Few-Shot Font Generation by Learning Fine-Grained Local Styles

Licheng Tang[1*] Yiyang Cai[2*] Jiaming Liu[1†] Zhibin Hong[1] Mingming Gong[3]
Minhu Fan[1] Junyu Han[1] Jingtuo Liu[1] Errui Ding[1] Jingdong Wang[1]
[1]Baidu Inc. [2]University of California, Berkeley [3]University of Melbourne
{tanglicheng,liujiaming03,hongzhibin,fanminhu,liujingtuo,dingerrui,wangjingdong}@baidu.com
frank_cai@berkeley.edu, mingming.gong@unimelb.edu.au

Abstract

Few-shot font generation (FFG), which aims to generate a new font with a few examples, is gaining increasing attention due to the significant reduction in labor cost. A typical FFG pipeline considers characters in a standard font library as content glyphs and transfers them to a new target font by extracting style information from the reference glyphs. Most existing solutions explicitly disentangle content and style of reference glyphs globally or component-wisely. However, the style of glyphs mainly lies in the local details, i.e. the styles of radicals, components, and strokes together depict the style of a glyph. Therefore, even a single character can contain different styles distributed over spatial locations. In this paper, we propose a new font generation approach by learning 1) the fine-grained local styles from references, and 2) the spatial correspondence between the content and reference glyphs. Therefore, each spatial location in the content glyph can be assigned with the right fine-grained style. To this end, we adopt cross-attention over the representation of the content glyphs as the queries and the representations of the reference glyphs as the keys and values. Instead of explicitly disentangling global or component-wise modeling, the cross-attention mechanism can attend to the right local styles in the reference glyphs and aggregate the reference styles into a fine-grained style representation for the given content glyphs. The experiments show that the proposed method outperforms the state-of-the-art methods in FFG. In particular, the user studies also demonstrate the style consistency of our approach significantly outperforms previous methods.

1. Introduction

In the modern era, both computer systems and humans process huge amounts of text information. Fonts, the representations of text, have thus played critical roles in many applications. Therefore, the stylish font generation has its unique commercial and artistic values. However, building commercial font libraries is costly and labor-intensive. The cost is even higher for those languages with a huge amount of characters (Chinese, Japanese Kanji, Korean, Thai, etc.).

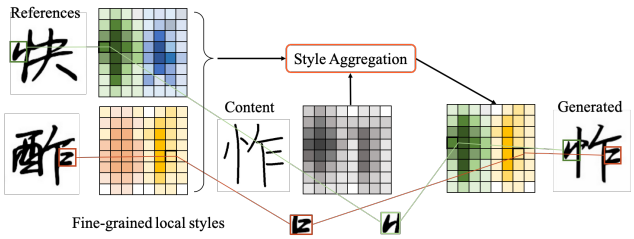

Figure 1. Our proposed fine-grained local style extraction and style aggregation process. Our proposed module enables fine-grained style extraction from references and learns the correspondence between content and reference, thus aggregate corresponding local styles into correct locations in content with high-fidelity.

Due to the expert's high cost of building fonts, automatic font generation with deep learning has drawn rising attention. It aims at generating a brand new font library with only a few characters as a reference. With the development of Generative Adversarial Networks (GANs) [10, 20], there have been many classic works of font generation. Early attempts, such as zi2zi [26], use Pix2Pix [14] like networks with a plug-in font category embedding conditions to learn multiple font styles with a single model. However, these methods require a large number of glyphs to train each unseen font.

In recent years, some works tried to tackle few-shot font generation (FFG) with a few-shot Image-to-Image translation (I2I) scheme [4,8,22,23,31]. Unlike zi2zi, the font style representation is learned from a few reference exemplars, rather than learning embeddings from different font labels. One popular strategy of these works is to explicitly disentangle the content and style representations from given content images and reference exemplars, and two representations are then combined and decoded into the target glyphs. With the advance of these works, the generated glyphs'

*Equal contribution.
†Corresponding author.

quality is significantly improved when the number of references is limited. Based on the explicit disentanglement ideas, the research of FFG can be divided into two different categories, i.e. global style representation and component-wise style representation. The former one models the glyph style as universal representation for each font [8, 18, 31], while the latter one utilizes component-wise style representation from different reference exemplars in the same font [4, 13, 22, 23].

However, in a commercial font, multiple levels of styles need to be considered. An expert would carefully design every possible detail. The detailed styles among components, strokes and, even edges are designed to be consistent. Previous works mostly focus on component-wise styles, while largely ignoring the finer-grained styles. Meanwhile, since the content and style are highly entangled, the commonly used explicit disentanglement can hardly assure the consistency of component-wise styles between the reference glyphs and the generated glyphs. To this end, we employ an references encoder to learn the fine-grained local styles (*FLS*) without explicit disentangled representation learning. Instead of regarding the overall reference map as style, we consider each spatial location of the feature map as a *FLS* representation of reference glyph. After learning the spatial correspondence between references and content, we further acquire the target style map by aggregating the corresponding *FLS*s from references. Each feature vector of the target style map also represents *FLS*s for the target glyph.

