Look Closer to Supervise Better: One-Shot Font Generation via Component-Based Discriminator

Yuxin Kong[1], Canjie Luo[1], Weihong Ma[1] , Qiyuan Zhu[2], Shenggao Zhu[2], Nicholas Yuan[2], Lianwen Jin*[1,3]
[1]School of Electronic and Information Engineering, South China University of Technology.
[2]Huawei Cloud AI. [3]Peng Cheng Laboratory.
{kongyxscut, canjie.luo, scutmaweihong, lianwen.jin, nicholas.jing.yuan}@gmail.com
{zhuqiyuan2, zhushenggao}@huawei.com

Abstract

Automatic font generation remains a challenging research issue due to the large amounts of characters with complicated structures. Typically, only a few samples can serve as the style/content reference (termed few-shot learning), which further increases the difficulty to preserve local style patterns or detailed glyph structures. We investigate the drawbacks of previous studies and find that a coarse-grained discriminator is insufficient for supervising a font generator. To this end, we propose a novel Component-Aware Module (CAM), which supervises the generator to decouple content and style at a more fine-grained level, i.e., the component level. Different from previous studies struggling to increase the complexity of generators, we aim to perform more effective supervision for a relatively simple generator to achieve its full potential, which is a brand new perspective for font generation. The whole framework achieves remarkable results by coupling component-level supervision with adversarial learning, hence we call it Component-Guided GAN, shortly CG-GAN. Extensive experiments show that our approach outperforms state-of-the-art one-shot font generation methods. Furthermore, it can be applied to handwritten word synthesis and scene text image editing, suggesting the generalization of our approach.

1. Introduction

To better tackle the few-shot font generation issue, we rethink the following two questions: 1) What determines people's judgment on font styles? 2) How do people learn to write a new character/glyph in the correct structure? To answer the first question intuitively, we present a text string in three different font styles in Figure 1. Since their overall architectures are similar, we naturally pay more attention to local details, including endpoint shapes, corner sharpness, stroke thickness, joined-up writing pattern, *etc.*, which appear at a more local level, *i.e.*, the components of a character. Although the components cannot present some font style properties, such as the inclination and aspect ratio, we argue that the components determine the font style to a greater extent than the whole character shape. As for the second question, one strong assumption is that when people learn a complicated glyph, they first learn the components that form the character. Intuitively, if all the components in a glyph are properly written, we can obtain the glyph correctly. Drawing inspiration from the above observations, an intuitive method for few-shot font generation is to utilize the component information that is largely correlated with the font style properties and glyph structures.

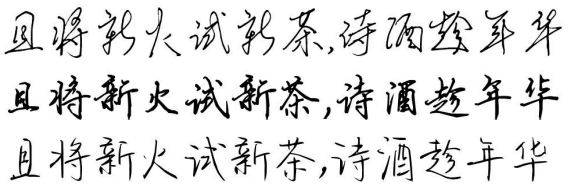


Figure 1. A same text string presented in three different font styles.

Few-shot font generation (FFG) has received considerable research interest in recent years due to its critical applications [3, 25, 26, 31, 36, 38]. An ideal FFG system can greatly reduce the burden of the time-consuming and labor-intensive font design process, particularly for those language systems with a massive number of glyphs, *e.g.*, Chinese with more than 25,000 glyphs. Another application is to create a cross-lingual font library, *e.g.*, Chinese to Korean, given the fact that Adobe and Google take years to create the Source Han Sans Font, a universal font style that supports Chinese, Korean, and Japanese simultaneously.

Recently, several attempts have been made to few-shot font generation; however, they all have certain limitations and further improvements are therefore required. For instance, [26] learns to map the source font style to a fixed

*Corresponding author.

target font style, and thus has to be retrained for another new style. "zi2zi" [38] learns multiple font styles by adding a pre-defined style category embedding, but still cannot generalize to unseen font styles. One notable approach is EMD [36], which is generalizable to unseen styles by disentangling the style and content representation, but the result is not promising due to its flaw in loss function design.

Lately, several methods utilize the idea of compositionality. Still, they have significant drawbacks. For instance, CalliGAN [31] generates glyph images conditioned on the learned embeddings of the component labels and style labels, hence cannot generalize to either unseen styles or unseen components. DM-Font [3] employs a dual-memory architecture for font generation. However, it requires a reference set containing all the components to extract the stored information, which is unacceptable for the FFG scenario. LF-Font [25] can be extended to unseen styles conditioned on the component-wise style features. However, its visual quality decreases significantly in the one-shot generation scenario. Although these component-based algorithms advance by successfully encoding the diverse local styles, they explicitly depend on the component category inputs to extract style features, and thus the capability of cross-lingual font generation is entirely beyond their reach. Meanwhile, the above methods [3, 25, 26, 31, 36, 38] have a common limitation, that is, they require large amounts of paired data for pixel-level strong supervision. Although DG-Font [32] achieves unsupervised font generation, the generated glyphs often contain characteristic artifacts. Overall, the performance of the state-of-the-arts is still unsatisfactory.

In this paper, we propose a novel component-guided generative network, namely CG-GAN, which might provide a new perspective for few-shot font generation. The proposed method is inspired by two human behavior: 1) people naturally pay more attention to component parts when distinguishing font styles, and 2) people learn a new glyph by first learning its components. Such a human learning scheme is perfectly adopted in our proposed Component-Aware Module (CAM), supervising the generator at the component level for both styles and contents. Specifically, CAM first employs an attention mechanism for component extraction, acting as a loss function to supervise whether each component is transferred properly during the generation. Then the learned attention maps, which represent the corresponding component information, are used to conduct per-component style classification and realism discrimination. Finally, with the multiple component-level discriminative outputs, CAM can feedback more fine-grained information to the generator by backpropagation, encouraging the generator to simultaneously focus on three critical aspects at the component level: style consistency, structural correctness, and image authenticity. Therefore, the quality of the generated glyph images is significantly boosted. Since CAM is only performed as the component-level supervision during training, it will not bring additional compute time in inference. In addition, paired training data are not required with our method. Once the model is trained, our generator is capable to generalize to unseen style, unseen content, even other unseen language glyphs, *i.e.*, cross-lingual font generation.

