Arbitrary Chinese Font Generation from a Single Reference

Zhichen Lai
College of Computer Science
Sichuan University
Chengdu, China
ryanlai.cs@gmail.com

Chenwei Tang
College of Computer Science
Sichuan University
Chengdu, China
tangchenwei826@gmail.com

Jiancheng Lv*
College of Computer Science
Sichuan University
Chengdu, China
lvjiancheng@scu.edu.cn

***Abstract*—Generating a new Chinese font from a multitude of references is an easy task, while it is quite difficult to generate it from a few references. In this paper, we investigate the problem of arbitrary Chinese font generation from a single reference and propose a deep learning based model, named One-reference Chinese Font Generation Network (OCFGNet), to automatically generate any arbitrary Chinese font from a single reference. Based on the disentangled representation learning, we separate the representations of stylized Chinese characters into style and content representations. Then we design a neural network consisting of the style encoder, the content encoder and the joint decoder for the proposed model. The style encoder extracts the style features of style references and maps them onto a continuous Variational Auto-Encoder (VAE) latent variable space while the content encoder extracts the content features of content references and maps them to the content representations. Finally, the joint decoder concatenates both representations in layer-wise to generate the character which has the style of style reference and the content of content reference. In addition, based on Generative Adversarial Network (GAN) structure, we adopt a patch-level discriminator to distinguish whether the received character is real or fake. Besides the adversarial loss, we not only adopt L1-regularized per-pix loss, but also combine a novel loss term Structural SIMilarity (SSIM) together to further drive our model to generate clear and satisfactory results. The experimental results demonstrate that the proposed model can not only extract style and content features well, but also have good performance in the generation of Chinese fonts from a single reference.**

***Index Terms*—Chinese font generation, variational auto-encoder, generative adversarial network**

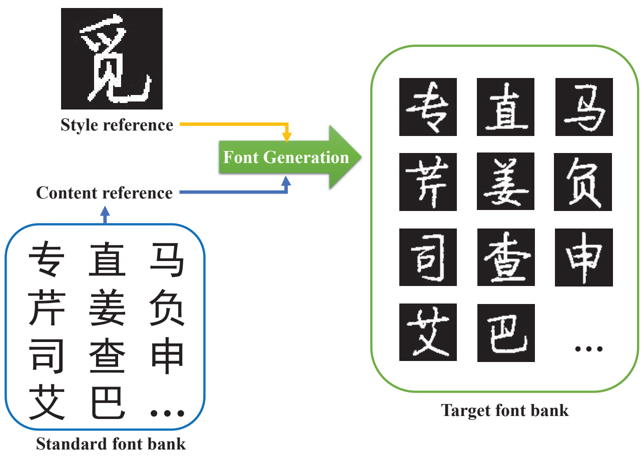

Fig. 1. Schematic diagram of the arbitrary Chinese font generation from a single reference. With a single style reference and the standard font bank for providing content references, the model generates the target font bank.

I. Introduction

Chinese fonts have been extensively used in various aspects of art and design. However, designing a new Chinese font is a vapid and time-consuming task. Different from English or Latin which consists of a small alphabet, Chinese consists of more than 90,000 characters, among which about 3,700 characters are frequently used. Furthermore, the complicated structures and diverse shapes of Chinese characters increase its difficulty. Building a new Chinese font bank involves considerable manpower, calligraphers have to design a large amount of Chinese characters manually and make sure all the characters share the same style. In order to simplify the procedures of building a Chinese font bank, we investigate the problem of Chinese font generation from a single reference.

Recently, Chinese font generation gains widespread interest in the research community. The most typical method is based on stroke extraction [1], which divides the generation into two stages: stroke extraction and stroke recombination. However, due to the complexity and variety of Chinese characters, the algorithms of stroke extraction can not yield reliable results, which leads to the failure of this kind of methods.

Inspired by the recent progress of neural style transfer [2], [4], [6], [16], [22], many attempts have been made to take Chinese font generation as an image-to-image translation problem [3], [17], [26]. However, unlike other neural style transfer tasks, Chinese font generation is not a fault-tolerant task, since any misplacement of strokes may result in the failure of generation [3]. Some approaches adopt Convolutional Neural Networks (CNNs) with a bottle-neck structure based on GAN [3], [11], [14], [19], [20], which show better performance than previous methods.

Although some acceptable results have been generated by some models, most existing methods require a large amount of initial reference set to train their models, which is about 3,000 characters for training supervision [3], [14], [19], [20]. Some researchers [11], [17], [26] attempt to generate font banks with a small initial reference set. However, they depend on extra labels or pretrained models, which still need considerable manpower. In addition, most existing methods aim to explicitly model the translation between the standard font and the target font [19], thus these models are not generalizable to new fonts.