In this paper, we propose a novel approach shown in Figure 1, named FSFont for few-shot font generation. The reference glyph images are encoded into reference maps, which represent the *FLS*s of the references. Our proposed cross-attention based style aggregation module (SAM) learns the spatial correspondence between references and the content glyph. The spatial correspondence is not only on the component level but also on the granular level, which contains more detailed local styles. Afterward, SAM aggregates the *FLS*s of references into the target style map, where each spatial location can be assigned to the right fine-grained style. Moreover, to enhance the model to recover details of the references better and learn correspondence more effectively, we adapt a self-reconstruction branch that takes the target character as the input of the reference encoder, and the generated result is supervised by itself. This branch makes learning the correspondence more easily, and helps to produce highly consistent output. Last but not least, we develop a strategy to select the references for each glyph automatically. After analyzing the compositional rules, we design a breadth-first search-based algorithm to search for the reference set and find the optimal references for each character.

In summary, the contribution of this paper is threefold:

- We devise a novel model for few-shot generation. The model extracts the *FLS* of the reference glyph images, and a cross-attention based style aggregation module aggregates the reference styles into a target style map. The details from reference glyphs are thereby transferred to the target glyph.

- We propose a unified training framework with a newly designed self-reconstruction branch. This branch significantly boosts the detail capture ability of the model and improves the output images' quality. As a result, the proposed full model achieves state-of-the-art font generation results.

- We analyze the relationship between characters and select a fairly small set of characters as references. Then we develop a rule to map each character with the elements in the reference set. With the proposed rule, the model's ability to extract component features is better exploited.

2. Related Works

2.1. Image-to-image translation

Image-to-image (I2I) translation refers to the task of learning a mapping function between the source domain and target domain, which preserves the content of the source image while merging the style of the target domain at the same time. According to many I2I methods [6, 7, 14, 17, 18, 32, 34], I2I methods have developed towards multi-mapping [7, 17, 32] and few-shot learning [18]. CycleGAN [34] introduces the cycle-consistency into generative models, which enables I2I methods to train cross-domain translation without paired data. FUNIT [18] accomplish the style transfer task by encoding content and style respectively and combine them with adaptive instance normalization (AdaIN) [12]. From an intuitive thought, font generation is a typical I2I translation task, since it tries to keep the content information of the source font and maps it into the target font. Thus, many font generation methods are based on I2I translation methods.

2.2. Many-shot font generation

Early font generation methods [5, 9, 11, 15, 16, 19, 24–26, 28, 30] aim at learning a mapping function between source fonts and target fonts. When new font references are introduced, these methods use hundreds of reference glyphs to fine-tune the original mapping function. zi2zi [26] and Rewrite [25] train GANs in a supervised way with one-hot style labels. AGEN [19] proposes an model based on the auto encoder to transfer standard font images to calligraphy images. HAN [5] designs skip connection and hierarchical loss functions to improve zi2zi's generation performance.

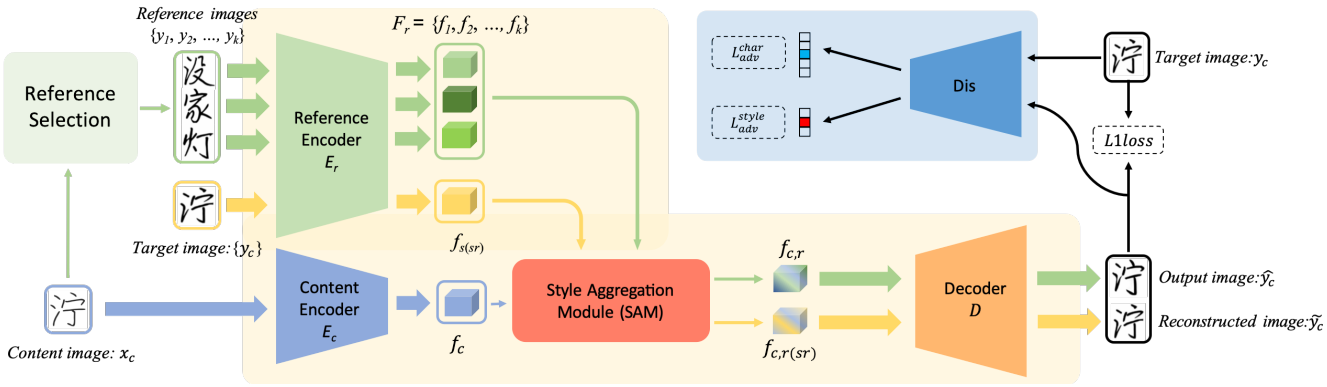

Figure 2. **Overview of our proposed model**. Our generator consists of four parts: a reference encoder E_r, a content encoder E_c, the **Style Aggregation Module(SAM)**, and a decoder D. Given a content image x_c and k-shot references $\{y_1, y_2, ..., y_k\}$ which is selected based on our proposed **Reference Selection**, E_r and E_c extract their features F_r and f_c respectively. Our SAM matches F_r and f_c based on the attention mechanism and links them with spatial correspondence, outputting the target style map $f_{c,r}$. Afterwards, we use D to obtain the generated image $\hat{y}_c$. We also propose a auxiliary branch of **Self-Reconstruction** during training stage (yellow branch) . It shares weights with the main branch and improves generated images' quality in details. A multi-task discriminator is employed to calculate the adversarial loss and simultaneously distinguish the content and style category of the generated character. We also compute the pixel-wise reconstruction loss between the ground truth y_c and the generated image $\hat{y}_c$, and between y_c and the reconstructed image $\tilde{y}_c$, respectively.

