# Few-shot Font Generation with Localized Style Representations and Factorization

Song Park,[1, *] Sanghyuk Chun,[2, *] Junbum Cha,[3] Bado Lee,[3] Hyunjung Shim[1, †]
[1] School of Integrated Technology, Yonsei University
[2] NAVER AI LAB
[3] NAVER CLOVA
song.park@yonsei.ac.kr, sanghyuk.c@navercorp.com, junbum.cha@navercorp.com, bado.lee@navercorp.com,
kateshim@yonsei.ac.kr

## Abstract

Automatic few-shot font generation is a practical and widely studied problem because manual designs are expensive and sensitive to the expertise of designers. Existing few-shot font generation methods aim to learn to disentangle the style and content element from a few reference glyphs, and mainly focus on a universal style representation for each font style. However, such approach limits the model in representing diverse local styles, and thus makes it unsuitable to the most complicated letter system, *e.g.*, Chinese, whose characters consist of a varying number of components (often called “radical”) with a highly complex structure. In this paper, we propose a novel font generation method by learning localized styles, namely component-wise style representations, instead of universal styles. The proposed style representations enable us to synthesize complex local details in text designs. However, learning component-wise styles solely from reference glyphs is infeasible in the few-shot font generation scenario, when a target script has a large number of components, *e.g.*, over 200 for Chinese. To reduce the number of reference glyphs, we simplify component-wise styles by a product of component factor and style factor, inspired by low-rank matrix factorization. Thanks to the combination of strong representation and a compact factorization strategy, our method shows remarkably better few-shot font generation results (with only 8 reference glyph images) than other state-of-the-arts, without utilizing strong locality supervision, *e.g.*, location of each component, skeleton, or strokes. The source code is available at https://github.com/clovaai/lffont.

## Introduction

Text is a critical resource taking a considerable portion of the information on the web. Thus, text design is essential to improve the quality of services and user experiences. However, font design is labor-intensive and heavily depends on the expertise of designers, especially for glyph-rich scripts such as Chinese. For this reason, various font generation methods have been investigated to address a few-shot font generation problem, which uses only a few reference font images for automatically generating all the glyphs.

In this paper, we tackle a few-shot font generation problem; generating a new font library with very few references, *e.g.*, 8. Without additional training procedure (*e.g.*, finetune the model on the reference characters), our goal is to generate high quality, diverse styles in the few-shot font generation scenario. Our few-shot generation scenario consists of training and generation stages. During model training, we rely on many paired data which is easily accessible by public font libraries. On the other hand, at the generation stage, we use only few-shot examples as unseen style references without additional model finetuning. This scenario is particularly effective when the target style glyphs are expensive to collect, *e.g.*, historical handwriting, but we have a large database for existing fonts, or computing resources are limited to run additional finetuning, *e.g.*, on mobile devices. A popular strategy to tackle the same problem is to separate style and content representations from the given glyph images (Sun et al. 2018; Zhang, Zhang, and Cai 2018; Gao et al. 2019; Srivatsan et al. 2019). These methods generate a full font library by combining the target style representation and the source content representations.

However, previous few-shot font generation methods learn a *universal* style representation for each style, limited in representing diverse local styles. It is particularly problematic when generating fonts for glyph-rich scripts, *e.g.*, Chinese, Korean, and Thai. Every Chinese character consists of a varying number of components (often called “radical”) with a highly complex structure. This property induces the visual quality of a Chinese character to be highly sensitive to local damage or a distinctive local component-wise style.

The same issue was also pointed out by Cha et al. (2020a,b) in that many previous methods often fail to transfer unseen styles for the few-shot Korean and Thai generation. To alleviate this problem, Cha et al. (2020a) propose dual-memory architecture, named DM-Font. DM-font extracts component-wise local features for all components at once, and then save them into two types of memory. Despite its notable generation quality, DM-Font is restricted to *complete compositional scripts*, such as Korean and Thai (Cha et al. 2020a,b). While each Korean or Thai character can be decomposed into the fixed number of components and positions, more complex script like Chinese can be decomposed into varying components and positions. As a result, DM-Font fails to disentangle complex glyph structures and

[*] Equal contribution. Work done at NAVER CLOVA.

[†] Hyunjung Shim is a corresponding author.

diverse local styles in the Chinese generation task, as shown in our experiments. Furthermore, DM-Font requires that all components are shown in the reference set at least once to construct their memories. These drawbacks make DM-Font not applicable to generate Chinese characters, consisting of hundreds of components, with a few references.

