Cross-Lingual Font Style Transfer

Peiyu Liao
Department of Computer Science
Stanford University
pyliao@stanford.edu

Fang-Yu Lin
Department of Electrical Engineering
Stanford University
fangyuln@stanford.edu

Abstract

This project aims to investigate the possibility of font style transfer under a cross-lingual setting. The dataset contains nearly fifty thousand Chinese characters as content images and five hundred English fonts as style images. Three different methods, Neural Style Transfer, Image Transformation Network, and CycleGAN are implemented, with the main focus on CycleGAN. Several training techniques are also applied to improve CycleGAN's performance, such as image cropping and content loss. From our experiments, CycleGAN significantly outperforms the other two methods, achieving better content and style evaluation scores in general. CycleGAN with a larger content loss weight has achieved an OCR accuracy of 28.93%, style loss of 2.3138, human content score of 5.0 out of 5, and human style score of 4.6 out of 5. Moreover, an analysis conducted on the training techniques shows that image cropping and content loss are critical to the performance.

1. Introduction

Designing a new Chinese font style is costly and inefficient since a standard font file contains nearly fifty thousand characters. While there are plenty of successful previous works on Chinese font style transfer that helps to apply a new font design to the whole set of characters with just a few font characters as seed, it often still requires a new Chinese font style to be designed.

In this project, a cross-lingual font style transfer model is built to apply existing English font styles to Chinese characters by image style transfer. This way, an existing font design of any language can potentially be applied to any other language characters in the world.

More explicitly, the inputs would be English fonts with the preferred style, and plain Chinese characters as the content to be applied with the style. The inputs are fed into a style transfer network to output Chinese fonts with the English font style.

Three style transfer methods will be experimented with. The first one is neural style transfer [3] based on image iteration which is slow at generation time, but it is implemented as the baseline for its simplicity. The second one, image translation network [9], is an extension that is based on model iteration, which takes more time to train but generates images more efficiently. The third one is CycleGAN [17], which is also based on model iteration but with a different training methodology, and is often adopted in recent font style transfer implementations for its good-quality results [1]. Several improvement techniques are devised and incorporated into the models, which is introduced in Section 3. The experimental results are shown and discussed in Section 5.

The project code is available on GitHub[1].

2. Related Work

Image style transfer methods can be generally divided into two classes, which are based on image iteration and model iteration respectively [8]. Image iteration methods such as neural style transfer [3] update the input image pixel by pixel through backpropagating content and style loss to achieve the styled image, which generally produce images of better quality but are computationally expensive at generation time.

On the contrary, model iteration methods optimize parameters of a generative model so that generation can be efficient, but the training process is often more difficult [1]. Image transformation network [9] is based on model iteration and is similar to neural style transfer, but is preceded with a transformation network to map an input image to a generated styled image, and the loss is backpropagated into the transformation network for optimization.

More recent techniques that have seen outstanding results are based on Generative Adversarial Nets (GAN)s [5], which trains a generator along with a discriminator through

[1] GitHub: https://github.com/pyliaorachel/cross-lingual-font-style-transfer. Neural style transfer and image translation network modified from Desai's work [2]. CycleGAN modified from https://github.com/aitorzip/PyTorch-CycleGAN.

a mini-max game. Pix2pix [7] is an image-to-image translation method based on conditional GAN, a variant of GAN that takes a condition as input to generate the paired output, that has seen successful results in a variety of image translation tasks. Zi2zi [16] extends on pix2pix to incorporate a category embedding so that multiple styles can be learned and generated by the same network. The work is also based on the Chinese font style transfer task.

Pix2pix and zi2zi both require paired inputs from the two domains. CycleGAN [17], on the other hand, is able to learn image-to-image translation with unpaired data. Two generators are trained in a cycle enforced by a cycle consistency loss, one mapping from input to output while the other mapping backwards. Chang et al. [1] utilizes CycleGAN to transfer handwritten Chinese font styles, which is most similar to our work.