These methods require paired data to train a mapping function, e.g,775 for [16]. Other methods [9, 11] focus on unpaired data, and use them for style extraction. Although these many-shot font generation methods have achieved remarkable performance, it is still a laborious task to collect hundreds of references for the fine-tuning process, especially when the reference font library is glyph-rich.

2.3. Few-shot font generation

Most Few-shot font generation (FFG) methods focus on disentangling the content feature and style feature from the given glyphs. Based on different kinds of feature representation, FFG methods can be divided into two main categories: global feature representation [1, 8, 29, 33] and component-based feature representation [4, 13, 22, 23]. In methods that apply global feature representation, vectors related to content and style are extracted from content glyphs and reference glyphs respectively. MCGAN [1] synthesize the ornamented glyphs with stacked conditional GANs to extract features from input images. EMD [33] and AGIS-Net [8] combine a style vector and content vector together to synthesize a glyph. ZiGAN [29] matches the features to Hilbert space to better capture the structural information. Works related to component-based feature representation concentrate on devising a feature representation that is related to the glyphs' components or localized features. RD-GAN [13] uses a radical encoder to extract features of glyphs' specific components. In DM-Font [4], it disassembles glyphs to stylized components and reassembles them to new glyphs by utilizing strong compositionality prior. LF-Font [22] designs a component-conditioned reference encoder to extract component-wise features from reference images. MX-Font [23] employs multiple encoders for each reference image with disentanglement between content and style which makes the cross-lingual task possible. DG-Font [31] is an unsupervised framework based on TUNIT [2] by replacing the traditional convolutional blocks with Deformable blocks which enables the model to perform better on cursive characters which are more difficult to generate.

However, the previous FFG works fail to fully explore the style maps from k-shot references. When receiving k-shot reference images, they tend to explicitly disentangle $style$ and $content$ of images globally or component-wisely and conduct a average operation among extracted features [22, 23]. The local details extracted from each reference are significantly weakened by disentanglement and the average operation. Therefore, we design *Style Aggregation Module* that aims to keep the very detailed features from reference images and to fully utilize the spatial details.

2.4. Attention Mechanism

The attention mechanism [27] is known to capture dependence. After its debut in machine translation, it has been applied in many vision tasks including font generation. RD-GAN [13] using attention mechanism to extract rough radicals from content characters and then render them into target style. HWT [3] uses transformer blocks to bridge the gap between image and text, making it capable of generating stylized handwriting English text images. Our *Style Aggregation Module* is highly motivated by the attention mechanism for fine-grained feature map re-composition.

3. Method

In this section, we present the details of our FS-Font method. We first briefly review the few-shot font generation problem setup and introduce the overall framework of our method (3.1). Next, we present the details of three crucial components of our approach, including the *Style Aggregation Module* (SAM) (Sec. 3.2), the Self-Reconstruction branch (Sec. 3.3), and Reference Selection (Sec. 3.4).

3.1. Problem Setting and Method Overview

In few-shot font generation, given a content glyph image x_c with the content c from a standard font $X = \{x_c\}_{c=1}^{N}$ and a k-shot reference glyph images with the style s: $R_s = \{y_i\}_{i=1}^{k} \subset Y_s$, our goal is to generate a stylized image $\hat{y}_c$ that has the content c and style s via a generator G:

$$\hat{y}_c = G(x_c, R_s), \tag{1}$$

where the style s of $\hat{y}_c$ is omitted for sake of simplicity.

In training, we collect L fonts with different styles $Y = \{Y_s\}_{s=1}^{L}$, where $Y_s = \{y_c\}_{c=1}^{N}$. Therefore, we have a paired dataset $\{x_c, y_c\}_{c=1}^{N}$ for each content c in s-th training style.

The overall framework is shown in Figure 2. The reference encoder E_r first encodes R_s into k-shot reference maps $F_r = \{f_i\}_{i=1}^{k}$, where f_i is encoded from y_i. The content encoder E_c extracts the content feature map f_c from the input content image x_c. Our proposed SAM takes F_r and f_c as inputs, attends to the corresponding spatially local styles in the reference maps F_r and aggregates the local styles into the target style map $f_{c,r} = SAM(f_c, F_r)$. In the end, the decoder D decodes $f_{c,r}$ into the generated output image $\hat{y}_c$. A multi-task projection discriminator [21] is employed to discriminate each generated image and real image. The discriminator outputs a binary classification of fake or real for each character's style and content category.

3.2. Style Aggregation Module

An overview of the SAM is depicted in Figure 3. The core in SAM is a multi-head cross attention block that attends to spatially local styles from the reference maps F_r and aggregates the reference styles into the fine-grained style representation for the given content image. For the m-th attention head, SAM learns a Query map Q^m from content feature map f_c and Key map K^m from reference maps F_r, which achieves the spatial correspondence matrix A^m between the pixels of Q^m and K^m. A value map V^m is simultaneously learned from F_r. Multiplying the correspondence matrix A^m with the value map V^m aggregates the local style styles into the target style S^m. As different head captures different information, we combine all the target styles together for decoding.