In this paper, we propose a deep learning based model OCFGNet to enlarge a Chinese font bank from one single reference. We design a dual-encoder generator to extract style and content features of style and content references and synthesize the target character with these features. It is worth mentioning that the style encoder extracts style features and maps the style reference onto a continuous VAE latent variable space and the content encoder extracts content features and maps them to the content representations. Then, the joint decoder connects the two representations in layer-wise to generate the target character. In addition, to generate more realistic results, we adopt a patch-level discriminator. In general, the training of OCFGNet includes these four steps: 1) inputting style and content references into style and content encoders to extract their representations, respectively, 2) connecting the two representations in layer-wise by the joint decoder to synthesize the target character, 3) cloning the previous two steps to contribute to the cycle consistency loss, 4) inputting the generated character and the ground-truth (GT) into the patch-level discriminator to distinguish whether the received character is real or fake.

The main contributions of our work can be summarized as follows:

- We propose a Chinese font generation model OCFGNet, which can generate any unseen Chinese font given one reference.
- We collect a Chinese font dataset based on different style categories. A total of 500 fonts were collected.
- To synthesize more realistic characters, we introduce the Structural SIMilarity (SSIM) as a loss term to optimize this model, which are effective in mitigating the blur of font generation.

II. RELATED WORK

A. Image Generation

Image generation is one of the most fundamental problems in computer vision, which has been widely researched for decades. By the great capability of extracting features, CNNs [13] have achieved great success in the improvement of image generation. Moreover, recent advances on representation learning with deep neural networks nourish a battery of deep generative models that enjoy joint generative modeling and representation learning through Bayesian inference or adversarial training.

Yan et al. proposed a conditional image generation model Attribute2Image, which models an image as a composite of foreground and background and extends the VAE with disentangled latent variables [25]. The proposal of Deep Recurrent Attentive Writer (DRAW) introduced the attention mechanism to VAE [8]. Since GAN was proposed by Goodfellow et al. [7], researchers have studied it vigorously. Pix2Pix, a conditional GAN, was proposed for image translation problems [9], which has been extended for generating high-resolution images [23]. The trainable loss function of discriminator makes GAN automatically adapt to the differences between the generated images and the real images. Afterwards, most image generation approaches are based on variant GANs.

B. Chinese Font Generation

Many efforts have been put into the research of Chinese font generation. In [1], Cao et al. proposed to extract the strokes first, then cluster the strokes into different groups, finally generate the new font characters by stroke replacement. Rewrite adopted a CNN-based structure to transfer one Chinese font to another [19]. Lyu et al. proposed the Auto-Encoder Guided GAN Network (AEGN), which is able to synthesize target font characters from standard font [14]. By introducing conditional GAN and assigning each style an one-hot categorical label, Zi-to-zi model gains much improvement [20]. Zhang et al. explored learning with less paired character samples, which still requires tens of the style references [26]. Jiang et al. proposed an one-shot Chinese character generation model, but the experimental results show that this model only learns the thickness between different fonts [10].

III. MODEL

Figure 2 shows the overall architectures of OCFGNet. Here, we introduce some notations and the problem definition. Generally, X denotes the Chinese font dataset consisting of J different characters with I different fonts. Specifically, X_j^i denotes a specific character in X. The superscript $i \in [0,1,2,...,I]$ represents the i-th font in the experimental font pool, and the subscript $j \in [1,2,...,J]$ denotes the j-th character in the experimental alphabet. Before training, we designate the specific font X^0 as the standard font for providing content references, which is then trained on the content auto-encoder (E_S and E_C) to learn the content representations of the standard font characters. During the training phase, in every training step we randomly pick the content character X_j^0 from the standard font X^0, named as the content reference C, and we also randomly designate a specific character X_k^i from the style reference font X^i as the style reference S ($j \neq k$ and $i \neq 0$). Thus X_j^i denotes the target character T which has the style of S and the content of C. Let E_S and E_C denote the style encoder and the content encoder, respectively, and D_T denotes the joint decoder of OCFGNet. The formulation of the generation is:

$$T = X_j^i = D_T(E_S(S), E_C(C)) = D_T(E_S(X_k^i), E_C(X_j^0)). \quad (1)$$

A. Features Extraction Networks

As is clear from Figure 2, we adopt two separate encoders E_S and E_C to extract style and content features, respectively. The style encoder E_S extracts the style features of the style reference S and outputs the standard deviation σ and the average μ of the style representation z_S. By the reparameterization trick, we obtain the style representation z_S. Meanwhile, the