However, in previous works, either a paired Chinese font dataset is required as in zi2zi [16], or a new Chinese font style needs to be designed for a few seed fonts to train the model, as in Chang et al [1]. In this project, we explore a more general option that transfers font styles cross-lingually so that no new font designs are required. We will show that this is indeed a more challenging task and how the challenges can be dealt with.

3. Methods

In this project, neural style transfer [3] is implemented as the baseline method for its simpler training process. Image transformation network [9] which generates results more efficiently are also implemented as a baseline model-based method. The main efforts will be put on constructing and optimizing the CycleGAN model [17].

3.1. Neural Style Transfer

In neural style transfer [3], a style image s, content image c, and initial output image y are fed-forward through a deep convolutional neural network, and the loss is based on the differences between the output image and the style or content image of each intermediate feature representation that comes from each network layer.

The total loss $\mathcal{L}_{\text{total}}$ to minimize is defined as a weighted sum of style loss $\mathcal{L}_{\text{style}}$ and content loss $\mathcal{L}_{\text{content}}$:

$$\mathcal{L}_{\text{total}}(s, c, y) = \alpha \mathcal{L}_{\text{content}}(c, y) + \beta \mathcal{L}_{\text{style}}(s, y) \quad (1)$$

where α and β are the weights for each loss.

The content loss is the sum of squared error losses of each feature representation defined as:

$$\mathcal{L}_{\text{content}}(c, y, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - C_{ij}^l)^2 \quad (2)$$

where F is the feature representation of y at layer l, C is the one for c, and F_{ij}^l is the activation of the i-th filter at each pixel position j in layer l.

In order to have a non-localized style feature representation, the feature correlations between the filters of F^l is defined as the Gram matrix G^l:

$$G_{ij}^l = \sum_k F_{ik}^l F_{kj}^l \quad (3)$$

The style loss is then defined as:

$$\mathcal{L}_{\text{style}}(s, y, l) = w_l \frac{1}{4 N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - S_{ij}^l)^2 \quad (4)$$

where w_l is the weight for layer l, N_l is the number of filters in layer l, M_l is the product of the height and width of the feature map in layer l, G is the Gram matrix of F, and S is that for the feature representation of s.

3.2. Image Transformation Network

Gradient descent is operated on raw image pixels in neural style transfer, thus each image generation is computationally expensive. Image transformation network [9], on the other hand, builds a transformation network that learns the style transformation at training time and thus applies the transformation efficiently through one pass at image generation time.

To achieve this, a transformation network composing of convolutional and transpose convolutional layers is prepended to the deep convolutional neural network, or loss network, from the neural style transfer method. That is, the initial output image y to the loss network now comes from the output of the transformation network, and the loss is backpropagated not to y but to the transformation network that generates y from an input image x.

As a result, after the transformation network is trained with a set of style images, style transfer can be applied to any given image via one pass through the network.

3.3. CycleGAN

CycleGAN [17] is an extension of GAN [5] with enforcement of cycle consistency loss. In GAN, we are trying to learn a generator $G : X \rightarrow Y$ that maps an input image x to the styled version $G(x)$, and a discriminator D that differentiates between generated samples $G(x)$ and real samples y. The two networks learn collaboratively through a mini-max game, where G tries to minimize the following objective and D tries to maximize it:

$$\begin{aligned} &\mathcal{L}_{\text{GAN}}(G, D) \\ =&\mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D(G(x)))] \end{aligned} \quad (5)$$

As in the original CycleGAN, we adopt a variation of GAN, the Least Squares Generative Adversarial Nets (LSGAN) [11], that mitigates the vanishing gradient problem in the original GAN and can be trained more stably while producing better quality results. The loss function becomes the least squares loss:

$$\begin{aligned}&\mathcal{L}_{\text{GAN-D}}(G, D)\\ =&\frac{1}{2}\mathbb{E}_{y\sim p_{\text{data}}(y)}[(D(y)-r)^2]+\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}[(D(G(x))-f)^2]\\ &\mathcal{L}_{\text{GAN-G}}(G, D)\\ =&\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}[(D(G(x))-r)^2]\end{aligned} \tag{6}$$

where r is the label for real samples, and f is for fake ones.