Formally, we reshape $f_c \in \mathbb{R}^{c\times h\times w}$ into a sequence $\tilde{f}_c \in \mathbb{R}^{c\times(h\cdot w)}$, where (h, w) is the resolution of the feature

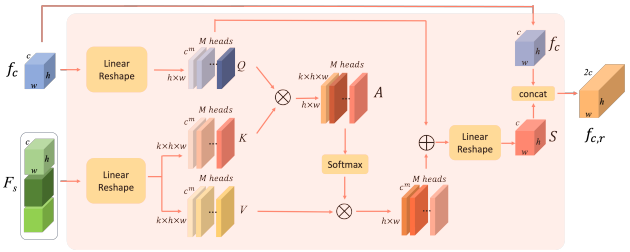

Figure 3. **The illustration of the Style Aggregation Module (SAM)**. We use multihead attention mechanism to calculate the spatial correspondence of content and references and generate the fine-grained feature map.

map, c is the number of channels. For the m-th head, after applying a linear layer $L^m_{query} \in \mathbb{R}^{c\times c^m}$, we acquire the query matrix $Q^m \in \mathbb{R}^{c^m\times(h\cdot w)}$. Meanwhile, the reference maps $F_r = \{f_i\}_{i=1}^{k}$ are reshaped and concatenated along the spatial dimension (h, w), forming a reference sequence $\tilde{f}_r \in \mathbb{R}^{c\times(k\cdot h\cdot w)}$. We multiply $\tilde{f}_r$ by two linear projections $L^m_{key}, L^m_{value} \in \mathbb{R}^{c\times c^m}$ and generate a key map K^m and a value map V^m as follows:

$$\begin{aligned} Q^m &= L^m_{query}(\tilde{f}_c), \quad Q^m \in \mathbb{R}^{c^m\times(h\cdot w)}, \\ K^m &= L^m_{key}(\tilde{f}_r), \quad K^m \in \mathbb{R}^{c^m\times(k\cdot h\cdot w)}, \\ V^m &= L^m_{value}(\tilde{f}_r), \quad V^m \in \mathbb{R}^{c^m\times(k\cdot h\cdot w)}. \end{aligned} \tag{2}$$

We then compute a spatial correspondence matrix A^m of which each element $A^m(u, v)$ is a pairwise feature correlation between content feature in position u and reference feature in position v, calculated as follows:

$$A^m = \frac{Q^{m\top}K^m}{\sqrt{c^m}} \in \mathbb{R}^{hw\times khw}, \tag{3}$$

where c^m is the hidden dimension of Q^m and K^m. The $1/\sqrt{c^m}$ factor follows Transformers [27] to prevent the magnitude of the dot product from growing extreme.

With the correspondence matrix A^m, we obtain the aggregated style from references by

$$S^m = softmax(A^m)V^{m\top} \in \mathbb{R}^{hw\times c^m}. \tag{4}$$

After permuting and reshaping S^m into $\mathbb{R}^{c^m\times h\times w}$, we concatenate all S^m along the channel dimension, and employ a linear projection $L_s \in \mathbb{R}^{(c^m\cdot M)\times c}$ to obtain S. The target style map $f_{c,r}$ is obtained as the concatenation between S and content feature f_c. The decoder D decodes it into the target image $\hat{y}_c$ as follows:

$$\begin{aligned} S &= L_s(S^1 \circ S^2, ..., \circ S^M) \in \mathbb{R}^{c\times h\times w}, \\ \hat{y}_c &= D(f_{c,r}) = D(f_c \circ S), \end{aligned} \tag{5}$$

where $\circ$ denotes concatenation operator and M is the number of total attention heads.

3.3. Self Reconstruction

To achieve a highly style-consistent glyph using SAM, it depends not only on the proper aggregation of styles, but also on the expressivity of the styles that depict fine and local details of the references. However, the k-shot font generation setup is a hard-to-learn problem, as it requires the attention learns spatial correspondence from weakly correlated content-references pairs, which is sometimes confusing even for a human expert. Thus, besides the main branch of k-shot learning, we introduce an easy-to-learn Self-Reconstruction (SR) branch to boost the learning process.

While the generator is trained in the above mentioned k-shot setup, for a given content input x_c, we have the ground truth image y_c to supervise the model training. Self Reconstruction is a parallel branch that shares the same model as the main branch during training. It takes y_c as the reference image $\tilde{R}_s = \{y_c\}$, and its output $\tilde{y}_c$ is also supervised by y_c itself. In contrast to Eq. 1, the generation process of this branch is

$$\tilde{y}_c = G(x_c, \tilde{R}_s). \tag{6}$$

The detailed training setup can be found in Sec.3.5. In the SR branch, the content and reference images are strongly correlated. The spatial correspondence matrix can be easily learned as the strokes and the components' relationship between content and reference is clear. With the well-learned correspondence, the generator can be optimized with well-aligned gradients. As a result, the expressivity of depicting details can be further learned within our framework.

3.4. Reference Selection

In previous works considering the decomposition of components like LF-Font [22], reference characters that contain the same components as content character are randomly selected from the training set during each iteration. The model can hardly learn how to extract the right component-wise features with varying combinations of the reference set. Thus, we introduce a strategy to select a fixed reference set whose components cover most of the commonly used characters and design a content-reference mapping that fixes the combination of reference set for each character. To establish this mapping function, we firstly decompose each character into a component tree, as shown in Figure 4, based on a commonly used decomposition table[1]. We define the components at the level 0, 1, and 2 as the conspicuous components, which contains both radical and compositional structures that easier be transfered from the references to the target.