In this paper, we propose a novel few-shot font generation with localized style representations and factorization (LF-Font). LF-Font learn to disentangle complex glyph structures and *localized* style representations, instead of *universal* style representations. Owing to powerful representations, LF-Font can capture local details in rich text design, thus successfully handle Chinese compositionality. Our disentanglement strategy preserves highly complex glyph structures, while DM-Font (Cha et al. 2020a) frequently loses the content information of complex Chinese characters. Consequently, our method shows remarkably better stylization performance than *universal* style encoding methods (Sun et al. 2018; Zhang, Zhang, and Cai 2018; Gao et al. 2019).

We define the localized style representation as a character-wise style feature which considers both a complex character structure and local styles. Instead of handling the large amount of characters in the glyph-rich script, we denote *the localized style representation* as a combination of component-wise local style representations. However, this strategy can have an inherent limitation; the reference set must cover the whole component set to construct the complete font library. It is infeasible when a target script has a large number of components, *e.g.*, over 200 for Chinese. To solve this issue, we introduce *factorization modules*, which factorizes a localized style feature to a component factor and a style factor. Consequently, our method can generate the whole vocabulary without having the entire components in the reference style, or utilizing strong locality supervision, *e.g.*, location of each component, skeleton, or strokes.

We demonstrate the effectiveness of the proposed LF-Font on the Chinese few-shot font generation scenario when the number of references is extremely small (namely, 8). Our method outperforms five state-of-the-art few-shot font generation methods with various evaluation metrics, with a significant gap. Careful ablation studies on our design choice shows that the proposed localized style representation and factorization module are an effective choice to tackle our target problem successively.

## Related Works

**Font generation as image-to-image (I2I) translation.** I2I translation (Isola et al. 2017; Zhu et al. 2017) aims to learn a mapping between source and target domains while preserving the contents in the source domain, *e.g.*, day to night. Recent I2I translation methods are extended to learn a mapping between multiple diverse domains (Choi et al. 2018; Liu et al. 2018; Yu et al. 2019; Choi et al. 2020), *i.e.*, multi-domain translation, thus can be naturally adopted into the font generation problem. For example, Tian (2017) attempted to solve the font generation task via paired I2I translation by mapping a fixed "source" font to the target font.

**Few-shot font generation.** The few-shot font generation task aims to generate new glyphs with very few numbers of style references without additional finetuning. The mainstream of few-shot font generation attempts to disentangle content and style representations as style transfer methods (Gatys, Ecker, and Bethge 2016; Huang and Belongie 2017; Li et al. 2017; Luan et al. 2017; Li et al. 2018; Yoo et al. 2019), but specialized to font generation tasks. For example, AGIS-Net (Gao et al. 2019) proposes the font-specialized local texture discriminator and the local texture refinement loss. Unlike other methods, DM-Font (Cha et al. 2020a) disassembles glyphs to stylized components and reassembles them to new glyphs by utilizing strong compositionality prior, rather than disentangles content and style.

Despite notable improvement over past years, previous few-shot font generation methods have significant drawbacks, such as infeasible to generate complex glyph-rich scripts (Azadi et al. 2018), failing to capture the local diverse styles (Sun et al. 2018; Zhang, Zhang, and Cai 2018; Gao et al. 2019; Liu et al. 2019; Srivatsan et al. 2019), or losing the complex content structures (Cha et al. 2020a,b). This paper proposes a novel few-shot font generation method that disentangles complex local glyph structure and diverse local styles, resulting in high visual quality of the generated samples for complex glyph-rich scripts, *e.g.*, Chinese.

**Other Chinese font generation methods.** Although we only focus on the few-shot font generation problem, there are several papers address the Chinese font generation task with numerous references or additional finetuning. SC-Font (Jiang et al. 2019) and ChiroGAN (Gao and Wu 2020) extract a skeleton or a stroke from the source glyphs and translate it to the target style. They require a large number of references for generating glyphs with a new style, *e.g.*, 775 (Jiang et al. 2019). Instead of expensive skeleton or stroke annotations, another approach (Sun et al. 2018; Huang et al. 2020; Wu, Yang, and Hsu 2020; Cha et al. 2020a) utilizes the compositionality to reduce the expensive search space in the character space to smaller component space. RD-GAN (Huang et al. 2020) aims to generate unseen characters in the fixed style, not feasible to our few-shot font generation scenario. CalliGAN (Wu, Yang, and Hsu 2020) encodes the styles by one-hot vectors; thus it requires additional finetuning for making unseen style during the training. ChiroGAN (Gao and Wu 2020) aims to solve unpaired font generation tasks as unpaired image-to-image translation tasks (Zhu et al. 2017). However, in our scenario, glyph images can be easily rendered from an existing font library, building a paired training dataset is cheap and does not limit the practical usage.

## Few-shot Font Generation with Localized Style Representations and Factorization

We propose a novel few-shot font generation framework, few-shot font generation with localized style representations and factorization (LF-Font), having strong representational power even with a very few reference glyphs, by introducing localized style features and factorize modules.