Essentially, our goal is to seek an algorithm that can effectively enhance the representational ability of the generator. The proposed CG-GAN allows to employ style-content disentanglement at a more fined grained level, *i.e.*, the component level, thus enabling to extract high-quality representations from even a single reference image. The method of utilizing component-level supervision rather than pixel-level strong supervision is a human-like method that shows effectiveness in capturing localized style patterns and preserving detailed glyph structures. Compared with existing component-based methods, CG-GAN has two outstanding properties:1) the performance improvement is gained by providing more effective supervision for the generator, not by struggling to increase the complexity of the generator; and 2) the generator is able to capture local style patterns without explicit dependency on the predefined component categories, showing remarkable one-shot Chinese font generation and cross-lingual font generation abilities.

Extensive experiments demonstrate that our proposed CG-GAN significantly outperforms state-of-the-arts in one-shot font generation. Furthermore, by coupling the component-level guidance with a novel framework design, CG-GAN can flexibly extend to two other different tasks: handwriting generation and scene text editing, producing stunning results that far exceed our expectations, indicating the significant potential of our proposed method.

2. Related works

2.1. Image-to-Image Translation

Image-to-image (I2I) translation aims to translate an input image in the source domain into a corresponding output image in the target domain. Pix2pix [14] is the first general framework for the I2I translation task, which is in supervised learning and built upon the conditional adversarial network [24]. However, paired training data are unavailable for many scenarios. Therefore, several methods are proposed to tackle the unpaired setting. UNIT [19] is an extension of CoGAN [21], which learns the joint distribution across domains with the combination of generative adversarial networks [8] and variational autoencoders [18]. One notable work is CycleGAN [37], which tackles the unsupervised image translation problem by introducing the cycle consistency loss. Concurrent to CycleGAN [37], DiscoGAN [17] and DualGAN [34] also utilize the cycle consistent constraint to implement unpair I2I translation. The above approaches are limited to translating images between

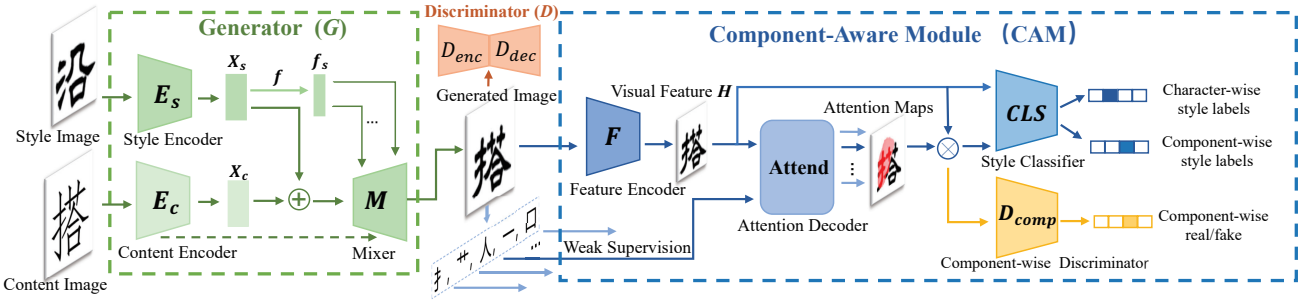


Figure 2. Overview of the proposed method.

two classes. Later, [1, 5, 13] are proposed to achieve multi-class unsupervised I2I translation, capable of translating images among multiple seen classes. FUNIT [20] further extends its generalization ability to unseen classes by learning to encode the content images and class images respectively.

2.2. Few-shot Font Generation

Few-shot font generation (FFG) aims to create a complete font library in the required style given only a few reference images. Several methods [4, 22, 26, 38] regard the FFG task as an I2I translation problem as both tasks learn a mapping from the source domain to the target domain. For instance, "Rewrite" [26] is built upon the pix2pix framework and learns to transfer only one font style. DC-font [15] learns the transformation relationship between two fonts in deep space via the feature reconstruction network. However, all of them cannot generalize to unseen styles. After that, EMD [36] and SA-VAE [28] are proposed to separate the representations for styles and contents, and are thus generalizable to unseen styles. However, they fail to capture local style patterns. Later, some component-based methods [3, 12, 25, 31] are proposed. For example, RD-GAN [12] can generate unseen glyphs in a one-shot setup by introducing a radical extraction module, but can only transfer to a fixed target font style. [3, 25] improve the generative quality by learning component-wise style representation, where DM-Font [3] introduces a dual memory structure and LF-Font [25] utilizes a factorization strategy. However, they suffer from estimation errors and cannot generalize to cross-lingual font generation due to the explicit dependency on the components labels. All of the above works are in supervised learning and require paired training data for strong supervision. After that, DG-Font [32] achieves unsupervised learning by introducing a deformation skip connection, but the results often contain characteristic artifacts. In contrast, our proposed CG-GAN addresses the above issues by employing the component supervision mechanism.

3. Methodology

The overall architecture of our proposed CG-GAN is shown in Figure 2, which consists of a generator G, a Component-Aware Module (CAM) and a discriminator D. The generator G aims to implement style-content disentanglement at the component level. To this end, CAM is employed to provide component-level feedback to the generator through multi component-level discrimination outputs. A U-Net based discriminator [27] D is also employed to perform per-image and per-pixel discrimination, further enhancing the quality of the generated glyph.

