the same style and different content.

4.2.3: Evaluation

We achieve a decoder loss of 0.04 (for generation from main word images), 0.1(for generation from images with same content but different style and images with same style but different content), 0.01(for K-L Divergence of content extractor), 0.01(for K-L Divergence of style extractor)

Figure 4.2.6 Figure 4.2.7 Figure 4.2.8

From left to right: ground truth, image generated from ground truth taking both its content and style, generated image, word image in which content is wanted, word image in which style is wanted

The network is well able to extract thickness features of strokes from the word fed into the style extractor, from figure 4.2.6, the rightmost word 'l' which we want the style has thick strokes while the second-rightmost word 'm' which we want the content has thin strokes, the model successfully ignore the thickness feature of the second-rightmost word but retain the thickness from the rightmost word as shown from the middle word, which also resembles the leftmost ground truth. Therefore, it shows that the network is well able to extract thickness information, one of the style informations. While in figure 4.27, the model is able to thicken the 'k' but is not able to comprehend the partial thinness presented in the rightmost 'a' and cannot implement into the 'k'. In figure 4.28, it is an example of model failure in which the 'i' is too thickened that the two parts of the i becomes one.

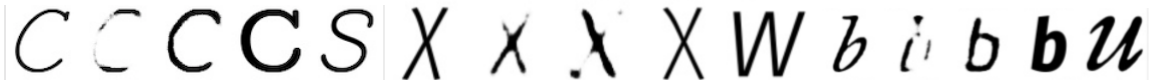

Figure 4.2.9 Figure 4.2.10 Figure 4.2.11

From left to right: ground truth, image generated from ground truth taking both its content and style, generated image, word image in which content is wanted, word image in which style is wanted

The network is also able to extract orientation features of strokes from the word fed into the style extractor, from figure 4.2.9, the rightmost word 's' which we want the style is tilted to the right while the second-rightmost word 'c' which we want the content is not tilted, the model successfully ignore the orientation feature of the second-rightmost word but retain the tiltness from the rightmost word as shown from the middle word, which also resembles the leftmost ground truth. Therefore, it shows that the network is well able to extract thickness information, one of the style information out. While in figure 4.2.10, the model is able to tilt the 'x' but shows model failure of broken strokes. In figure 4.2.11, the 'b' is able to tilt like 'u' but is unable to tilt in the extent of 'u' and is far from the tiltness of the leftmost ground truth 'b'.

Figure 4.2.12 (Greek example 1) Figure 4.2.13 (Greek example 2) Figure 4.2.14 (Japanese) example

From left to right: generated image, word image in which content is wanted, word image in which style is wanted

The network is well able to generate words with two images from different languages. For example, from figure 4.2.13 , it is shown clearly that multilingual word generation is possible, though with flaws as presented in figure 4.2.12 and 4.2.14 that the foreign style may alter the shape of the content in a manner that the client's own font would change vigorously that the image might blur.

On the other hand, the generated words have a high resolution, where strokes are clear without edges and are as smooth as the input words.

Due to the nature of the network that any two words can be combined into one word with the content of the former and the style of the latter through this network, this well-trained network can take in any combinations of the two words. Therefore, theoretically, the number of words it can generate is unlimited as long as you can provide the two words you are willing to extract their content and style. Therefore, this makes this network universal. It can be reduced to task of generating words of a particular written font style by some human by inputting the one's written word into the style extractor and input other words of a standard font into the content extractor, thus able to create the whole bank of words with the one's font style with one-shot capture.

But when the words have an orientation that is not 0 degree, the results cannot follow to change the orientation of the ground truth. For example, in output 3, the Chinese words do not have an orientation of 0 degree. But in the generated results, the orientation is not reflected although the thickness, relative location of strokes in a word is

well reflected. The reason for this is maybe because most of the fonts have standard orientation of 0 degree, and the rest of the fonts have their orientation features cancelled out by each other during the learning of the net. Therefore, the orientation of all results are of 0 degree.

It is also observed that some of the strokes of the more complicated words diffuse out into the background, for example, the last word of all the outputs above. It might be due to the complexity of the formation of the word, that different strokes in different words bear a greater difference in relative location and thickness than the different strokes in different words with simpler formation such as the first word of all the outputs above, which cause the inability of the network to fit those words for generic style generation. Accompanied with the observation of the blurring of the more complicated shapes in a word, we too observe that the style with words having thinner strokes tend to have a blurred image. This might be due to the fact that thinner strokes will easily have the strokes broken in the generalization of the network.