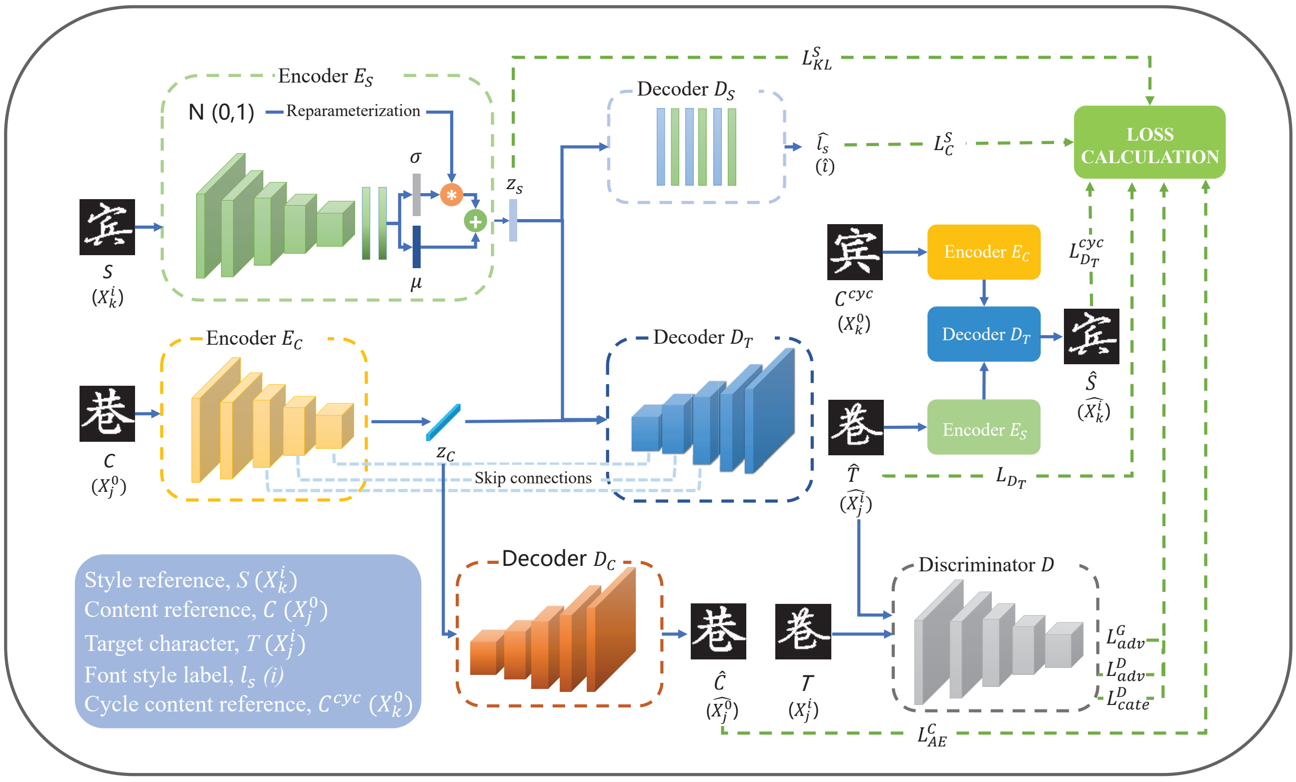

Fig. 2. The framework of the proposed OCFGNet.

content encoder E_C extracts the content features of the content reference C and directly encodes it as z_C by down-sampling.

Besides, we adopt an auxiliary style decoder D_S to classify z_S into its font style label l_s, which is able to aggregate z_S with the same style. We also implement an auxiliary content decoder D_C to reconstruct the content reference C for learning the content information of the standard font and determine the content representations z_C of the standard font characters. It is noteworthy that the training of the auto-encoder module (E_C and D_C) is completed and the parameters of the auto-encoder module are fixed before the training of other modules.

B. *Features Synthesis Network*

After extracting both the style representation z_S and the content representation z_C, we reshape the style representation z_S into the same shape of the content representation z_C. Then, the joint decoder D_T concatenates both representations in layer-wise to generate the generated target character $\hat{T}$. It is worth noting that the structure of U-Net with skip connections is introduced for shuttling the information directly from the content encoder E_C to the joint decoder D_T [15]. Each skip connection simply concatenates all channels at layer i in E_C with those at layer n-i in D_T.