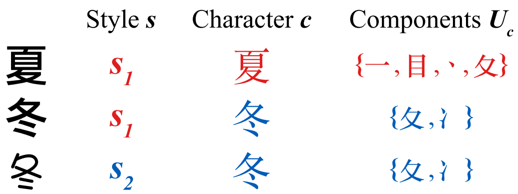

Figure 1: The character label $c$, the style label $s \in \{s_1, s_2\}$ and the component label set $U_c$ is shown.

## Problem Definition

We define three annotations for a glyph image $x$: the style label $s \in \mathcal{S}$, the character label $c \in \mathcal{C}$, and the component labels $U_c = [u_1^c, \ldots, u_m^c]$, where $m$ is the number of components in character $c$. Here, each character $c$ can be decomposed into components $U_c$ by the pre-defined decomposition rule as Figure 1. In our Chinese experiments, the number of the styles $|\mathcal{S}| = 482$, the number of the characters $|\mathcal{C}| = 19,514$, and the number of the components $|U| = 371$. In other words, all $19,514$ characters can be represented by the combination of 371 components. Note that our problem definition is not limited to the Chinese, but easily extended to other languages as shown in Appendix.

The goal of few-shot font generation task is to generate a glyph $x_{\widetilde{s},c}$ with unseen target style $\widetilde{s}$ for all $c \in \mathcal{C}$ with very few number of references $x_{\widetilde{s},\widetilde{c}} \in \mathcal{X}_r$, *e.g.*, $|\mathcal{X}_r| = 8$. A common framework for few-shot font generation is to learn a generator $G$, which takes the style representation $f_{\widetilde{s}} \in \mathbb{R}^d$ from $\mathcal{X}_r$ and the content representation $f_c \in \mathbb{R}^d$ as inputs, and synthesize a glpyh $x$ having reference styles $\widetilde{s}$ but representing a source character $c$. It is formulated by developing the generator $G$ and encoders $E_s$ and $E_c$ for extracting style and content representations, respectively as follows:

$$\begin{aligned} x_{\widetilde{s},c} &= G(f_{\widetilde{s}}, f_c), \\ f_{\widetilde{s}} = E_s(\mathcal{X}_r) &\text{ and } f_c = E_c(x_{s_0,c}), \end{aligned} \tag{1}$$

where $s_0$ is the source style label.

## Localized Style Representations

Previous methods assume that the style representation $f_s$ is universal for each style $s$, uniquely determined over all characters. However, the universal style assumption can overlook complex local styles, resulting in poor performances for unseen styles, as pointed by (Cha et al. 2020a). Here, we design the style encoder $E_s$ to encode character-wise style. This strategy is useful when a style is defined very locally and diversely as Chinese characters. However, the huge vocabulary size of Chinese script ($|\mathcal{C}| > 20,000$) makes it impossible to exploit all character-wise styles.

Instead of handling all character-wise styles, we first represent the character as a combination of multiple components, and develop the component-wise styles to minimize the redundancy in character-level representations. For that, we utilize the component set $U_c$ instead of the character label $c$, where $|U| \ll |\mathcal{C}|$. We extract a component-wise style feature $f_{s,u}(x,u) = E_{s,u}(x,u) \in \mathbb{R}^d$ from a reference glyph image $x$ and a component label $u \in U_c$ by introducing a component-wise style encoder $E_{s,u}$. Then, we compute *the character-aware localized style feature* $f_{s,c}$ by taking the summation over component-wise features $f_{s,u}$. Now, we can rewrite Eq (1) with the proposed character-aware localized style features as follows:

$$\begin{aligned} x(\widetilde{s}, c) = G(f_{\widetilde{s},c}, f_c), &\quad f_c = E_c(x_{s_0,c}), \\ f_{\widetilde{s},c} = \sum_{u \in U_c} f_{\widetilde{s},u} &= \sum_{u \in U_c} E_{s,u}(x_{\widetilde{s},\widetilde{c}_u}, u), \end{aligned} \tag{2}$$

where $x_{\widetilde{s},\widetilde{c}_u}$ is a glyph image from reference set $\mathcal{X}_r$ whose character is $\widetilde{c}_u$, which contains component $u$. However, the minimum required size of $\mathcal{X}_r$ is too large for Chinese because a total number of component set $\mathcal{U}$ in Chinese is still too large, *e.g.*, 229.