In a more abstract manner of grading the result of the network, it is observed that an important technique in calligraphy, which is the presentation of stroke direction (筆鋒) is missing. The network is incapable of capturing this, especially at output 1, where the low resolution even makes the strokes to not have a sharp shape. In Chinese calligraphy, each stroke should have its starting point and ending point, in which the starting point is broader and the ending point is narrower, but looking at all the outputs of our generator, it is observed that there is no such a pattern in the strokes of one end being broader and one end otherwise. This might be due to the small size of the input image not being big enough to capture the small features of each stroke which causes the inability to present the stroke direction.

5. Conclusion and future work

First, concerning the thickness feature, to make the generated results not to be too thick that different strokes in the same word merge into the same stroke, we can extract the content features of the output once again using the content extractor and thus training the generated word to retain its content information, thus avoiding the merging of strokes as merging of strokes changes its content.

Then, concerning the orientation feature, to make the generated results to have the same orientation as the ground truth, we may try add another orientation net [2] that evaluates on the orientation of the result to transform the words in the image to have the same orientation as the words in that font.

Next, to eliminate the diffusion of strokes to the background, we may add another attribute of the word array for loss comparison, which we call total variation loss, this forces the difference between the stroke and the background to be significant so as to prevent diffusing out. The loss of two consecutive pixels can be represented as

$$L(n, n+1) = log(1 - |P(n) - P(n+1)|) \quad (8)$$

, where L(n, n+1) is the total variation loss of two consecutive pixels n and n+1 and P(n) denotes the pixel value of n.

In response to the inability of the network to present the directions of stroke, we may add another convolution neural network which is able to distinguish each stroke from each other and learn the basic stroke directions of each stroke and apply the stroke directions on the product. The stroke direction loss to be minimized on can be represented as

$$L_{\text{stroke direction}}(G(x)) = |S(z) - S(G(x)| \quad (9)$$

, where $L_{\text{stroke direction}}(G(x))$ is the stroke direction loss of the product G(x) generated by generator network G and input x, where S(z) is the stroke representation vector of ground truth z and S(G(x) is the stroke representation of product G(x).

In conclusion, we demonstrate the potential and ability of a deep auto-encoder with multi-content with manual style loss minimizing is able to generate a whole set of Chinese words with the style of the style of the given few words. Although there are still improvements to be made, we nonetheless demonstrate the possibility of generalizing our method of style generation of Chinese words and to the extent, all words that are composed of strokes and originated from pictures.

6. References

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).

[2] Azadi, S., Fisher, M., Kim, V. G., Wang, Z., Shechtman, E., & Darrell, T. (2018). Multi-content gan for few-shot font style transfer. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7564-7573).

[3] Lyu, P., Bai, X., Yao, C., Zhu, Z., Huang, T., & Liu, W. (2017, November). Auto-encoder guided gan for Chinese calligraphy synthesis. In 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) (Vol. 1, pp. 1095-1100). IEEE.

[4] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[5] Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576.

[6] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[7] Abe, Kotaro & Iwana, Brian & Holmér, Viktor & Uchida, Seiichi. (2017). Font Creation Using Class Discriminative Deep Convolutional Generative Adversarial Networks. 10.1109/ACPR.2017.99.

[8] Xu, B., Wang, N., Chen, T., & Li, M. Empirical evaluation of rectified activations in convolutional network. arXiv 2015. arXiv preprint arXiv:1505.00853.

[9] Grödl, M., & Resl, M., AIfont: AI-generated Typeface - Generating new Typefaces with Deep Learning

[10] Narusawa, A., Shimoda, W., & Yanai, K., Font Style Transfer Using Neural Style Transfer and Unsupervised Cross-domain Transfer

[11] Xu, S., Lau, F. C., Cheung, W. K., & Pan, Y. (2005). Automatic generation of artistic Chinese calligraphy. IEEE Intelligent Systems, 20(3), 32-39.

[12] Xu, S., Jiang, H., Jin, T., Lau, F. C., & Pan, Y. (2009). Automatic generation of Chinese calligraphic writings with style imitation. IEEE Intelligent Systems, (2), 44-53.

[13] Kohtaro Abe, Seiichi Uchida (2019) GlyphGAN: Style-Consistent Font Generation Based on Generative Adversarial Networks. arXiv:1905.12502

[14] Saad, D. (1998). Online algorithms and stochastic approximations. Online Learning, 5, 6-3.

[15] Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. In Proceedings of the IEEE international conference on computer vision (pp. 3730-3738).

[16] Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., & Xiao, J. (2015). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365.

[17] Sun, D., Ren, T., Li, C., Su, H., & Zhu, J. (2017). Learning to write stylized chinese characters by reading a handful of examples. arXiv preprint arXiv:1712.06424.