Reference set selection. This reference set should cover as many conspicuous-level components as possible. Initially, we select a small subset (typically including 100 characters). First, we decompose the characters into component trees. Once the character contains two or more new components, we add this character to our reference set. When the elements in the reference set reach their limits, we stop searching and obtain the style reference set and its corresponding contained components.

[1]https://github.com/cjkvi/cjkvi-ids/blob/master/ids.txt

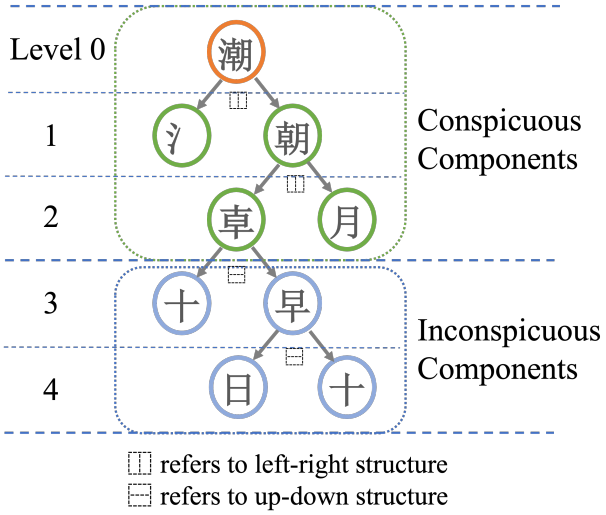

Figure 4. A component tree. To make the mapping process more sensible, the component's structure information is also taken into consideration.

Content-reference mapping. After completing the reference set selection, we then establish the mapping relations between content glyphs and style references. We propose a greedy process to find k-shot references for a glyph. In this process, we search the reference set for k times to establish a mapping relation. During every searching step, we find the reference glyph that shares the most components with the target glyph. If there are multiple solutions, we select the optimal solution that has the most components with the same structure composition. After selection, we remove the reference from the reference set and continue the next searching step. By this process, we can determine every glyph's corresponding k-shot references.

3.5. Training

We train our model to generate the image $\hat{y}_c$ from a content glyph image x_c and a fixed set of reference glyph images R_s. In each iteration, the main branch and the SR branch generate $\hat{y}_c$ and $\tilde{y}_c$ simultaneously and are supervised by the same losses. **FSFont** learns the reference encoder E_r, content encoder E_c, Style Aggregation Module and decoder D with following losses: 1) Adversarial loss with the multi-task discriminator. 2) L1 loss among $\hat{y}_c$, $\tilde{y}_c$ and a paired ground truth image y_c.

Adversarial loss. Since we our aim is to generate visu-

ally high-quality images, we employ a multi-head projection discriminator [22] in our framework. The loss function is represented as follows:

$$\begin{aligned}\mathcal{L}_{adv}^{D} &= \mathbb{E}_{y_c \in P_d}[min(0, 1 - D_{s,c}(y_c))] \\ &+ \mathbb{E}_{\bar{y}_c \in P_g}[min(0, -1 - D_{s,c}(\bar{y}_c))], \\ \mathcal{L}_{adv}^{G} &= -\mathbb{E}_{\bar{y}_c \in P_g} D_{s,c}(\bar{y}_c),\end{aligned} \tag{7}$$

where $\bar{y}_c \in \{\hat{y}_c, \tilde{y}_c\}$, $D_{s,c}(\cdot)$ refers to the logits from the character c head and style s head from the discriminator, P_g denotes the set of generated images from both main branch and Self-Reconstruction branch, and P_d denotes the set of real glyph images.

L1 loss. For learning the pixel-level consistency, we employ an L1 loss between the generated images $\hat{y}_c$, $\tilde{y}_c$ and ground truth image y_c.

$$\mathcal{L}_{l1} = \mathbb{E}\left[\|\hat{y}_c - y_c\|_1 + \|\tilde{y}_c - y_c\|_1\right]. \tag{8}$$

Overall objective loss. Combining all losses mentioned above, we train the whole model by the following objective:

$$\min_G \max_D \lambda_{adv}(\mathcal{L}_{adv}^{G} + \mathcal{L}_{adv}^{D}) + \lambda_{l1}\mathcal{L}_{l1}, \tag{9}$$

where λ_{adv} and λ_{l1} are hyperparameters to control the weight of each loss. We empirically set $\lambda_{adv} = 1$ and $\lambda_{l1} = 0.1$ in our experiments.

4. Experiments

In this section, we evaluate FSFont for the representative Chinese font generation task. We first introduce the datasets we used and compare our framework with other state-of-the-art(SOTA) methods. After that, an ablation study evaluates the effectiveness of each module in our framework and how they affect final results.

4.1. Datasets and evaluation metrics

Datasets. We choose 407 fonts and 3396 commonly used Chinese characters as our datasets including handwriting fonts, printed fonts as well as artistic fonts. All images are 128×128 pixels. We select 100 characters from datasets as our reference set and create a Content-Reference mapping with the strategy discussed in Sec.3.4. Reference set and Content-Reference mapping are fixed in both training set and testing set which means only 100 characters are needed to generate a new font library. The training set contains 397 fonts, 2896 characters. The test set are from 10 representative fonts including typewriter fonts, artistic fonts, and handwriting fonts to evaluate the generalization of our model on variant unseen fonts. We test the methods with two setups *Unseen Fonts Unseen Characters*(UFUC) and *Unseen Fonts Seen Characters*(UFSC). UFUC refers to the 500 characters in the test fonts, and UFSC refers to the 2896 characters in the test fonts.