## Completing Missing Localized Style Representations by Factorization Modules

In our scenario, only partial components are observable from the reference set, while the other components are not accessible by $E_{s,u}$. Hence, the localized style feature $f_{s,c}$ for a style $s$ and a character $c$ with unseen components cannot be computed, and $G$ therefore, cannot generate a glyph with $c$. In other words, the few-shot font generation is not achievable if the reference glyphs cannot cover the whole component set ($|U| = 371$). To tackle the problem, we formulate the few-shot font generation problem as a reconstruction problem; given observations with a few style-component pairs, we aim to reconstruct the missing style-component pairs. Inspired from classical matrix completion approaches (Candès and Recht 2009; Cai, Candès, and Shen 2010), we decompose the component-wise style feature $f_{s,u} \in \mathbb{R}^d$ into two factors: a component factor $z_u \in \mathbb{R}^{k \times d}$ and a style factor $z_s \in \mathbb{R}^{k \times d}$, where $k$ is the dimension of factors. Formally, we decompose $f_{s,u}$ into $z_s$ and $z_u$ as follows:

$$f_{s,u} = \mathbf{1}^\top (z_s \odot z_u), \tag{3}$$

where $\odot$ is an element-wise matrix multiplication, and $\mathbf{1} \in \mathbb{R}^k$ is an all-ones vector. Eq (3) can be interpreted as the element-wise matrix factorization of $f_{s,u}$. In practice, we extract the style factor $z_s$ from the reference set and combine them with the component factor $z_u$ from the source glyph to reconstruct a component-wise style feature $f_{s,u}$ for the given source character $c$. Note that Tenenbaum and Freeman (2000); Srivatsan et al. (2019) also use a factorization strategy to font generation, but they directly apply the factorization to the complex glyph space, *i.e.*, each element is an image, while LF-Font factorizes the localized style features into the style and the content factors.

Traditional matrix completion methods require heavy computations and memory consumption. For example, expensive convex optimization (Candès and Recht 2009) or alternative algorithm (Cai, Candès, and Shen 2010), are infeasible in our scenario: repeatedly apply matrix factorization $d$ times to obtain a $d$ dimensional feature $f_{s,u}$. Instead, we propose an style and component factorization modules $F_s$

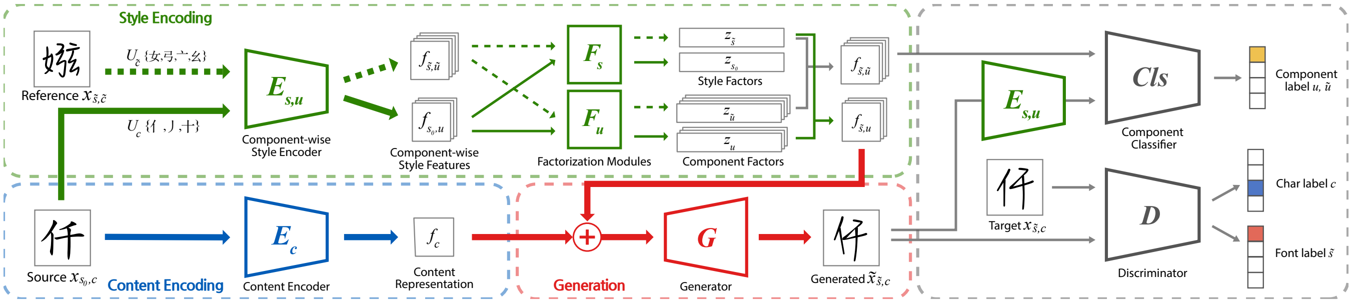

Figure 2: LF-Font consists of four parts; the content-encoding $E_c$, the style-encoding $E_{s,u}, F_s, F_u$, the generation $G$, and the shared modules $D, Cls$ for training. $E_c$ encodes the source glyph to the content representation $f_c$. In our style encoding stage, the source image (solid line) and reference images (dashed line) are encoded to component-wise style features $f_{s,u}$, and further factorized into style and component factors $z_s, z_u$. The extracted style and component factors are combined to the character-wise style representation $f_{s,c}$ of the target glyph. The generator $G$ synthesizes the target glyph from the content feature $f_c$ and the localized style feature $f_{s,c}$. The discriminator and the component classifier are employed for training.

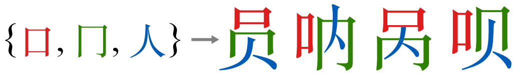

Figure 3: Examples to show that a component set is mapped to diverse characters.

and $F_u$ which extracts factors $z_s, z_u \in \mathbb{R}^{k \times d}$ from the given feature $f_{s,u} \in \mathbb{R}^d$ as follows:

$$z_s = F_s(f_{s,u}; W, b), \quad z_u = F_u(f_{s,u}; W, b). \tag{4}$$

We use a linear weight $W = [w_1; \dots; w_k] \in \mathbb{R}^{k \times d}$ and a bias $b \in \mathbb{R}^k$ as a factorization module, where each factor is computed by $z = [w_1 \odot f_{s,u} + b_1; \dots; w_k \odot f_{s,u} + b_k]$.