However, with this objective alone, there is no guarantee that x and y can be paired up, which leads to the mode collapse problem where any input image is mapped to the exact same output image that the discriminator scores high [17]. To enforce that the output y is related to x, an additional cycle consistency loss is introduced in CycleGAN, in which there is another mapping $F : Y \to X$ that maps an output image y back to its original image x, and we want $F(G(x)) \approx x$ and $G(F(y)) \approx y$. The cycle consistency loss is defined as:

$$\begin{aligned}&\mathcal{L}_{\text{cyc}}(G, F)\\ =&\mathbb{E}_{x\sim p_{\text{data}}(x)}\|F(G(x))-x\|_1+\mathbb{E}_{y\sim p_{\text{data}}(y)}\|G(F(y))-y\|_1\end{aligned} \tag{7}$$

The full objective is:

$$\begin{aligned}&\mathcal{L}(G, F, D_X, D_Y)\\ =&\mathcal{L}_{\text{GAN-G}}(G, D_Y)+\mathcal{L}_{\text{GAN-D}}(G, D_Y)\\ +&\mathcal{L}_{\text{GAN-G}}(F, D_X)+\mathcal{L}_{\text{GAN-D}}(F, D_X)+\lambda\mathcal{L}_{\text{cyc}}(G, F)\end{aligned} \tag{8}$$

where λ is the relative weight of cycle consistency loss.

3.3.1 Additional Techniques

Some additional techniques are explored to seek better performance, which are listed below:

Label Smoothing and Label Flipping Replacing 0 and 1 labels with smoothed values, e.g. 0 to 0.3 for 0, and 0.7 to 1.2 for 1, tends to make the model more robust to adversarial examples [13]. Flipping the real label to 0 and fake label to 1 is also a practically useful technique that helps with the gradient flow in early training.

Identity Loss In addition to the above losses, an identity loss [15] defined as follows is also enforced so that the generator generates outputs as close to the inputs if the inputs are from the target domain:

$$\begin{aligned}&\mathcal{L}_{\text{identity}}(G, F)\\ =&\mathbb{E}_{x\sim p_{\text{data}}(x)}\|F(x)-x\|_1+\mathbb{E}_{y\sim p_{\text{data}}(y)}\|G(y)-y\|_1\end{aligned} \tag{9}$$

The loss encourages the color composition to be more consistent between the input and output [17], which prevents the output fonts to take on random colors.

Spectral Normalization Spectral normalization [12] is a weight normalization technique proposed to stabilize the training of the discriminator. Specifically, the weights in each layer W is divided by the spectral norm $\sigma(W)$, the singular value of W, that is estimated through power iteration method [4].

Single Channel Input For our experiment, all images are in grayscale. The original three-channel images are thus reduced to single-channel images in our training, which speeds up convergence as the search domain shrinks. Identity loss may have minimal effects in this case. If color is part of the font style, this technique should not be applied.

Image Cropping An interesting challenge that arises in cross-lingual font style transfer, as opposed to font style transfer of the same language, is the huge content difference in the characters. For example, English characters generally have less strokes than Chinese characters. If the entire fake or real images are fed into the discriminator, it may simply learn to classify the images by the number of dark pixels in the input.

Image cropping is introduced to mitigate the problem. Instead of feeding the whole image, the image is cropped at center for English fonts, and randomly cropped for Chinese fonts as a smaller patch of the original image. This way, the discriminator can focus on capturing only the local textures instead of the global content information. Whether random cropping or center cropping is used is tailored for the specific languages at use.

Note that this technique looks similar to PatchGAN [7], where discriminator outputs multiple predictions for smaller patches of the image, but this also encodes global information. In our image cropping method, only a single patch is used.

Content Loss A serious issue found through the experiments is the generated images tend to "white out" soon after a few iterations, but they can still be recovered to the

original input images. This implies that while the generator output does encode the desired content information, the correct grayscale information is less stressed throughout the training and is hard to pick up.