3.1. Generator

As shown in Figure 2, the generator consists of a style encoder, a content encoder and a mixer. Given a style image I_s and a content image I_c, the generated image I_g should present the font style of I_s while maintaining the same underlying structure of I_c. Specifically, the content encoder encodes the input content image into a style-invariant content feature map X_c. Meanwhile, the style encoder is employed to extract the style representation at two different levels from the style reference image: a style feature map X_s and a style latent vector f_s. Here, X_s is extracted from the style reference image, and is later mapped to a style latent vector f_s through a mapping network f, which is implemented using a multi-layer perception (MLP).

Finally the mixer is employed to integrate the style and content representation and reconstruct the target image. The style feature map X_s and the content feature map X_c, which have the same spatial dimensions, are concatenated in the channel-wise dimension and are later fed to the mixer. Meanwhile, the style latent vector f_s is injected into each up-sampling block of the mixer M through the AdaIN [11] operation. In addition, we adopt the skip-connection between the content encoder and the mixer, where the output of every down-sampling block in the content encoder is concatenated to the input of the up-sampling block that has the same resolution in the mixer.

3.2. Component-Aware Module

Intuitively, a glyph font style and structure are closely related to the component information. However, most existing methods [15, 26, 36, 38] employ pixel-level strong supervision while ignoring the critical component information. Therefore, we introduce the Component-Aware Mod-

ule (CAM), where the main idea is to make full use of the component information to better guide the font generation process. Hence, CAM is designed to supervise the generator at the component level using the following strategies:

Component extraction A prerequisite for font generation is to preserve the detailed structure of the target glyph. Therefore, the component extraction process aims to supervise whether the glyph structure is correctly transferred. Since every Chinese glyph can be decomposed into a unique component set following a depth-first reading order, we treat the component extraction process as a sequential problem. To proceed, the CNN-based feature encoder $\mathcal{F}$ extracts high-level visual features from the input image $\mathbf{x}$: $\boldsymbol{H} = \mathcal{F}(\boldsymbol{x})$, where $\boldsymbol{H}$ has a spatial dimension of $C \times H \times W$. Thus the encoder output $\boldsymbol{H}$ is a feature vector of $L = H \times W$ elements, where each element h_i is a C-dimensional vector that represents its corresponding region in the input image, denoted by $\boldsymbol{H} = \{h_1, h_2, ..., h_L\}$.

Compared with other sequential learning methods, *e.g.*, CTC [9] and Transformer [29], the attention mechanism is particularly well suited for our purposes due to its efficiency and ease of convergence. Therefore, we adopt the attention-based decoder $\mathcal{A}$ to generate the component sequences, denoted by $\boldsymbol{Y} = \{y_1, y_2, ..., y_T\}$, where T is the length of the component sequence. Note that the length of the component sequence is variable. The decoder predicts the output sequence one symbol at a time until an end-of-sequence token $'EOS'$ is predicted. At every time step t, output y_t is,

$$y_t = Softmax(W_o x_t + b_o), \tag{1}$$

where x_t is the output vector at time step t. We update x_t alongs with the hidden state s_t using a gated recurrent unit (GRU) as:

$$(x_t, s_t) = \mathrm{GRU}((g_t, y_{prev}), s_{t-1}), \tag{2}$$

where (g_t, y_{prev}) is the concatenation of the glimpse vectors g_t and the embedding vetors of the previous output y_{t-1}; g_t is calculated through the attention mechanism as follows:

$$y_{prev} = \mathrm{Embedding}(y_{t-1}), \tag{3}$$

$$e_t = \tanh(s_{t-1} W_s + y_{prev} W_y + b), \tag{4}$$

$$\alpha_{t,i} = \exp(e_{t,i}) \,/\, \sum_{j=1}^{L} (\exp(e_{t,j})), \tag{5}$$

$$g_t = \sum_{i=1}^{L} (\alpha_{t,i} h_i), \tag{6}$$

where W_o, b_o, W_s, W_y and b are trainable parameters; h_i denotes the i-th feature vector in the input feature map $\boldsymbol{H}$.

Using only component labels as weak supervision, the attention-based decoder is able to localize every component by minimizing the structure retention loss. Note that $\mathcal{F}$ and $\mathcal{A}$ are only optimized with the real samples I_s, denote as:

$$\mathcal{L}_{strc}^{CAM} = \mathbb{E}_{I_s \in P_s} \Big[- \sum_{i=1}^{T} \widehat{y}_t \log(\mathcal{A}(\mathcal{F}(I_s)))_t \Big], \tag{7}$$

where $\widehat{y}_t$ denotes the corresponding ground-truth component label at time step t. As shown in Figure 3, at every time step t, the decoder is able to focus on the corresponding component region. Thus, if the component prediction of the generated glyph goes wrong, the generator G is penalized for an incorrect structure transfer, denote as:

$$\mathcal{L}_{strc}^{G} = \mathbb{E}_{I_s \in P_s, I_c \in P_c} \Big[- \sum_{i=1}^{T} \widehat{y}_t \log(\mathcal{A}(\mathcal{F}(G(I_s, I_c))))_t \Big]. \tag{8}$$

In this manner, G is supervised to generate glyph structure at the component level, devoted to preserving every single component correctly. Unlike most existing methods that only extract a global content representation, which often leads to incomplete structures, we employ $\mathcal{L}_{strc}^{G}$ to supervise the content encoder E_c at the component level, guiding E_c to actively decompose the content representation from I_c at the component level. Such a learning scheme is more capable of handling the enormous Chinese categories, as well as preserving the complicated structures.