Note that solely employing the factorization modules, *i.e.*, Eq (4), does not guarantee that factors with the same style (or component) from different glyphs have identical values. Thus, we train the factorization modules $F_s$ and $F_u$ by minimizing the consistency loss $\mathcal{L}_{consist}$ as follows:

$$\mathcal{L}_{consist} = \sum_{s \in \mathcal{S}} \sum_{u \in \mathcal{U}} \|F_s(f_{s,u}) - \mu_s\|_2^2 + \|F_u(f_{s,u}) - \mu_u\|_2^2,$$

$$\mu_s = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} F_s(f_{s,u}), \quad \mu_u = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} F_u(f_{s,u}). \tag{5}$$

After training $F$, we can extract $z_s$ from even a random single reference glyph. Furthermore, by combining $z_s$ with the content factor $z_u$ from the known source glyph, we can reconstruct the localized style feature $f_{s,c} = \sum_{u \in U_c} f_{s,u}$ even for the unseen component $u$ in the reference set.

## Generation

Once LF-Font is trained with many paired training samples, it is able to generate any unseen style fonts with only a few references by extracting the style factor $z_{\widetilde{s}}$ from the reference glyphs, and by extracting $z_u$ and $f_c$ from the known source glyphs. Then, we combine $z_c$ and $z_{\widetilde{s}}$ for generating the localized style feature $f_{\widetilde{s},u}$ as described in the latter section. Finally, we generate a glyph $x$ using Eq (2). Formally, LF-Font consists of three sub-modules as illustrated in Figure 2. We describe the details of each sub-module below.

**Style encoding.** LF-Font encodes the localized style representation $f_{s,c}$ by encoding the component-wise features $f_{s,u}$ as formulated in Eqs (2), (3) and (4). There are three main modules in this stage: the component-wise style encoder $E_{s,u}$, the style and content factorization modules $F_s$ and $F_c$. $E_{s,u}$ is simply defined by a conditional encoder where a component label $u$ is used for the condition label, and encodes a glyph image $x$ into several component-wise style features $f_{s,u}$.

A component-wise style feature $f_{s,u}$ is factorized into the style factor $z_s$ and the component factor $z_u$ with factorization modules $F_s$ and $F_u$, respectively. We combine the style factor $z_{\widetilde{s}}$ from the reference glyphs and the component factor $z_u$ from the source glyph to reconstruct re-stylized the component-wise feature $f_{\widetilde{s},u}$. If there are more than one reference sample, we take the average over the style factors, extracted from each reference glyph, to compute $z_{\widetilde{s}}$.

**Content encoding.** Although our proposed style encoding strategy effectively captures the local component information, it requires guidance on the complex global structure (*e.g.*, relative locations of components) of each character, because a component set can be mapped to many characters – See Figure 3. We employ the content encoder $E_c$ to capture the complex global structural information of the source glyph. It facilitates to generate the target glyph while preserving complex structural information without any strong localization supervision of the source glyph.

**Generation.** Finally, the generator $G$ produces the target glyph $\widetilde{x}_{\widetilde{s},c}$ by combining the localized style representations $f_{\widetilde{s},c}$ from the style encoding and the global complex structural representation $f_c$ from the encoding.

## Training

Given the source glyph $x$ and the references $\mathcal{X}_r$ having the target style $s$, LF-Font learns the style encoder $E_{s,u}$, the content encoder $E_c$, the factorization modules $F_s, F_u$, and the generator $G$ for generating a glyph $\widetilde{x}$. We fix the source style $s_0$ during training and optimize the model parameters with diverse reference styles using the following losses.

**Adversarial loss.** As we strive to generate a plausible glyph in terms of both style and content, we employ a multi-head conditional discriminator for style label $s$ and character label $c$. The hinge GAN loss (Zhang et al. 2019) is used.

$$\begin{aligned}\mathcal{L}^D_{adv} =& -\mathbb{E}_{(x,s,c)\sim p_{data}} \max\left(0, -1 + D_{s,c}(x)\right) \\ & -\mathbb{E}_{(\widetilde{x},s,c)\sim p_{gen}} \max\left(0, -1 - D_{s,c}(\widetilde{x})\right) \\ \mathcal{L}^G_{adv} =& -\mathbb{E}_{(\widetilde{x},s,c)\sim p_{gen}} D_{s,c}(\widetilde{x}).\end{aligned} \tag{6}$$

**L1 loss and feature matching loss.** These objectives enforce the generated glyph $\widetilde{x}$ to reconstruct the ground truth glyph $x$ in pixel-level and feature-level.

$$\begin{aligned}\mathcal{L}_{l1} &= \mathbb{E}_{(x,s,c)\sim p_{data}}\left[\|x - \widetilde{x}\|_1\right], \\ \mathcal{L}_{feat} &= \mathbb{E}_{(x,s,c)\sim p_{data}}\left[\sum_{l=1}^{L} \|D_f^{(l)}(x) - D_f^{(l)}(\widetilde{x})\|_1\right]\end{aligned} \tag{7}$$

where $L$ is the number of layers in the discriminator $D$ and $D_f^{(l)}(x)$ is the intermediate feature in the $l$-th layer of $D$.