C. *Patch-level Discriminator Network*

The patch-level discriminator D concatenates the style and content reference characters with the real target character T or the generated character $\hat{T}$ as inputs, and tries to distinguish them. D outputs the probabilities that inputs are real or fake in the corresponding patch. In addition, to improve the discernibility of the discriminator D, it is designated by adding a fully-connected layer to implement the auxiliary classifier in the discriminator D, which could classify the character into its style label i-th.

D. *Optimization Strategies and Losses*

For the style encoder E_S, to aggregate the style representation z_S from the same style, we adopt the auxiliary style decoder D_S to classify z_S into its font style label i-th. Thus, we introduce the style representation categorical loss $\mathcal{L}_{cate}^S$:

$$\mathcal{L}_{cate}^S = \log D_S(i|z_S) = \log D_S(i|E_S(X_k^i)). \qquad (2)$$

In addition to minimizing the categorical loss $\mathcal{L}_{cate}^S$ of the style encoder, the VAE regularizes the encoder by imposing a prior over the latent distribution $p(z)$, where $z \sim \mathcal{N}(0,1)$. Thus, we introduce the KL-regularized loss $\mathcal{L}_{KL}^S$:

$$\mathcal{L}_{KL}^{S} = KL(q(z_S|S)\,\|\,N(0,1)) = \frac{1}{2}(1+\log(\sigma^2)-\mu^2-\sigma^2), \quad (3)$$

where σ and μ denote the standard deviation and the average of z_S, respectively.

For the content encoder E_C, to extract the content features of the content references, we introduce the auxiliary content decoder D_C, which constitutes an auto-encoder module with E_C. Unlike traditional auto-encoders, we adopt both L1-regularized per-pix loss and Structural SIMilarity (SSIM) loss to reconstruct content references, since L1-regularized per-pix loss and SSIM loss can evaluate the similarity between two images at the pixel and structure levels, respectively. Thus, the loss term of the content auto-encoder $\mathcal{L}_{AE}^{C}$ is:

$$\begin{aligned}\mathcal{L}_{AE}^{C} &= \left\|C-\hat{C}\right\|_1 + \alpha SSIM(C,\hat{C}) \\ &= \left\|X_j^0-\hat{X}_j^0\right\|_1 + \alpha SSIM(X_j^0,\hat{X}_j^0),\end{aligned} \quad (4)$$

where α denotes the weight of SSIM loss, and SSIM($\cdot$) denotes the SSIM function which is referred in Eq. 11 .

For the joint decoder D_T, we adopt the same reconstruction loss term $\mathcal{L}_{D_T}$ as the content decoder D_C:

$$\begin{aligned}\mathcal{L}_{D_T} &= \left\|T-\hat{T}\right\|_1 + \beta SSIM(T,\hat{T}) \\ &= \left\|X_j^i-\hat{X}_j^i\right\|_1 + \beta SSIM(X_j^i,\hat{X}_j^i),\end{aligned} \quad (5)$$

where β denotes the weight of SSIM loss, and SSIM($\cdot$) denotes the SSIM function.

It is worth nothing that we introduce a patch-level discriminator D. During the training phase, the style encoder E_S and the joint decoder D_T are optimized by minimizing $\mathcal{L}_{adv}^{G}$, while the discriminator D is optimized by minimizing $\mathcal{L}_{adv}^{D}$:

$$\begin{aligned}\mathcal{L}_{adv}^{G} &= \mathbb{E}[D(S,C,D_T(E_S(S),E_C(C)))] \\ &= \mathbb{E}[D(X_k^i,X_j^0,D_T(E_S(X_k^i),E_C(X_j^0)))],\end{aligned} \quad (6)$$

$$\begin{aligned}\mathcal{L}_{adv}^{D} &= \mathbb{E}[D(S,C,T)] - \mathbb{E}[D(S,C,D_T(E_S(S),E_C(C)))] \\ &= \mathbb{E}[D(X_k^i,X_j^0,X_j^i)] - \mathbb{E}[D(X_k^i,X_j^0,D_T(E_S(X_k^i),E_C(X_j^0)))].\end{aligned} \quad (7)$$

In addition, to improve the discernibility of the discriminator D, we also adopt the auxiliary classifier D_{cate} implemented in the discriminator D to contribute to the categorical loss $\mathcal{L}_{cate}^{D}$ for the discriminator D:

$$\begin{aligned}\mathcal{L}_{cate}^{D} &= \log D_{cate}(i|T) + \log D_{cate}(i|D_T(E_S(S),E_C(C))) \\ &= \log D_{cate}(i|X_j^i) + \log D_{cate}(i|D_T(E_S(X_k^i),E_C(X_j^0))).\end{aligned} \quad (8)$$