Multi component-level discrimination We further introduce a style classifier $CLS(\cdot)$ and a discriminator D_{comp} to conduct component-level discrimination. Intuitively, people naturally pay more attention to the local parts/components but less to the entire shape when distinguishing different font styles. Therefore, we conduct the style classification and realism discrimination by utilizing the attention maps $\boldsymbol{A} = \{\alpha_1, \alpha_2, ..., \alpha_T\}$, $\boldsymbol{\alpha_t} \in \mathbb{R}^{H \times W}$ as the label of component regions. To effectively guide the generation process, we conduct multi-component-level discrimination for each input image simultaneously.

3.3. Loss Function

Adversarial loss The generator G aims to synthesize a realistic image that is indistinguishable from real samples. Hence, we adopt a U-Net based discriminator [27], where the encoder part D_{enc} and the decoder part D_{dec} perform per-image and per-pixel discrimination, respectively. Therefore, the generator now has to fool both D_{enc} and D_{dec} via the adversarial loss:

$$\mathcal{L}_{adv} = \mathcal{L}_{adv}^{enc} + \lambda_{dec} \mathcal{L}_{adv}^{dec}, \tag{9}$$

$$\mathcal{L}_{adv}^{enc} = \mathbb{E}_{I_s \in P_s, I_c \in P_c} [\log D_{enc}(I_s) + \log(1 - D_{enc}(G(I_s, I_c)))], \tag{10}$$

$$\mathcal{L}_{adv}^{dec} = \mathbb{E}_{I_s \in P_s, I_c \in P_c} \Big[\sum_{i,j} \log[D_{dec}(I_s)]_{i,j} + \sum_{i,j} \log(1 - [D_{dec}(G(I_s, I_c))]_{i,j}) \Big], \tag{11}$$

where $[D_{dec}(\cdot)]_{i,j}$ denotes the discrimination output at position(i, j). We set λ_{dec} to 0.1 in our experiments.

Style matching loss In addition to using structure retention loss to supervise the structure correctness (Sec. 3.2), the generated image should also maintain both global and local style coherence. To this end, the style classifier $CLS(\cdot)$ performs the style classification over the whole input image to ensure global style coherence, along with performing this classification on a per-component basis to estimate the local style consistency. Therefore, the style matching loss is computed by considering the above two aspects simultaneously. Given the 2D-attention map α_t at time step t and the corresponding style label w of the reference style image I_s, the style matching loss is defined as:

$$\mathcal{L}_{sty}^{CAM} = \mathbb{E}_{I_s \in P_s}\Big[- w \log(CLS(\mathcal{F}(I_s)) - \sum_{t=1}^{T} w \log(CLS(\alpha_t \otimes \mathcal{F}(I_s))\Big]. \tag{12}$$

Here, $\otimes$ refers to an element-wise multiplication. Note that $CLS(\cdot)$ is only optimized with real samples I_s, thus it can guide the generator to synthesize images with a highly similar font style w to the reference image I_s. Correspondingly, the generator is optimized by minimizing:

$$\mathcal{L}_{sty}^{G} = \mathbb{E}_{I_s \in P_s, I_c \in P_c}\Big[- w \log(CLS(\mathcal{F}(G(I_s, Ic)) - \sum_{t=1}^{T} w \log(CLS(\alpha_t \otimes \mathcal{F}(G(I_s, I_c))))\Big]. \tag{13}$$

Essentially, the $\mathcal{L}_{sty}^{G}$ enforces the style encoder E_s to disentangle the style representation at the component level, thus enabling the E_s to capture diverse local styles while maintaining global style coherence. Notably, the $\mathcal{L}_{sty}^{G}$ results in a more powerful style encoder E_s, which can accurately encode local style patterns from any reference style sample I_s without accessing the corresponding component labels.

Component realism loss Furthermore, a discriminator D_{comp} is additionally employed to classify each component patch into being real or fake, further supervising the visual verisimilitude of I_{gen} at the component level, denoted as:

$$\mathcal{L}_{comp} = \mathbb{E}_{I_s \in P_s, I_c \in P_c}\Big[\log D_{comp}(\mathcal{F}(I_s)) + \sum_{i=1}^{T} \log(1 - D_{comp}(\alpha_t \otimes \mathcal{F}(G(I_s, I_c)))\Big], \tag{14}$$

encouraging the generator to pay more attention to local realism of the generated glyph images.

Identity loss We additionally adopt the identity loss to guarantee the identity mapping in the generator G: the generator G is able to reconstruct the style reference image I_s when I_s is also provided as the content input, *i.e.*,

$$\mathcal{L}_{idt} = \mathbb{E}_{I_s \in P_s} \left\| I_s - G(I_s, I_s) \right\|_1. \tag{15}$$

This identity loss stabilizes the training process to a certain extent, as it avoids an excessive style transfer.

Content loss We adopt a content loss to guarantee that the extracted content representation X_c is style-invariant, denoted as:

$$\mathcal{L}_{cnt} = \mathbb{E}_{I_s \in P_s, I_c \in P_c} \left\| X_c - E_c(G(I_s, I_c)) \right\|_1. \tag{16}$$

Full objective Finally, the discriminator D, the Component-Aware Module CAM and the generator G of our proposed CG-GAN are optimized respectively as

$$\mathcal{L}_D = -\mathcal{L}_{adv}, \mathcal{L}_{CAM} = -\mathcal{L}_{comp} + \mathcal{L}_{strc}^{CAM} + \mathcal{L}_{sty}^{CAM}, \tag{17}$$

$$\mathcal{L}_G = \mathcal{L}_{adv} + \mathcal{L}_{comp} + \mathcal{L}_{strc}^{G} + \mathcal{L}_{sty}^{G} + \mathcal{L}_{idt} + \lambda_{cnt}\mathcal{L}_{cnt}. \tag{18}$$

The entire framework is trained from scratch in an end-to-end manner. We set λ_{cnt} to 10 in our experiments.