**Component-classification loss.** We employ additional component-wise classifier $Cls$ which classifies the component label $u$ of the given component-wise style feature $f_{s,u}$. We optimize the cross entropy loss (CE) as follows:

$$\mathcal{L}_{cls} = \sum_{\widetilde{u}\in U_{\widetilde{c}}} \mathrm{CE}(Cls(f_{s,\widetilde{u}}), \widetilde{u}) + \sum_{u\in U_c} \mathrm{CE}(Cls(f_{s,u}), u), \tag{8}$$

where $f_{s,\widetilde{u}}$ and $f_{s,u}$ are extracted from the reference glyph $x_{s,\widetilde{c}}$, and the generated glyph $\widetilde{x}_{s,c}$.

**Full objective.** Finally, we optimize LF-Font by the following full objective function:

$$\begin{aligned}\min_{\substack{E_c,E_{s,u},G,\\ F_s,F_u,Cls}} \max_{D}\ & \mathcal{L}_{adv(font)} + \mathcal{L}_{adv(char)} + \lambda_{L1}\mathcal{L}_{L1} \\ & + \lambda_{feat}\mathcal{L}_{feat} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{consist}\mathcal{L}_{consist},\end{aligned} \tag{9}$$

where $\lambda_{L1}, \lambda_{feat}, \lambda_{cls}, \lambda_{rep}$ are hyperparameters for controlling the effect of each objective. We set $\lambda_{L1} = 1.0$ and $\lambda_{feat} = \lambda_{cls} = \lambda_{rep} = 0.1$ throughout all the experiments.

**Training details.** We optimize our model with Adam optimizer (Kingma and Ba 2015). For stable training, we first train the model without factorization modules as Eq (2). Here, the model is trained to generate a target glyph from the component-wise style features $f_{s,u}$ directly extracted from the reference set $\mathcal{X}_r$. We construct a mini-batch with pairs of a reference set and a target glyph. To build each pair, we randomly select a style from the training style set and construct a reference set and a target glyph, where the components of the target glyph belong to the components in the reference set, but the target glyph is not in $\mathcal{X}_r$. After enough iterations, we add the factorization modules to the model and jointly train all modules. In this phase, the reference set is changed to have diverse styles and the target style is randomly chosen from the reference styles. More details of our method are described in Appendix.

# Experiments

This section shows the comparison results of LF-Font and previous methods in the Chinese few-shot font generation (Korean generation results are also shown in Appendix). Extensive analysis shows that our design choice successfully deals with the few-shot font generation task. We also provide ablation studies on the effects of objective functions, size of the reference set, and factor size $k$ in Appendix.

## Datasets and Evaluation Metrics

We collect public $482$ Chinese fonts from the web. The dataset has a total of $19,514$ characters (each font has a varying number of characters and it is $6,654$ characters on average), which can be decomposed by $371$ components. We sample $467$ fonts corresponding to $19,234$ characters for training, and the remaining unseen $15$ fonts are used for the evaluation. The models are separately evaluated with $2,615$ *seen characters* and $280$ *unseen characters* to measure the generalizability to the unseen characters.

We evaluate the visual quality of generated glyphs using various metrics. To measure how faithful the generated glyphs match their ground truths, LPIPS (Zhang et al. 2018) with ImageNet pre-trained VGG-16 is used. LPIPS is popularly used for assessing the similarity between two images by considering the perceptual similarity.

| | Localized<br>style? | Contents<br>encoder? | Restricted<br>to generate |
|---|---|---|---|
| SA-VAE | ✘ | ✘ | unseen chars (train) |
| EMD | ✘ | ✔ | |
| AGIS-Net | ✘ | ✔ | |
| FUNIT | ✘ | ✔ | |
| DM-Font | ✔ | ✘ | unseen components (refs.) |
| Ours | ✔ | ✔ | |

Table 1: We show the taxonomy of few-shot font generation by the localized style and the content encoder. Note that SA-VAE cannot generate unseen characters during the training, and DM-Font is unable to synthesis a glyph whose component is not observable in the reference glyphs.