Finally, to maintain the cycle consistency, we add a cycle consistency loss to guarantee that the synthesized character T can be used as the style reference to translate the standard font character to the character in the corresponding style. Formally, we define the cycle consistency loss $\mathcal{L}_{D_T}^{cyc}$ as:

$$\begin{aligned}\mathcal{L}_{D_T}^{cyc} &= \left\|S-\hat{S}\right\|_1 + \gamma SSIM(S,\hat{S}) \\ &= \left\|X_k^i-\hat{X}_k^i\right\|_1 + \gamma SSIM(X_K^i,\hat{X}_k^i),\end{aligned} \quad (9)$$

where γ denotes the weight of SSIM loss, and SSIM($\cdot$) denotes the SSIM function.

Algorithm 1 shows the details of training procedure of OCFGNet.

Algorithm 1 Training procedure of OCFGNet

```
Input:
    The style reference, $S/X_k^i$;
    The content reference, $C/X_j^0$;
    The target character, $T/X_j^i$;
    The style label, $l_s/i$;
    The cycle content reference, $C^{cyc}/X_k^0$;
    The weights of losses, $\alpha$, $\beta$ and $\gamma$.
Output:
    The generated target character, $\hat{T}/\hat{X}_j^i$.
 1: $z_C \leftarrow E_C(C)$;
 2: $\hat{C} \leftarrow D_C(z_C)$;
 3: Compute $\mathcal{L}_{AE}^{C}$ by Eq. 4;
 4: Update $D_C$ and $E_C$ by descending the gradient of $\mathcal{L}_{AE}^{C}$;
 5: Repeat step 1-4 for 100 epochs;
 6: $z_S \leftarrow E_S(S)$, $z_C \leftarrow E_C(C)$;
 7: $z_C \leftarrow E_C(C)$;
 8: $\hat{l_s} \leftarrow D_S(z_S)$;
 9: $\hat{T} \leftarrow D_T(z_S, z_C)$;
10: $z_S^{cyc} \leftarrow E_S(\hat{T})$, $z_C^{cyc} \leftarrow E_C(C^{cyc})$;
11: $z_C^{cyc} \leftarrow E_C(C^{cyc})$;
12: $\hat{S} \leftarrow D_T(z_S^{cyc}, z_C^{cyc})$;
13: Compute$\mathcal{L}_{cate}^{S}$, $\mathcal{L}_{KL}^{S}$, $\mathcal{L}_{D_T}$, $\mathcal{L}_{D_T}^{cyc}$, $\mathcal{L}_{adv}^{G}$, $\mathcal{L}_{adv}^{D}$ and $\mathcal{L}_{cate}^{D}$
    by Eq. 2, Eq. 3, Eq. 5, Eq. 9, Eq. 6, Eq. 7 and Eq . 8;
14: Update $E_S$ and $D_S$ by descending gradients of $\mathcal{L}_{cate}^{S}$ and
    $\mathcal{L}_{KL}^{S}$;
15: Update $E_S$ and $D_T$ by descending gradients of $\mathcal{L}_{D_T}$, $\mathcal{L}_{D_T}^{cyc}$
    and $\mathcal{L}_{adv}^{G}$;
16: Update $D$ by descending the gradient of $\mathcal{L}_{adv}^{D}$ and $\mathcal{L}_{cate}^{D}$;
17: Randomly choose new style and content references;
18: Repeat step 6-17 for 600 epochs.
```

E. Implementation Details

For the style encoder E_S, it contains five convolution layers followed by two fully-connected layers(The convolution layers have 8, 16, 32, 64 and 128 channels with filter size of 5×5, 3×3, 3×3, 3×3 and 3×3, and stride 2; the fully-connected layers both have 256 neurons). The output of the style encoder E_S is the the standard deviation σ and the average μ of the style representation z_S. With the reparameterization trick, we obtain the style representation z_S which is one-dimensional. For the style decoder D_S, it contains three fully-connected layers (The fully-connected layers have 256, 512 and 400 neurons).

For the content encoder E_C, it contains five convolution layers (The convolution layers have 8, 16, 32, 64 and 128 channels with filter size of 5×5, 3×3, 3×3, 3×3 and 3×3, and stride 2). The output of the content encoder E_C is the representation of the content reference z_C. The output size is set as $2 \times 2 \times 128$. For the content decoder D_C, it contains five de-convolutional layers (The de-convolutional layers have 64, 32, 16, 8 and 1 channel(s) with filter size of 3×3, 3×3, 3×3, 3×3 and 5×5, and stride 2).