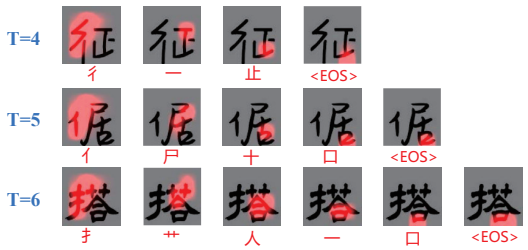


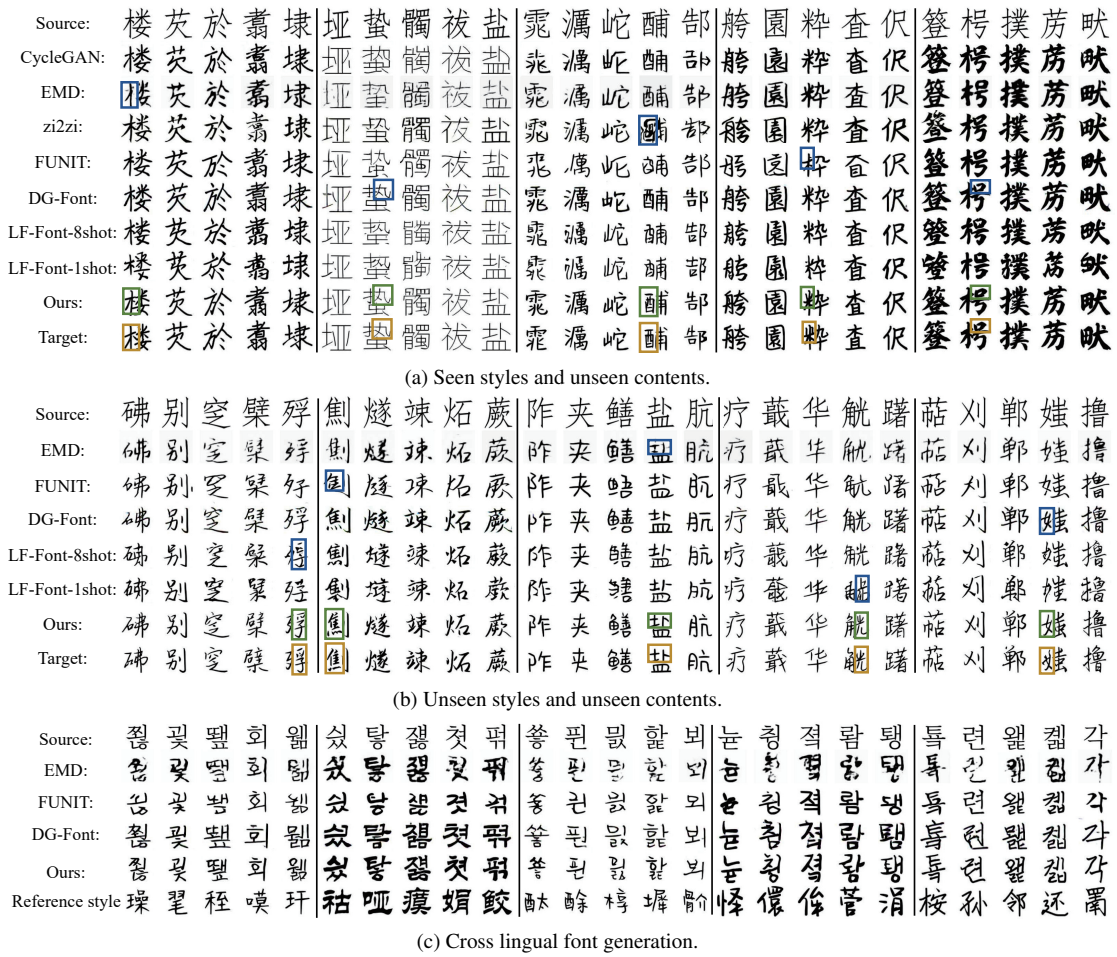
Figure 3. Visualization of attention maps on different lengths of component sequences. Symbols below the images are the predicted components.

4. Experiments

4.1. Chinese font generation

Datasets To evaluate our method with the Chinese font generation task, we collect a dataset containing 423 fonts. We randomly select 399 fonts as the training set (*i.e.* seen fonts), where each font contains 800 Chinese characters (*i.e.* seen characters) that can be decomposed by 385 components. We evaluate the one-shot Chinese font generation ability on two test sets: One is the 399 seen fonts with 150 unseen characters per font. The other is the remaining 24 unseen fonts with 200 unseen characters per font. We additionally evaluate the generalization ability to the unseen language glyphs by using a Korean glyph test set consisting of the 24 unseen fonts with 200 Korean characters per font.

Comparison with state-of-the-art methods We compared our model with six state-of-the-art methods, including four few-shot Chinese font generation methods (zi2zi [38], EMD [36], LF-font [25], DG-Font [32]), and two unsupervised image-to-image translation methods (CycleGAN [37], FUNIT [20]). For a fair comparison, we use



(a) Seen styles and unseen contents.

(b) Unseen styles and unseen contents.

(c) Cross lingual font generation.

Figure 4. Comparisons with the state-of-the-art methods for font generation.

the font of Song as the source font which is a common setting in the font generation task [25, 32, 36]. Since CycleGAN can only learn the mapping from one domain to another at a time, we train a total of 399 CycleGAN models. LF-Font exhibits a low visual quality in inference if only one reference sample is provided. Hence, we evaluate its performance in an eight-shot setup (its original setting) and a one-shot setup respectively. All models are trained from scratch using their official code.