| | | LPIPS ↓ | Acc (S) ↑ | Acc (C) ↑ | Acc (Hmean) ↑ | FID (S) ↓ | FID (C) ↓ | FID (Hmean) ↓ |
|---|---|---|---|---|---|---|---|---|
| Seen chars | SA-VAE (IJCAI'18) | 0.310 | 0.2 | 41.0 | 0.3 | 231.8 | 66.7 | 103.6 |
| | EMD (CVPR'18) | 0.248 | 11.9 | 63.7 | 20.1 | 148.1 | 25.7 | 43.8 |
| | AGIS-Net (TOG'19) | 0.182 | 34.0 | **99.8** | 50.7 | 79.8 | 4.0 | 7.7 |
| | FUNIT (ICCV'19) | 0.217 | 39.0 | 97.1 | 55.7 | 58.5 | 3.6 | 6.8 |
| | DM-Font (ECCV'20) | 0.275 | 10.2 | 72.4 | 17.9 | 151.8 | 8.0 | 15.2 |
| | LF-Font (proposed) | **0.169** | **75.6** | 96.6 | **84.8** | **40.4** | **2.6** | **4.9** |
| Unseen chars | EMD (CVPR'18) | 0.250 | 11.6 | 64.0 | 19.7 | 151.7 | 41.4 | 65.0 |
| | AGIS-Net (TOG'19) | 0.189 | 33.3 | **99.7** | 49.9 | 85.4 | 10.0 | 18.0 |
| | FUNIT (ICCV'19) | 0.216 | 38.0 | 96.8 | 54.5 | 63.2 | 12.3 | 20.6 |
| | DM-Font (ECCV'20) | 0.284 | 11.1 | 53.0 | 18.4 | 153.4 | 26.5 | 45.2 |
| | LF-Font (proposed) | **0.169** | **72.8** | 97.1 | **83.2** | **44.5** | **8.7** | **14.6** |

Table 2: Performance comparison on few-shot font generation scenario. Six few-shot font generation methods are compared with eight reference glyphs. LPIPS shows a perceptual similarity between the ground truth and the generated glyphs. We also report accuracy and FID measured by style-aware (S) and content-aware (C) classifiers. The harmonic mean (Hmean) of style- and content-aware metrics shows the overall visual quality of the generated glyphs. All numbers are average of 50 runs with different reference glyphs.

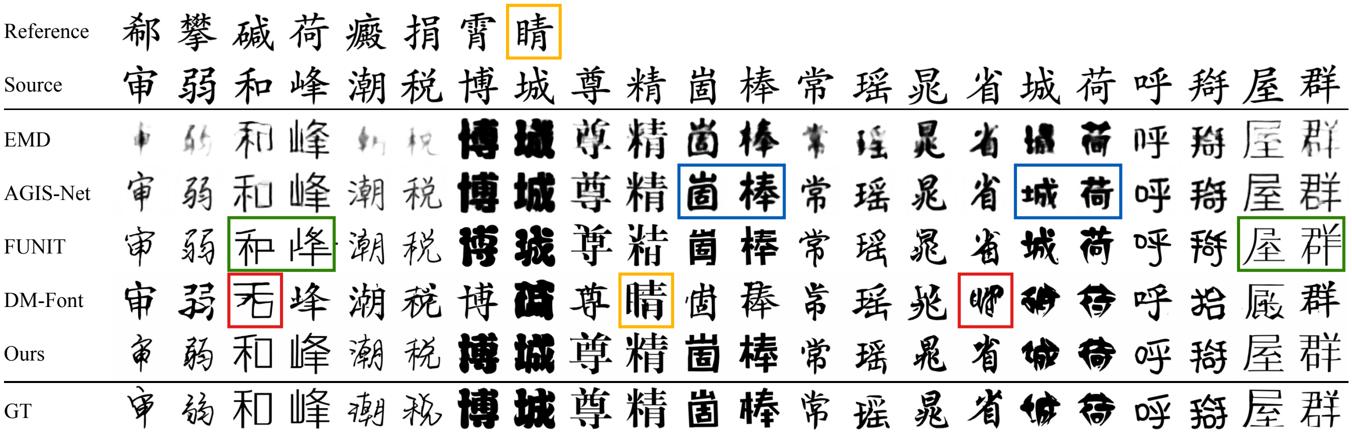

Figure 4: We show characters in the reference set (refer the character only, not style), source images, generated samples of LF-Font and five comparison methods, and the target glyphs (see GT). The reference images in each style are provided in Appendix. We also highlight samples which show the apparent limitation of each method by the colored boxes. Each color denotes the different failure cases discussed in the later section

We further assess the visual quality of generated glyphs in two aspects; content-preserving and style-adaptation aspects as Cha et al. (2020a). We train two classifiers, each to distinguish the style or content labels of the test dataset. Note that we train the evaluators independently from our generation models, and the character and font labels for the evaluation have no overlap with training labels. ResNet-50 (He et al. 2016) is employed for the backbone architecture. Comparing to photorealistic images, glyph images are highly sensitive to the local damage or a distinctive local component-wise information. We optimize evaluation classifiers by CutMix augmentation (Yun et al. 2019), which let a model learn localizable and robust features (Chun et al. 2019), and AdamP optimizer (Heo et al. 2020). More details are in Appendix. We report the accuracies of the generated glyphs by *style-aware* and *content-aware* models, respectively. We also use each classifier as a feature extractor and compute Frechét inception distance (FID) (Heusel et al. 2017). In the experiments, we denote metrics computed by content and style classifiers as *content-aware* and *style-aware*, respectively.