Evaluation metrics. To evaluate the similarity between generated images and ground truth, we compare our framework with other SOTA methods in the following metrics, i.e. **L1**, **RMSE**, **SSIM** and **LPIPS**. Additionally, we conduct user studies to calculate the **Character Accuracy**, as CNN-based classifiers are tolerant of little defects like missing or broken strokes and blurry edges in a stroke-rich character. We hire 51 volunteers to rigorously count the correct ones from 500 generated characters of each method. A character will be counted as correct only if the volunteer can not spot a defect. For **Style Consistency**, the volunteers are required to evaluate which of the methods generates the most similar character given a set of reference glyph images from 30 randomly selected cases.

4.2. Comparison methods

We compare our method with previous SOTA methods. 1) **FUNIT** [18] is a early work on Few-shot image translation in an Unsupervised way which introduces two different encoders and an AdaIN module to generate a new image with mixed content and style. 2) **DG-Font** [31] is an Unsupervised network using Deformable Convolution in Generator to achieve a better effect on cursive characters 3) **MX-Font** [23] adopts multiple experts to extract different local structures which makes cross-lingual generation task possible. (4) **AGIS-net** [8] uses two different decoders to generate images with shape and texture information which makes generated image more stable. (5) **LF-Font** [22] proposes localized style representation which makes it enable to extract the component-wise features. For a fair comparison, we choose the Kaiti Line Font [2] as the standard font and re-train all models on the training datasets in Sec. 4.1.

4.3. Experimental results

Quantitative comparison. Table 1 shows the FFG performance of our **FSFont** and other competitors. We conducted the experiment on UFUC and UFSC. FSFont clearly outweighs previous SOTA methods on all of the similarity metrics from pixel-level to perceptual-level. On UFSC setup, MX-Font [23] and AGIS-Net [8] generates little more correct characters than FSFont. However, FSFont still generates the most correct characters on UFUC. AGIS-Net suffers from a performance drop on **Character Accuracy** in UFUC. This may suggest that its generalization to new characters are limited. Meanwhile, the user study of **Style Consistency** of FSFont is remarkably better than other methods, which further verifies that FSFont generates the visually similar results from users' perspective.

[2]Font is available at https://chanind.github.io/hanzi-writer-data/

Unseen Fonts Seen Characters						
Methods	L1 loss ↓	RMSE ↓	SSIM ↑	LPIPS ↓	User(C) % ↑	User(S) % ↑
FUNIT [18]	0.148	0.344	0.565	0.2543	86.0	0.3
DG-FONT [31]	0.131	0.329	0.604	0.2154	92.4	6.1
MX-FONT [23]	0.152	0.347	0.584	0.2291	**97.4**	8.4
AGIS-NET [8]	0.105	0.289	0.651	0.1865	97.2	5.2
LF-FONT [22]	0.129	0.322	0.607	0.2006	93.4	11.3
Ours	**0.097**	**0.268**	**0.671**	**0.1618**	96.6	**68.7**
Unseen Fonts Unseen Characters						
Methods	L1 loss ↓	RMSE ↓	SSIM ↑	LPIPS ↓	User(C) % ↑	User(S) % ↑
FUNIT [18]	0.152	0.345	0.532	0.2424	84.6	1.2
DG-FONT [31]	0.141	0.341	0.573	0.2151	86.4	3.1
MX-FONT [23]	0.153	0.352	0.573	0.2317	93.2	11.2
AGIS-NET [8]	0.114	0.302	0.623	0.1877	89.8	4.1
LF-FONT [22]	0.138	0.334	0.577	0.2018	90.6	13.3
Ours	**0.106**	**0.283**	**0.642**	**0.1627**	**93.8**	**67.1**

Table 1. **Qualitative comparison on UFUC and UFSC datasets**

Figure 5. **Generated results of each method on UFUC datasets.** We represent the generated samples of four different kinds of fonts. We mark the main component of each character with boxes. The blue boxes in Reference are components we hope the model recover. It shows our model(green boxes) is capable of recovering more details from Reference than other SOTA works do(red boxes).

Qualitative comparison We illustrate the generated samples in Figure 5 for each method. We selected four different fonts including typewriter fonts, artistic fonts as well as handwriting fonts to see the generalization of all competitors. As demonstrated in Figure 5, FSFont are available to recover as many details from reference images. Though other methods like LF-Font [22] or AGIS-Net [8] are able to generate stable characters and recover coarse features such as the thickness of strokes and inclination of font, they could not recover details as explicitly as our method does.

Ablation studies. In this part, we discuss the effectiveness of each module we introduce in **FSFont**. We discard each module at a time and train the model with other settings unchanged. The overall evaluation results on UFUC datasets are shown in Table 2. We replace SAM with averaging features in F_r to test its effect. For Reference Selection and Content-Reference mapping, we replace them with the strategy of LF-Font [22] by randomly selecting reference glyph images with a common component set for each content character. Both two modules have a positive effect on the final results. The Self-Reconstruction Branch has a significant impact on outputs. As shown in Table 3, the model trained without SR branch can hardly recover details from reference glyph images.