Evaluation metrics We use several metrics for quantitative evaluation. First, SSIM and RMSE are employed to measure whether pixel-level details can be preserved, and a higher SSIM and a lower RMSE represent less image distortion of the generated images. Second, LPIPS [35] is adopted to quantify the perceptual similarity; where a lower LPIPS denotes the generated image is more in line with human visual perception. Third, FID [10] is employed to measure whether the model can match the target data domain distribution. A lower FID represents a higher quality and variety of the generated images. Finally, a user preference study is conducted to quantify the subjective quality of output images. We randomly select 30 seen fonts and 20 unseen fonts from the two Chinese glyph test sets. At each time, the participants are shown the reference style image along with n generated samples generated by the n different methods, and asked to pick the best result. In total, we have collected 2,400 responses in two scenarios of seen styles and unseen styles respectively, contributed by 48 participants.

Quantitative comparison The quantitative results are shown in Table 1. Except for the LF-Font-eight-shot, all the reported results are tested in a one-shot setting. As shown in Table 1, CG-GAN achieves the best performance on all the evaluation metrics for both seen styles and unseen styles. Particularly, CG-GAN outperforms previous state-of-the-arts with significant gaps in both perceptual-level metrics and human visual preference, *e.g.*, achieves 8.92 lower FID in seen styles and 10.87 lower FID in unseen styles than the second-best LF-Font-8-shot, and gain more than 60% of the visual choice under both scenarios. Notably, with just one shot, CG-GAN still outperforms the second-best LF-Font-eight-shot, which further demonstrates the powerful generation ability of our proposed method.

Table 1. **Quantitative evaluation on the whole dataset**. We evaluate methods on seen styles and unseen contents, unseen styles and unseen contents. The bold number indicates the best.

Methods	SSIM↑	RMSE↓	LPIPS↓	FID↓	User preference (%)
Seen styles and **Unseen** contents					
cycleGAN [37]	0.7092	0.0247	0.3010	64.85	4.48
FUNIT [20]	0.7269	0.0244	0.2720	57.72	5.24
zi2zi [38]	0.7666	0.0216	0.2268	59.79	3.43
EMD [36]	0.7519	0.0213	0.2536	61.29	0.10
DG-Font [32]	0.7697	**0.0212**	0.2079	41.56	8.19
8-LF-Font [25]	0.7535	0.0223	0.2227	15.46	13.81
1-LF-Font [25]	0.7427	0.0232	0.2499	19.36	-
CG-GAN (ours)	**0.7703**	**0.0212**	**0.1919**	**6.54**	**64.76**
Unseen styles and **Unseen** contents					
FUNIT [20]	0.7074	0.0249	0.2892	64.68	3.90
EMD [36]	0.7373	0.0219	0.2620	84.84	0.19
DG-Font [32]	0.7553	0.0221	0.2195	55.73	13.52
8-LF-Font [25]	0.7419	0.0225	0.2295	28.81	8.00
1-LF-Font [25]	0.7310	0.0232	0.2529	33.49	-
CG-GAN (ours)	**0.7568**	**0.0218**	**0.2058**	**17.94**	**74.38**

Qualitative comparison In Figure 4 (a) and (b), we provide a visual comparison of seen styles and unseen styles, which intuitively explains the significant gaps of CG-GAN in the user preference study. For these two challenging scenarios, our method generates glyph images of higher quality than state-of-the-arts, particularly better satisfying both style consistency and structure correctness. CycleGAN and FUNIT often produce results in an incomplete structure. EMD often produces severe blur and an unclear background. Zi2zi loses some detailed structure if the target glyph is complex. LF-Font suffers from a significant decrease in visual quality if only one reference image is provided. DG-Font generates glyphs containing characteristic artifacts, which can be observed in the highlighted region. As shown in Figure 4 (c), we further test the generalization ability to unseen components, *i.e.*, cross-lingual font generation. Owing to a stronger representation capability, our model shows superior cross-lingual FFG performance.

4.2. Handwriting generation

By coupling component-level supervision with a novel framework design, CG-GAN can be directly applied to handwriting generation task without any adjustment. To evaluate this, we conduct experiments on IAM handwriting dataset [23]. IAM dataset consists of 9,862 text lines with 62,857 handwritten words, contributed by 500 different writers. In our experiments, only the training and validate sets are used for model training, and the test set is kept apart for evaluation. For a fair comparison, we evaluate our methods with the state-of-the-art handwriting generation methods under the following two scenarios:

Writer-relevant handwriting generation Following previous studies [2, 16], we first evaluate the writer-relevant scenario, where FID is calculated for each writer between its corresponding generated samples and real samples, and finally average the sum of FID of all writers. Thus the final FID scores evaluate the generative quality and simultaneously, the style imitation capability. We use HWT [2] and GANwriting [16] as our baselines, which can synthesize images with referenced style. Shortly, HWT is a transformer-based method that can synthesize arbitrary-length texts. GANwriting can generate short word images with no more than ten letters. We evaluate the competing methods in four different settings: IV-S, IV-U, OOV-S, OOV-U, respectively.

Table 2. **Writer-relevant handwriting generation quality comparison.** All four settings: In-Vocabulary words and seen style (IV-S), In-Vocabulary words and unseen style (IV-U), Out-of-vocabulary words and seen styles (OOV-S), Out-of-vocabulary words and unseen styles (OOV-U).

	IV-S	IV-U	OOV-S	OOV-U
GAN writing [16]	120.07	124.30	125.87	130.68
HWT [2]	106.97	**108.84**	109.45	114.10
CG-GAN (ours)	**102.18**	110.07	**104.81**	**113.01**

Table 3. **Writer-irrelevant handwriting generation quality comparison.** Writer identity is ignored when calculating FID.