For the joint decoder D_T, we first resize the style representation z_S as $2 \times 2 \times 64$. Then, z_S and z_C are concatenated at the first layer of the joint decoder D_T followed by five convolution layers(The de-convolutional layers have 64, 32, 16, 8 and 1 channel(s) with filter size of 3×3, 3×3, 3×3, 3×3 and 5×5, and stride 2).

For the patch-level discriminator D, it contains five convolution layers (with 64, 128, 256, 512, 1 channel(s) with filter size of 4×4 and stride 2, 2, 2, 1). In addition, we design one fully-connected layer with 500 neurons following the last second convolution layer of D, which constitutes the auxiliary classifier D_{cate} to contribute to the categorical loss of the discriminator D.

For the training phase of the generator G of the OCFGNet, there are two steps. First, we pretrain the auto-encoder of the content reference which consists of the encoder E_C and the decoder D_C. After that, we determine and the content representations of all standard font characters. Then, we train the style encoder E_S, the style decoder D_S and the joint decoder D_T together. Specially, since we have already determined the content representations z_C by pretraining the encoder E_C and the decoder D_C, the encoder E_C and the decoder D_C are fixed in the following training phase.

For the training phase of the discriminator D of the OCFGNet, the style reference S and the content reference C are concatenated with the generated target character $\hat{T}$ and the ground-truth target character T by channel axis to form the negative sample and positive sample, respectively.

For the whole training phase, we pretrain the auto-encoder of content references (the encoder E_C and the decoder D_C) for 100 epochs. Then, we train the OCFGNet generator (the style encoder E_S, the style decoder D_S and the joint decoder D_T) and the discriminator D alternately for another 600 epochs. All networks are trained with ADAM optimizers [12] with a batch size of 64 and a learning rate of 0.0005. For the setting of the hyper parameters, we set $\alpha = \beta = \gamma = 0.01$.

For the testing phase, we input the style reference S and the content reference C into the encoder E_S and E_C to obtain z_S and z_C, respectively. Then, the joint decoder D_T generates the generated target character $\hat{T}$ from the latent variable z_S and z_C.

IV. Experiments

To demonstrate the effectiveness and superiority of our model, we conduct extensive experiments. First, we present the experimental settings, mainly including the dataset and evaluation metrics. Then, we compare the proposed model with some existing state-of-the-art methods. Finally, we analyze the qualitative results of our model.

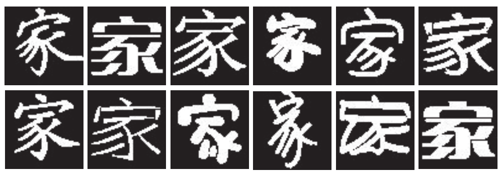
Fig. 3. Some character samples from our collected dataset.

A. Experimental Settings

Dataset Since there is no benchmark dataset available for Chinese font generation, we collect 500 Chinese fonts for the experiments which are collected from several font libraries. Table I shows the organization of our collected dataset. All fonts in the dataset contain 3,755 level-1 simplified Chinese characters. For the experimental settings, as illustrated in Figure 3, each character image is made into a 64×64 gray value image separately. To be specific, the backgrounds are set as black pixels and the strokes are set as white pixels. It is worth noting that characters with less than 4 strokes were excluded, since those may not provide enough style information. 1,000 characters from level-1 simplified Chinese characters are collected randomly for both the training set and testing set. We split the dataset into training and testing sets by the proportion of 8 : 2.

TABLE I
THE ORGANIZATION OF OUR COLLECTED DATASET

Category	Number of fonts
Regular script	100
Running script	100
Official script	100
Cursive script	50
Calligraphy script	50
Unspecified script	100
Total	**500**

Evaluation Metrics To measure the similarity between generated images and Ground Truth images (GT), we adopt Mean Square Error (MSE) [5] and Structural SIMilarity (SSIM) [24] as the evaluation metrics.

MSE measures the average squared difference of pixel values between two images. It is always non-negative, and values closer to zero are better. MSE between the generated image X and the GT image Y of the same size is defined as:

$$MSE(X, Y) = \frac{1}{N} \sum_{i,j} (X_{i,j} - Y_{i,j})^2, \tag{10}$$

where N denotes the pixel number of the image.

SSIM evaluates the similarity of the structural information between two images without being influenced by the light or small variance. Its value ranges from 0 to 1. The higher the SSIM, the better the image quality. SSIM between two images X and Y is defined as:

$$SSIM(X,Y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{11}$$

where μ_x and μ_y denote the averages of pixel values, σ_x and σ_y denote the standard deviations of pixel values, σ_{xy} denotes the covariance of X and Y, c_1 and c_2 are two constants to stabilize the division with weak denominator, which are determined by the dynamic range of the images.