Inspired by neural style transfer, an additional content loss defined below is added to the model:

$$\begin{aligned}&\mathcal{L}_{\text{content}}(G, F)\\ =&\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}(x)}[(G_{\text{content}}(x) - F_{\text{content}}(G(x)))^2]\end{aligned} \tag{10}$$

where G_{content} and F_{content} are the first convolutional blocks in the respective generator that output the encoded low-level contents. The loss forces the generated image $G(x)$ to be close in content to the original image x while allowing some style changes.

Loss Weights Weights are introduced for the various losses, and the combined objective is:

$$\begin{aligned}&\mathcal{L}(G, F, D_X, D_Y)\\ =&\mathcal{L}_{\text{GAN-G}}(G, D_Y) + \mathcal{L}_{\text{GAN-D}}(G, D_Y)\\ +&\mathcal{L}_{\text{GAN-G}}(F, D_X) + \mathcal{L}_{\text{GAN-D}}(F, D_X)\\ +&\lambda\mathcal{L}_{\text{cyc}}(G, F) + \alpha\mathcal{L}_{\text{identity}}(G, F) + \beta\mathcal{L}_{\text{content}}(G, F)\end{aligned} \tag{11}$$

where λ, α, β are the respective weights for the cycle loss, identity loss, and content loss.

3.3.2 GAN Architecture

The generator G is a ResNet [6], and the discriminator D is a simple convolutional network. The architecture and layer specifications of the respective model are detailed in Table 1 and Table 2 respectively.

Module	Specifications
Encoder	ReflectionPad, pad 2
	7x7 Conv, 64 filters, stride 1, pad 0
	InstanceNorm + ReLU
	3x3 Conv, 128 filters, stride 2, pad 1
	InstanceNorm + ReLU
	3x3 Conv, 256 filters, stride 2, pad 1
	InstanceNorm + ReLU
Transfer	*(7 layers of the following ResNet)*
	ReflectionPad, pad 1
	3x3 Conv, 256 filters, stride 1, pad 0
	InstanceNorm + ReLU
	ReflectionPad, pad 1
	3x3 Conv, 256 filters, stride 1, pad 0
	InstanceNorm
Decoder	3x3 ConvT, 128 filters, stride 2, pad 1, out pad 1
	InstanceNorm + ReLU
	3x3 ConvT, 64 filters, stride 2, pad 1, out pad 1
	InstanceNorm + ReLU
Output	ReflectionPad, pad 3
	7x7 Conv, 1 filter, stride 1, pad 0
	Tanh

Table 1. Generator architecture specification. ConvT is the transposed convolutional layer.

Module	Specifications
Encoder	4x4 Conv, 64 filters, stride 2, pad 1
	LeakyReLU, negative slope 0.2
	4x4 Conv, 128 filters, stride 2, pad 1
	Norm + LeakyReLU, negative slope 0.2
	4x4 Conv, 256 filters, stride 2, pad 1
	Norm + LeakyReLU, negative slope 0.2
	4x4 Conv, 512 filters, stride 2, pad 1
	Norm + LeakyReLU, negative slope 0.2
Classification	4x4 Conv, 1 filter, stride 1, pad 1
	HxW AvgPool + Flatten

Table 2. Discriminator architecture specification. Norm can be instance norm or spectral norm. H and W are the height and width of the output of the last convolutional layer.

4. Dataset

The font image dataset is created by exporting the existing fonts in our own computer with FontForge python library[2]. Each character is exported, padded, and resized into 128×128 square image.

In this project, "PingFang" is used as the Chinese font and "Chalkduster" as the English font style. Some data samples are shown in Table 3. A total of 512 English fonts and 48900 Chinese fonts are exported as 128×128 3 RGB images, including symbols and special characters.

Data Splits The training set for CycleGAN contains all English fonts as style and 48644 Chinese fonts as content, and the test set contains 256 Chinese fonts. The image transformation network is trained on the same training set but with 1 English font style, and evaluated on the same test set. There is no training set for neural style transfer, and it is evaluated on the test set with 1 English font style.

The test set is chosen to be small as the generation time for neural style transfer is long.