Method	ScrabbleGAN [6]	HiGAN [7]	HWT [2]	CG-GAN (ours)
FID	23.78	**17.28**	19.40	19.03

As shown in Table 2, our proposed CG-GAN shows comparable performance with the state-of-the-arts. We surpass the second-best HWT under three settings of IV-S, OOV-S, and OOV-U. Particularly for the most challenging one, where both words and styles have never been seen during training(OOV-U), CG-GAN still achieves around 1.0 lower FID than the second-best HWT. Note that both HWT and GANwriting use 15 style reference images for training, and their proposed results are tested in a 15-shot setting, whereas our method is trained and tested under only a 1-shot setting.

Writer-irrelevant handwriting generation We further evaluate the writer-irrelevant scenario, where writer identity is ignored when calculating the FID score. Under this scenario, we use HWT [2], ScrabbleGAN [6] and HiGAN [7] as our baselines. Briefly, ScrabbleGAN can synthesize long texts with random styles but cannot imitate referenced styles. HiGAN can synthesis an arbitrary-length text either with a random or referenced style. As shown in Table 3, our method achieves comparable performance with the state-of-the-arts. A visual comparison is shown in Figure 5.

4.3. Ablation study

We perform multiple ablation studies to evaluate the effectiveness of our proposed CAM on the one-shot Chinese font generation task. The results are tested on the unseen

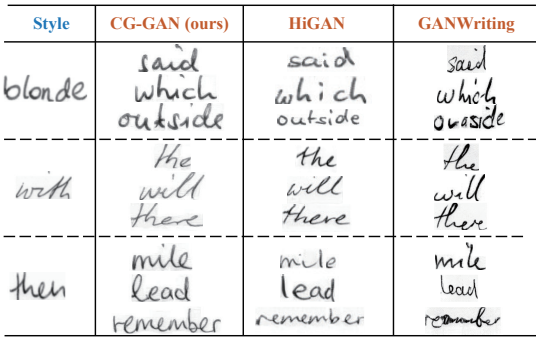


Figure 5. Visual comparison for synthesizing handwritten words.

styles test dataset.

Effectiveness of the Component-level supervision We compare our component-level supervision with commonly used pixel-level supervision and character-level supervision. Pixel-level supervision is performed by removing the CAM module and replacing the component-level objectives with the L1 loss. Note that pixel-level supervision is trained using paired data which uses the same reference style images as our unpaired data settings. Character-level supervision is implemented by replacing the component labels with the character labels. In this manner, the loss supervision is conducted at the character level. As shown in Table 4, we can see that the quantitative results obviously improve in terms of SSIM, RMSE, LPIPS and FID, which demonstrate the effectiveness of our proposed component-level supervision.

Effectiveness of the Component-Aware Module We further analyze the influence of each component-level supervision provided by CAM. First, we build a baseline, remove the CAM module and replace it with a style classifier at the image level, and thus the baseline no longer contains any component-level supervision. Next, we successively add different parts of the multi-component-level supervision and analyze their impact, including structure retention loss, style matching loss and component realism loss. The results are reported in Table 5, we can observe that all of our proposed component-level objective functions are essential, the addition of each objective can make a further improvement on both visual quality and quantitative results.

5. Extension

Our framework can be further extended to the scene text editing (STE) task, which is challenging due to large variations in font style, text shape and background. Existing STE methods [30, 33] generally approach this task in two stages: first rendering the target textual content to obtain the text-modified foreground, and erasing the original text to obtain the text-erased background, finally fusion the two to obtain the desired target image. However, these two-stage methods do not generalize well to real-world scene

Table 4. Effectiveness of the Component-level supervision.

Method	SSIM↑	RMSE↓	LPIPS↓	FID↓
pixel-level	0.7479	0.0223	0.2298	51.44
character-level	0.7529	0.0223	0.2142	33.21
component-level	**0.7568**	**0.0218**	**0.2058**	**17.94**

Table 5. Effectiveness of the Component-Aware Module.

	SSIM↑	RMSE↓	LPIPS↓	FID↓
baseline	0.7517	0.0225	0.2251	49.09
$+\mathcal{L}_{strc}$	0.7487	0.0227	0.2138	22.33
$+\mathcal{L}_{strc}+\mathcal{L}_{sty}$	0.7532	**0.0216**	0.2084	18.67
$+\mathcal{L}_{strc}+\mathcal{L}_{sty}+\mathcal{L}_{dcomp}$	**0.7568**	0.0218	**0.2058**	**17.94**

text images due to strong mutual interference of the background and foreground. By contrast, our framework abandons the inefficient multi-stage rendering and alleviates the intervention problem with the help of component-level supervision. As shown in Figure 6, our framework generates quite promising results that exceed our expectations, showing the impressive potential of our proposed approach. The implementation details are shown in the Appendix A.

(a) Original image (b) Edited text image

Figure 6. Visualization of scene text editing results. (a) and (b) are original text images and text edited images with different text contents and lengths.

6. Conclusion

In this paper, we propose a simple yet effective CG-GAN for one-shot font generation. Specifically, we introduce a CAM to supervise the generator. The CAM decouples the style and content at a more fine-grained level, *i.e.*, the component level, to guide the generator to achieve more promising representation ability. Furthermore, to our best knowledge, CG-GAN is the first FFG method that can be potentially extended to both handwriting word generation and scene text editing, showing its generalization ability.

Acknowledgement

This research is supported in part by NSFC (Grant No.: 61936003) and GD-NSF (No.2017A030312006, No.2021A1515011870).