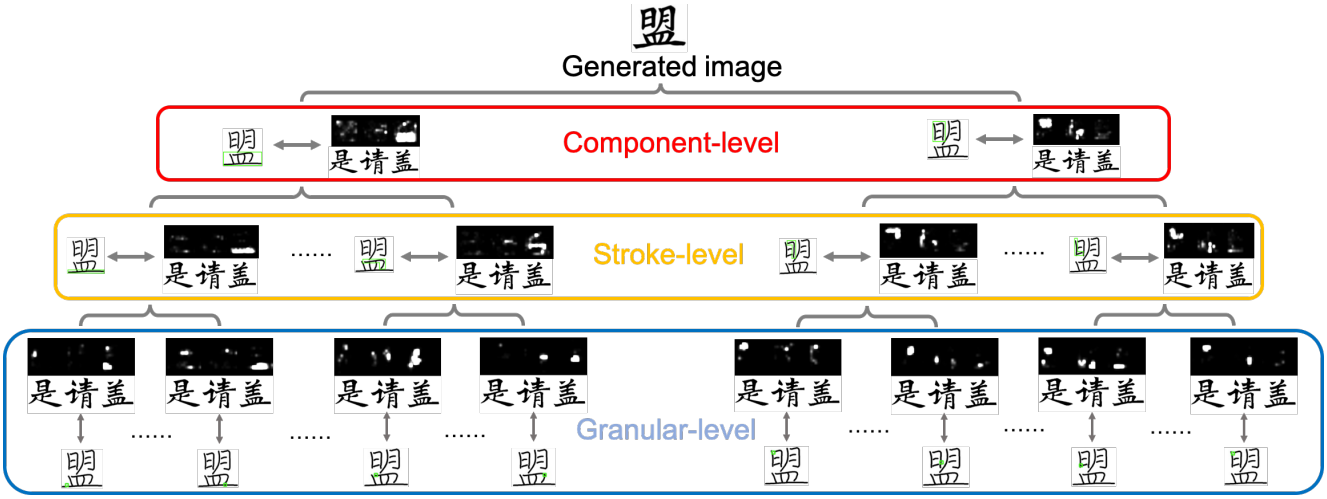

Figure 6. **Visualization of Style Aggregation Module.** The brighter spot in the attention maps denotes the larger contribution of corresponding features in the reference feature maps. The light-green dot, line and bounding box denote the queries from different level.

Unseen Fonts Unseen Characters						
SAM	SR	RS	L1 loss ↓	RMSE ↓	SSIM ↑	LPIPS ↓
✓	✓	✓	**0.106**	**0.283**	**0.642**	**0.1627**
✗	✓	✓	0.113	0.294	0.621	0.1822
✓	✗	✓	0.127	0.318	0.596	0.1859
✓	✓	✗	0.114	0.292	0.624	0.1798
✗	✗	✗	0.131	0.338	0.584	0.2189

Table 2. **Analysis of different modules in our proposed framework.** By discarding the Style Aggregation Module (SAM), Self Construction (SR), and Reference Selection (RS) individually, we can see that all these modules have positive effects on the original model respectively. The model with three modules has the best performance in all evaluation metrics.

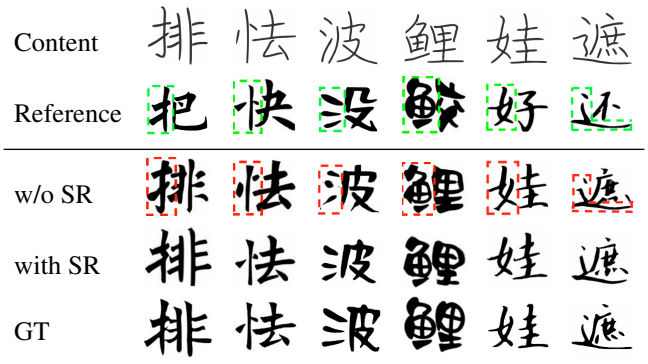

Table 3. **Comparison of generated images with and without Self-Reconstruction branch**. The green box in the reference set is the component we hope the model to recover. The red box shows the insufficient details generated from the model trained without Self Reconstruction branch.

Visualization of SAM. To demonstrate the effectiveness of SAM, we visualize the attention maps for different levels in Figure 6. Specifically, given a certain spatial point q in the content feature map as a query, we can obtain the corresponding correlations $A_q^m \in \mathbb{R}^{khw}$ from the spatial correspondence matrix $A^m \in \mathbb{R}^{hw \times khw}$, and construct an attention map by reshaping A_q^m to $h \times kw$. We consider the queries from the **Granular**, **Stroke**, and **Component-level**, respectively, and compute the final attention map by summing over the attention maps related to the queries. It shows that our SAM module empowers the model to attend to the correct *FLS*s from reference images and extract a sub-component level feature representation for content images.

5. Conclusion

In this paper, we propose a novel FFG model, which is able to calculate the spatial correspondence of the content and reference based on their component features. Our proposed Style Aggregation Module aggregates fine-grained local styles of references to content's corresponding location with high-fidelity. Besides, we propose a Self-Reconstruction branch to help model to recover details from references. Last but not least, our Reference Selection strategy guarantee that each content can match references that share common conspicuous components. Our extensive experiments show that FSFont significantly outperforms other methods in both objective and subjective similarity. **Limitation** The model is trained on limited data, it can not faithfully replicate every detail of the font. **Negative Impact** Though FSFont can be potentially used to imitate handwriting, a human expert can still spot the difference between generated and real handwritings.