Data Augmentation Five hundred style fonts are not enough of a variation for CycleGAN, hence the style dataset is augmented with random translation, rotation, and scaling on the fly when the style images are loaded. All pixel values are normalized into the range $[0, 1]$.

[2] https://github.com/fontforge/fontforge

Font	1	2	3	4	5
PingFang (Chinese)	但	利	唐	字	未
Chalkduster (English)	a	B	C	D	E

Table 3. Data samples.

5. Experiments

The qualitative and quantitative evaluation methods, experimental settings for each of the three methods, and experimental results are discussed in this section.

5.1. Evaluation Methods

The qualitative evaluation will be based on the blurriness, content preservation, and the overall style performance of the generated fonts.

The quantitative evaluation is based on content accuracy measured by Optical Character Recognition (OCR) correctness, and style discrepancy measured by style loss defined in the Neural Style Transfer algorithm [3]. Human evaluation is also conducted by the two team members as additional metrics to evaluate the content and style, and the range of the scores is 0 to 5.

For OCR methods, Tesseract OCR Engine[3] and its python wrapper[4] are used for Chinese character recognition. However, the OCR engine can only recognize commonly used characters, and its prediction hugely depends on how our inputs align with the predefined format parameters. Therefore, the accuracy measured will be based on whether a character can be identified instead of whether the predicted character is correct.

5.2. Experimental Settings

5.2.1 Neural Style Transfer

The pretrained VGG-19 network [14] is used as the initial loss network. The output of the forth convolutional layer is used as the feature representation for computing content loss, and the outputs of the first to the fifth convolutional layers are used for style loss. The loss weights α and β are set to 1 and 1000000 respectively. The character "E" as shown in Table 3 is used as the style image. The image is optimized with Limited Memory BFGS (L-BFGS) optimizer. The training lasts for 100 epochs and the transferred image with the least summation of content and style loss is saved.

[3] https://github.com/tesseract-ocr

[4] https://github.com/madmaze/pytesseract

5.2.2 Image Transformation Network

The settings for the loss network is similar to those in Neural Style Transfer, and the loss weights α and β are set to 1 and 10000. The character "E" is used as the style image. The network is optimized with Adam optimizer with a learning rate of $1e-4$. The model is trained for 4 epochs with a batch size of 64.

The architecture of the transformation network is similar to the generator in CycleGAN with convolutional encoder, transfer channels, and transposed convolutional decoder. We skip the detailed layer specifications here to save space, but it can be found in the project source code[5].

5.2.3 CycleGAN

Experiments are conducted on three models with different settings. The first one (`no_spec_norm`) has all additional techniques applied except spectral norm, and the second one (`spec_norm`) incorporates all techniques. For these two models, $\lambda = 10$, $\alpha = 5$, and $\beta = 1$. The third model (`high_beta`) also has the same settings but with a higher content loss weight $\beta = 5$.

They are all trained for 5 epochs with batch size 16, initial learning rate 0.0001 with decay starting at the second epoch, and the discriminator is trained 2 times more than the generator in each iteration. The optimizer is Adam [10] with betas being $(0.5, 0.999)$.

5.3. Results and Discussion

The five models are trained and evaluated on the test set. Qualitative results on six test samples are displayed and compared in Table 4. The quantitative results with OCR accuracy, style loss, and human evaluation scores are presented in Table 5.

5.3.1 Three Methods Comparison

It is seen from Table 4 that one of our baseline models, Neural Style Transfer, can only be applied on characters with simple structures. While from the qualitative perspective, the font style does get transferred to the test set, the content is not preserved. A deeper look into the output image shows

[5] https://github.com/pyliaorachel/cross-lingual-font-style-transfer/blob/master/project/src/style_transfer/itn/net.py

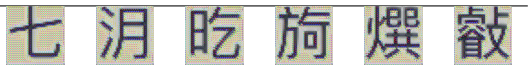
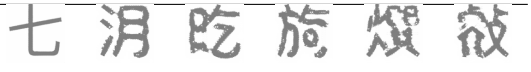
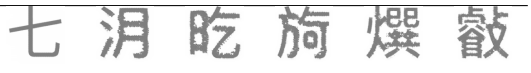
Model	1	2	3	4	5	6
Original Content Image						
Neural Style Transfer						
Image Transformation Network						
CycleGAN (`no_spec_norm`)						
CycleGAN (`spec_norm`)						
CycleGAN (`high_beta`)						

Table 4. Qualitative results on six test samples for the five models.