References

[1] Asha Anoosheh, Eirikur Agustsson, Radu Timofte, and Luc Van Gool. ComboGAN: Unrestrained Scalability for Image Domain Translation. In *CVPRW*, pages 783–790, 2018. 3

[2] Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, and Mubarak Shah. Handwriting Transformers. In *ICCV*, pages 1086–1094, 2021. 7

[3] Junbum Cha, Sanghyuk Chun, Gayoung Lee, Bado Lee, Seonghyeon Kim, and Hwalsuk Lee. Few-shot Compositional Font Generation with Dual Memory. In *ECCV*, pages 735–751, 2020. 1, 2, 3

[4] Jie Chang, Yujun Gu, Ya Zhang, Yan-Feng Wang, and CM Innovation. Chinese handwriting imitation with hierarchical generative adversarial network. In *BMVC*, 2018. 3

[5] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. In *CVPR*, pages 8789–8797, 2018. 3

[6] Sharon Fogel, Hadar Averbuch-Elor, Sarel Cohen, Shai Mazor, and Roee Litman. ScrabbleGAN: Semi-Supervised Varying Length Handwritten Text Generation. In *CVPR*, pages 4324–4333, 2020. 7

[7] Ji Gan and Weiqiang Wang. HiGAN: Handwriting Imitation Conditioned on Arbitrary-Length Texts and Disentangled Styles. In *AAAI*, pages 7484–7492, 2021. 7

[8] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014. 2

[9] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, pages 369–376, 2006. 4

[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *NeurIPS*, pages 6626–6637, 2017. 6

[11] Xun Huang and Serge Belongie. Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization. In *ICCV*, pages 1501–1510, 2017. 3

[12] Yaoxiong Huang, Mengchao He, Lianwen Jin, and Yongpan Wang. RD-GAN: Few/Zero-Shot Chinese Character Style Transfer via Radical Decomposition and Rendering. In *ECCV*, pages 156–172, 2020. 3

[13] Le Hui, Xiang Li, Jiaxin Chen, Hongliang He, and Jian Yang. Unsupervised Multi-Domain Image Translation with Domain-Specific Encoders/Decoders. In *ICPR*, pages 2044–2049, 2018. 3

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-Image Translation with Conditional Adversarial Networks. In *CVPR*, pages 1125–1134, 2017. 2

[15] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. DCFont: an end-to-end deep chinese font generation system. In *SIGGRAPH Asia Technical Briefs*, pages 1–4, 2017. 3

[16] Lei Kang, Pau Riba, Yaxing Wang, Marçal Rusiñol, Alicia Fornés, and Mauricio Villegas. GANwriting: Content-Conditioned Generation of Styled Handwritten Word Images. In *ECCV*, pages 273–289, 2020. 7

[17] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. In *ICML*, pages 1857–1865, 2017. 2

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 2

[19] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised Image-to-Image Translation Networks. In *NeurIPS*, pages 700–708, 2017. 2

[20] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-Shot Unsupervised Image-to-Image Translation. In *ICCV*, pages 10551–10560, 2019. 3, 5, 7

[21] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *NeurIPS*, pages 469–477, 2016. 2

[22] Pengyuan Lyu, Xiang Bai, Cong Yao, Zhen Zhu, Tengteng Huang, and Wenyu Liu. Auto-encoder guided gan for chinese calligraphy synthesis. In *ICDAR*, pages 1095–1100, 2017. 3

[23] U-V Marti and Horst Bunke. The IAM-database: an English sentence database for offline handwriting recognition. In *ICDAR*, pages 39–46, 2002. 7

[24] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint*, 2014. 2

[25] Song Park, Sanghyuk Chun, Junbum Cha, Bado Lee, and Hyunjung Shim. Few-shot Font Generation with Localized Style Representations and Factorization. In *AAAI*, pages 2393–2402, 2021. 1, 2, 3, 5, 6, 7

[26] Rewrite. https://github.com/kaonashi-tyc/rewrite. 1, 2, 3

[27] Edgar Schonfeld, Bernt Schiele, and Anna Khoreva. A U-Net Based Discriminator for Generative Adversarial Networks. In *CVPR*, pages 8207–8216, 2020. 3, 4

[28] Danyang Sun, Tongzheng Ren, Chongxun Li, Hang Su, and Jun Zhu. Learning to Write Stylized Chinese Characters by Reading a Handful of Examples. In *IJCAI*, pages 920–927, 2018. 3

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017. 4

[30] Liang Wu, Chengquan Zhang, Jiaming Liu, Junyu Han, Jingtuo Liu, Errui Ding, and Xiang Bai. Editing text in the wild. In *ACM MM*, pages 1500–1508, 2019. 8

[31] Shan-Jean Wu, Chih-Yuan Yang, and Jane Yung-jen Hsu. CalliGAN: Style and Structure-aware Chinese Calligraphy Character Generator. In *CVPRW*, 2020. 1, 2, 3

[32] Yangchen Xie, Xinyuan Chen, Li Sun, and Yue Lu. DG-Font: Deformable Generative Networks for Unsupervised Font Generation. In *CVPR*, pages 5130–5140, 2021. 2, 3, 5, 6, 7

[33] Qiangpeng Yang, Jun Huang, and Wei Lin. Swaptext: Image based texts transfer in scenes. In *CVPR*, pages 14700–14709, 2020. 8

[34] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. In *ICCV*, pages 2849–2857, 2017. 2

[35] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*, pages 586–595, 2018. 6

[36] Yexun Zhang, Ya Zhang, and Wenbin Cai. Separating Style and Content for Generalized Style Transfer. In *CVPR*, pages 8447–8455, 2018. 1, 2, 3, 5, 6, 7

[37] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *ICCV*, pages 2223–2232, 2017. 2, 5, 7

[38] Zi2zi. https://github.com/kaonashi-tyc/zi2zi. 1, 2, 3, 5, 7