B. Comparison with State-of-The-Art Methods

We compare the performance of our model with several state-of-the-art Chinese font generation models: Rewrite [19], PEGAN [18] and Zi2zi [20]. One thing to be particularly noted is that Rewrite, PEGAN and Zi2zi are modified for the few-shot Chinese font generation task by the fine-tuning strategy. During the fine-tuning phase, we provide Rewrite, PEGAN and Zi2zi with 50 target style references, while in our model we only provide one target style reference for the training supervision. Table II shows the average SSIM and MSE values of the results of these models. We can see that Rewrite has the lowest SSIM and MSE values among these models on the dataset. PEGAN and Zi2zi show significant improvement. Our model shows the greatest performance among these models, which indicates that our model has the state-of-the-art ability to generate realistic Chinese font characters from only one reference.

TABLE II
SSIM AND MSE VALUES OF TESTED MODELS

Model	*Reference Num*	*SSIM*	*MSE*
Rewrite [19]	50	0.39	0.48
PEGAN [18]	50	0.42	0.23
Zi2zi [20]	50	0.47	0.25
Ours	**1**	**0.62**	**0.20**

As illustrated in Figure 4, we also compare some representative samples of state-of-the-art models with our model. Rewrite can only generate blurry results, and even the stroke placements are incorrect. PEGAN shows clearer results than Rewrite and the generated characters present richer stylized details. Moreover, the thicknesses and style factors of the generated characters are quite similar to GT. However, it can be seen that the generated characters looks still unrealistic and unclear. Zi2zi presents similar results with PEGAN. Our model shows clearer results than other models. Our model can not only replace strokes correctly, but also imitate the corresponding style factors to present satisfactory and clear results.

To verify the recognizability of the generated Chinese characters, we adopt the Chinese Optical Character Recognition (OCR) algorithm [21] to recognize the generated results. Table III shows the recognition accuracies of the tested models. Our model shows great advantages in this test, which indicates that our model is capable of generating realistic and recognizable characters in the target font style.

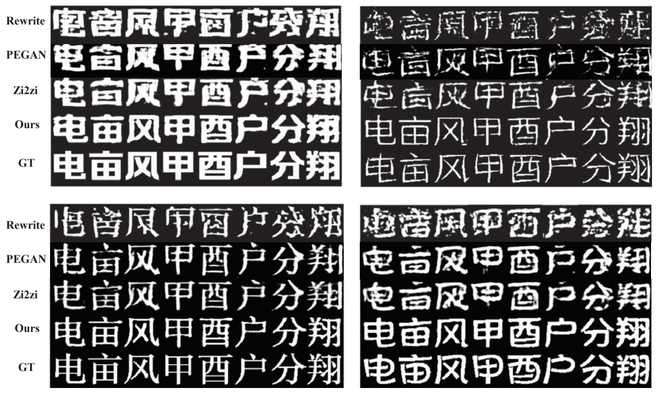

Fig. 4. Comparison with several state-of-the-art methods.

TABLE III
OCR RECOGNITION ACCURACIES OF TESTED MODELS

Model	*Recognition Accuracy*
Rewrite [19]	35.4%
PEGAN [18]	73.9%
Zi2zi [20]	75.1%
Ours	**91.2**%
Ground Truth	**96.7**%

C. Qualitative Results

Figure 5 shows several groups of examples by our model. In each group, images in the top row are GT images from the dataset, the images in the bottom row are the generated results by our model. Compared with the GT, the style of the generated results are consistent with the GT. All the results demonstrate that our model is able to generate realistic Chinese font characters with rich stylized details from only a single reference.

Additionally, we give some failure examples by our model in Figure 6. The failure examples show that some strokes are unclear and blurry. It may be caused by the strokes being too thick or too thin, which urges us to continue our research in the future.