	OCR Accuracy	Style Loss	Human Content Score	Human Style Score
Neural Style Transfer	15.09%	2.5043	0.6	2.3
Image Transformation Network	4.40%	30.1335	5.0	0
CycleGAN (`no_spec_norm`)	13.21%	2.0773	4.6	4.8
CycleGAN (`spec_norm`)	20.13%	2.4889	4.4	4.8
CycleGAN (`high_beta`)	28.93%	2.3138	5.0	4.6

Table 5. Quantitative evaluation results on the test set for the five models.

	`no_spec_norm`			`spec_norm`			`high_beta`		
Epoch 1									
Epoch 2									
Epoch 3									
Epoch 4									
Epoch 5									

Table 6. The qualitative results over epochs for the three models of CycleGAN.

the general outline of the original character, but the overall character image is blurry and could hardly be correctly recognized. For Image Transformation Network, the second baseline model, the result is the opposite of Neural Style Transfer. Most content is preserved, but the style seems not being transferred to the test set. Compared to the previous two, CycleGAN performs the best with the content easy to recognize and the Chalkduster font style clearly observable.

For the quantitative results shown in Table 5, since the OCR accuracy is based on whether the OCR is able to identify a character, outputs with higher contrast are likely to result in higher accuracy since they are more recognizable. This is why the OCR score contradict with human evaluation score in Neural Style Transfer. Nevertheless, the style loss shows that the style quality is acceptable with Neural Style Transfer. CycleGAN models, as expected, have better evaluation scores on both content and style metrics. Image Transformation Network has a style loss 15 times larger than the other models, which also aligns with our qualitative observations.

Overall, CycleGAN is likely the best option for font style transfer. Outputs from Neural Style Transfer can be applied with the desired style but tend to be blurry. More explorations may be needed in the training techniques and network architectures of the Image Transformation Network to achieve better results.

5.3.2 CycleGAN Models Comparison

To better understand how the three CycleGAN model settings compare with each other, we have also plotted the in-

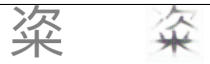
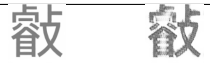
Technique	Original	Transferred
w/o image crop		
w/ image crop		
w/ PatchGAN		
w/o content loss		
w/ content loss		
w/o augmentation, $\beta = 0.5$		
w/o augmentation, $\beta = 1$		
w/ augmentation, $\beta = 1$		

Table 7. Generated images from models with and without content loss trained after one epoch. β is the content loss weight.

termediate results of the three models over the five epochs in Table 6.

There are some obvious trade-offs among the three models. With higher stress of content loss in `high_beta`, the strokes in the transferred images maintain most of their completeness without being occasionally wiped out as in the other two models, and thus the OCR accuracy and human accuracy score significantly outperform others. However, as the content consistency is so strictly enforced, model learning is conservative during training and the style improves slowly over time.

The model without spectral norm (`no_spec_norm`) has learned to transfer the style most properly while losing more or less the same amount of content information than the one with spectral normalization (`spec_norm`). From the results from the first few epochs, we can see that `no_spec_norm` might be the most aggressive in learning style transfer. Spectral normalization might have served as a stabilizing factor during training so that the model `spec_norm` is more conservative in adjusting the style, but this may not necessarily be a benefit in our style transfer task.

Overall, there is no clear winner, but with a more balanced content preservation and style transfer aggressiveness, cross-lingual font style transfer can be practicable.

5.4. CycleGAN Training Techniques Analysis

For the various training techniques introduced for CycleGAN in Section 3.3, we investigate the effects of each technique below. A comparison result is shown in Table 7.