References

[1] Samaneh Azadi, Matthew Fisher, Vladimir G Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. Multi-content gan for few-shot font style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7564–7573, 2018. 3

[2] Kyungjune Baek, Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Hyunjung Shim. Rethinking the truly unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14154–14163, October 2021. 3

[3] Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, and Mubarak Shah. Handwriting transformers. *arXiv preprint arXiv:2104.03964*, 2021. 3

[4] Junbum Cha, Sanghyuk Chun, Gayoung Lee, Bado Lee, Seonghyeon Kim, and Hwalsuk Lee. Few-shot compositional font generation with dual memory. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16*, pages 735–751. Springer, 2020. 1, 2, 3

[5] Jie Chang, Yujun Gu, Ya Zhang, Yan-Feng Wang, and CM Innovation. Chinese handwriting imitation with hierarchical generative adversarial network. In *BMVC*, page 290, 2018. 2

[6] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797, 2018. 2

[7] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8188–8197, 2020. 2

[8] Yue Gao, Yuan Guo, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. Artistic glyph image synthesis via one-stage few-shot learning. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019. 1, 2, 3, 6, 7

[9] Yiming Gao and Jiangqin Wu. Gan-based unpaired chinese character image translation via skeleton transformation and stroke rendering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 646–653, 2020. 2, 3

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 1

[11] Ammar Ul Hassan, Hammad Ahmed, and Jaeyoung Choi. Unpaired font family synthesis using conditional generative adversarial networks. *Knowledge-Based Systems*, 229:107304, 2021. 2, 3

[12] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017. 2

[13] Yaoxiong Huang, Mengchao He, Lianwen Jin, and Yongpan Wang. Rd-gan: few/zero-shot chinese character style transfer via radical decomposition and rendering. In *European Conference on Computer Vision*, pages 156–172. Springer, 2020. 2, 3

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei 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. 1, 2

[15] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. Dcfont: an end-to-end deep chinese font generation system. In *SIGGRAPH Asia 2017 Technical Briefs*, pages 1–4. 2017. 2

[16] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. Scfont: Structure-guided chinese font generation via deep stacked networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4015–4022, 2019. 2, 3

[17] Alexander H Liu, Yen-Cheng Liu, Yu-Ying Yeh, and Yu-Chiang Frank Wang. A unified feature disentangler for multi-domain image translation and manipulation. *arXiv preprint arXiv:1809.01361*, 2018. 2

[18] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10551–10560, 2019. 2, 6, 7

[19] Pengyuan Lyu, Xiang Bai, Cong Yao, Zhen Zhu, Tengteng Huang, and Wenyu Liu. Auto-encoder guided gan for chinese calligraphy synthesis. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1095–1100. IEEE, 2017. 2

[20] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1

[21] Takeru Miyato and Masanori Koyama. cgans with projection discriminator, 2018. 4

[22] Song Park, Sanghyuk Chun, Junbum Cha, Bado Lee, and Hyunjung Shim. Few-shot font generation with localized style representations and factorization. *arXiv preprint arxiv:2009.11042*, 2020. 1, 2, 3, 5, 6, 7

[23] Song Park, Sanghyuk Chun, Junbum Cha, Bado Lee, and Hyunjung Shim. Multiple heads are better than one: Few-shot font generation with multiple localized experts. *arXiv preprint arXiv:2104.00887*, 2021. 1, 2, 3, 6, 7

[24] Donghui Sun, Qing Zhang, and Jun Yang. Pyramid embedded generative adversarial network for automated font generation. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 976–981. IEEE, 2018. 2

[25] Yuchen Tian. Rewrite: Neural style transfer for chinese fonts. *https://github.com/kaonashi-tyc/Rewrite*, 2016. 2

[26] Yuchen Tian. zi2zi: Master chinese calligraphy with conditional adversarial networks. *https://github. com/kaonashi-tyc/zi2zi*, 2017. 1, 2

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural*

information processing systems, pages 5998–6008, 2017. 3, 4

[28] Chuan Wen, Yujie Pan, Jie Chang, Ya Zhang, Siheng Chen, Yanfeng Wang, Mei Han, and Qi Tian. Handwritten chinese font generation with collaborative stroke refinement. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3882–3891, 2021. 2

[29] Qi Wen, Shuang Li, Bingfeng Han, and Yi Yuan. Zigan: Fine-grained chinese calligraphy font generation via a few-shot style transfer approach. *arXiv preprint arXiv:2108.03596*, 2021. 3

[30] Shan-Jean Wu, Chih-Yuan Yang, and Jane Yung-jen Hsu. Calligan: Style and structure-aware chinese calligraphy character generator. *arXiv preprint arXiv:2005.12500*, 2020. 2

[31] Yangchen Xie, Xinyuan Chen, Li Sun, and Yue Lu. Dgfont: Deformable generative networks for unsupervised font generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5130–5140, 2021. 1, 2, 3, 6, 7

[32] Xiaoming Yu, Yuanqi Chen, Thomas Li, Shan Liu, and Ge Li. Multi-mapping image-to-image translation via learning disentanglement. *arXiv preprint arXiv:1909.07877*, 2019. 2

[33] Yexun Zhang, Ya Zhang, and Wenbin Cai. Separating style and content for generalized style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8447–8455, 2018. 3

[34] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei 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. 2