D. Conclusion

In this work, we propose OCFGNet model to automatically generate stylized Chinese fonts with only one reference. We adopt the ideas of disentangled representation learning in our model. To generate target font characters, we present the generator of the OCFGNet consisting of style encoder, content encoder and joint decoder. In order to model richer style details and mitigate the blur, we adopt a patch discriminator, the idea of cycle consistency and the SSIM loss term. Experimental results on the dataset support that our model is able to generate more realistic results than other state-of-the-art methods with

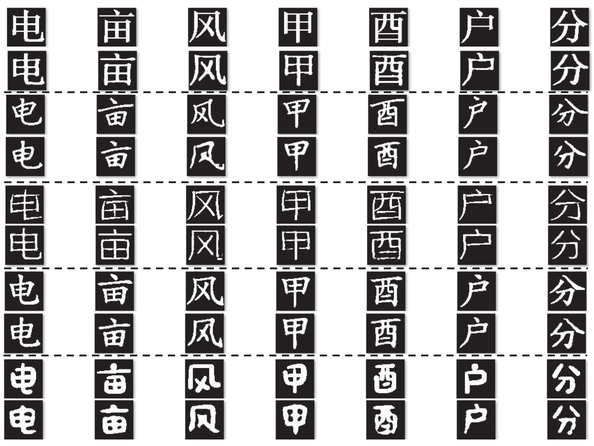
Fig. 5. Several groups of examples by our model.

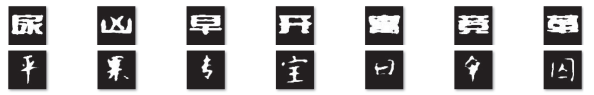
Fig. 6. Some failure examples by our model.

only one reference. In the future, we will further optimize our model and explore the adaptability and the generalization ability of our model in other scenes. The practical applications are to be developed not only restricted in the Chinese font domain, but also in other stylized image generation tasks.

ACKNOWLEDGMENT

This work is supported by the National Key R&D Program of China under contract No. 2017YFB1002201, the National Natural Science Fund for Distinguished Young Scholar (Grant No. 61625204), and the Key Program of National Science Foundation of China (Grant No. 61836006).

REFERENCES

[1] R. Cao and C. L. Tan, "A model of stroke extraction from chinese character images," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 4. IEEE, 2000, pp. 368–371.
[2] H. Chang, J. Lu, F. Yu, and A. Finkelstein, "Pairedcyclegan: Asymmetric style transfer for applying and removing makeup," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 40–48.
[3] J. Chang, Y. Gu, Y. Zhang, Y.-F. Wang, and C. M. Innovation, "Chinese handwriting imitation with hierarchical generative adversarial network." in *BMVC*, 2018, p. 290.
[4] Y. Chen, Y.-K. Lai, and Y.-J. Liu, "Cartoongan: Generative adversarial networks for photo cartoonization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9465–9474.
[5] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," *IEEE transactions on acoustics, speech, and signal processing*, vol. 33, no. 2, pp. 443–445, 1985.
[6] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2414–2423.
[7] 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*, 2014, pp. 2672–2680.
[8] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.
[9] 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*, 2017, pp. 1125–1134.
[10] H. Jiang, G. Yang, K. Huang, and R. Zhang, "W-net: One-shot arbitrary-style chinese character generation with deep neural networks," in *International Conference on Neural Information Processing*. Springer, 2018, pp. 483–493.
[11] Y. Jiang, Z. Lian, Y. Tang, and J. Xiao, "Dcfont: an end-to-end deep chinese font generation system," in *SIGGRAPH Asia 2017 Technical Briefs*. ACM, 2017, p. 22.
[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[13] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.
[14] P. Lyu, X. Bai, C. Yao, Z. Zhu, T. Huang, and W. Liu, "Auto-encoder guided gan for chinese calligraphy synthesis," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 1095–1100.
[15] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
[16] A. Sanakoyeu, D. Kotovenko, S. Lang, and B. Ommer, "A style-aware content loss for real-time hd style transfer," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 698–714.
[17] D. Sun, T. Ren, C. Li, H. Su, and J. Zhu, "Learning to write stylized chinese characters by reading a handful of examples," *arXiv preprint arXiv:1712.06424*, 2017.
[18] D. Sun, Q. Zhang, and J. Yang, "Pyramid embedded generative adversarial network for automated font generation," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 976–981.
[19] Y. Tian, "Rewrite: Neural style transfer for chinese fonts," https://github.com/kaonashi-tyc/Rewrite, 2016.
[20] ——, "Zi2zi: Master chinese calligraphy with conditional adversarial networks," https://github.com/kaonashi-tyc/zi2zi, 2017.
[21] J. Wang, "Chinese ocr implemented using crnn," https://github.com/jinxiwang/ocrTDR, 2018.
[22] J. Wang and A. Cherian, "Learning discriminative video representations using adversarial perturbations," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 685–701.
[23] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8798–8807.
[24] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
[25] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," in *European Conference on Computer Vision*. Springer, 2016, pp. 776–791.
[26] Y. Zhang, Y. Zhang, and W. Cai, "Separating style and content for generalized style transfer," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8447–8455.