Image Cropping and PatchGAN Without image cropping, the generator tends to wipe out the content surrounding the center of the transferred image. It is likely because the style images are composed of English characters that often only have content at the center, hence the discriminator learns to differentiate the real and fake images by the number of dark pixels in the image, and whether they are concentrated at the center. With image cropping, the white-out effect is much alleviated.

We also compare the results from using PatchGAN to verify that if predictions are based on all local patches instead of the entire image, it is still not sufficient to produce satisfying results. The result shows that while the centering effect is reduced, the discriminator still takes the number of dark pixels as a feature in its classification.

Content Loss After training a model without content loss for a few iterations, some of the transferred images tend to be completely wiped out. This shows that the discriminator might have been weighting other features more than the color information during training. After introducing content loss, the generator and discriminator have learned that adjusting the color will be heavily penalized and may soon shift towards tuning the stroke style.

Content Loss Weight and Data Augmentation There are some trade-offs tuning the content loss weight. If it is too low, it adversely affects the stroke completeness of the transferred image, as shown in the table. If it is too high, style learning tend to slow down as the generator learns not to adjust too much of the original content in each iteration. Nevertheless, we can always train it for longer to gradually learn the style, which is why `high_beta` trained for 5 epochs produces characters with complete strokes as well as the desired style.

Data augmentation is also critical to the performance. From the table, it is observed that model trained with data augmentation is able to learn the style more efficiently. It is also helpful in preventing the discriminator from memorizing the few style strokes from the center image crops, providing generalization for model training.

6. Conclusion and Future Work

In this project, three style transfer methods are implemented to achieve cross-lingual font style transfer from English font style to Chinese characters, namely Neural Style Transfer, Image Transformation Network, and CycleGAN. To improve the training efficiency and output performance, CycleGAN is augmented with several techniques.

An experiment on a dataset containing over around 50 thousand Chinese content characters "PingFang" and 500 English style fonts "Chalkduster" is conducted on these

three methods. Neural Style Transfer is able to produce images with the style pattern, but the content is much distorted. Image Transformation Network only learns to produce images similar to the original content image and fails to transfer the style. CycleGAN performs the best among the three, producing clearly recognizable characters with observable styling.

To understand the better settings of CycleGAN, we have also experimented with CycleGAN with spectral normalization and CycleGAN with a larger content loss weight. Despite learning more conservatively than the original setting, increasing the content loss weight has seen high content preservation and acceptable font style transfer. A better model is likely to be found if we train it for even longer.

Another contribution made in the project are the techniques devised for the improvement of CycleGAN, among which the improvement from image cropping and content loss are the most salient. Image cropping helps to prevent the discriminator from learning the content difference between characters of different languages, which is a challenge specific under the cross-lingual setting, by only feeding a random crop of image into the discriminator for training. Content loss forces the low-level encoded content between the original image and the transferred image to be close to each other. The two together prevents the transferred image content from being wiped out. Experiments have proved the impact of these two techniques to be critical on the output performance.

In the future, we would like to further tune the settings of our model to achieve a best model that both preserves all the content while having the style maximally transferred. We would also like to experiment with a different font style dataset or different languages to explore other kinds of challenges intrinsic to the cross-lingual setting that are not noticed.

7. Contributions

Peiyu Liao is responsible for implementing all three models, training CycleGAN and developing all the training techniques ; Fang-Yu Lin is responsible for data collection, preprocessing, evaluation, and training and fine-tuning the two baseline models.

References

[1] B. Chang, Q. Zhang, S. Pan, and L. Meng. Generating handwritten chinese characters using cyclegan. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 199–207. IEEE, 2018.

[2] S. Desai. Neural artistic style transfer: A comprehensive look, 2017.

[3] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[4] G. H. Golub and H. A. Van der Vorst. Eigenvalue computation in the 20th century. In *Numerical analysis: historical developments in the 20th century*, pages 209–239. Elsevier, 2001.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[8] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song. Neural style transfer: A review. *arXiv preprint arXiv:1705.04058*, 2017.

[9] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[12] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[13] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[16] Y. Tian. zi2zi: Master chinese calligraphy with conditional adversarial networks.(2017). *Retrieved Jun*, 3:2017, 2017.

[17] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.