## Comparison Methods

We compare our model with five state-of-the-art few-shot font generation methods. For the sake of understanding the similarity or dissimilarity between methods, we categorize them by whether or not they explicitly model style representations or content representations as Table 1.

SA-VAE (Sun et al. 2018) extracts a universal style feature and utilizes a content code from the character classifier instead of the content encoder. This method cannot synthesize the characters unseen during training.

EMD (Zhang, Zhang, and Cai 2018), AGIS-Net (Gao et al. 2019), and FUNIT (Liu et al. 2019) employ the content encoder but their style representation is universal for the given style. For FUNIT, we use the modified FUNIT for the font task as Cha et al. (2020a,b). We empirically show that this universal style representation strategy fails to capture diverse styles, even incorporating specialized modifications, *e.g.*, the local texture discriminator, and the local texture refinement loss for AGIS-Net.

DM-Font (Cha et al. 2020a) would be the most direct competitor to LF-Font. Both DM-Font and LF-Font utilize the component-wise style features to capture the local details. However, DM-Font is restricted to generate a glyph whose component is not in the reference set because it uses the learned codebook for each component instead of the content encoder. Since DM-Font affords to generate neither Chinese characters nor glyphs with unseen components, we use the source style to extract local features for substituting the component-wise features for the unseen component. The modification details are described in Appendix.

## Experimental Results

**Quantitative evaluation.** We evaluate the visual quality of the generated images by six models with eight reference glyphs per style. To avoid randomness by the reference selection, we repeat the experiments 50 times with different reference characters. A font generation method is required to satisfy two contradictory task objectives: it should preserve contents and stylize well. As an extreme failure case, it performs an identity mapping, which will show the perfect content preserving score but it will show zero style transfer score. Hence, we report harmonic mean of content and style scores to probe whether a method can satisfy both objectives well. Table 2 shows that our method outperforms previous state-of-the-arts with significant gaps, *e.g.*, 28.7pp higher harmonic mean accuracy than FUNIT, and 3.4 lower harmonic mean FID than AGIS-Net for the unseen characters. Our method particularly outperforms other methods in style-aware benchmarks while the content-aware benchmarks are not much damaged. For example, FUNIT and AGIS-Net show comparable performance in content-aware benchmarks to LF-Font, but they show far lower performances than LF-Font in style-aware benchmarks. In other words, FUNIT and AGIS-Net only focus on content preserving, while fail in good stylization.

**Qualitative evaluation.** We also compare generated samples by the methods qualitatively in Figure 4. For the reference style, please to refer the font style in GT and Appendix. We observe that AGIS-Net often drops local details such as serif-ness, varying thickness (blue boxes). The green boxes show that FUNIT overly relies on the structure of source images. Thus, FUNIT tends to destroy the local structures in generated glyphs when the source and the target glyphs' overall structure differ a lot. We argue that the universal style representation strategy by AGIS-Net and FUNIT causes the problems. We further provide extensive analysis of the style representations in the latter section.

We observe that DM-Font frequently fails to generate correct characters. For example, as the red boxes, DM-Font often generates a glyph whose relative component locations are muddled. Another example is in the yellow boxes; DM-Font generates glyphs with the wrong component, observable in the references. We conjecture that the absence of the content encoder makes DM-Font suffer from the complex structures of glyphs. In the latter section, we show that the content encoder is critical to capture the complex structures.

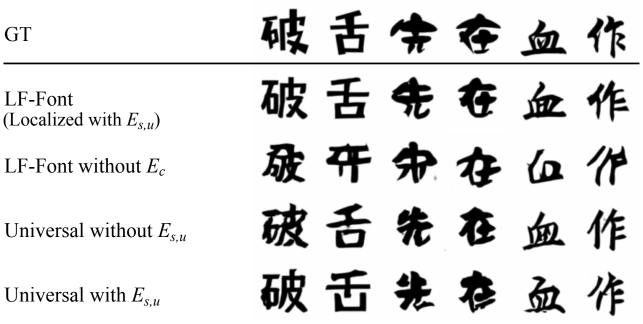
GT
LF-Font (Localized with $E_{s,u}$)
LF-Font without $E_c$
Universal without $E_{s,u}$
Universal with $E_{s,u}$

Figure 5: Visual samples of style and content module analysis. Visual samples in Table 4 and Table 3 are shown.

Compared to others, LF-Font generates the most plausible results that preserve the local details of each component and global structure of characters of target styles.

## Style and Content Module Analysis

In this subsection, we provide extensive analysis of our design choice for the style encoder and the content encoder.

**Localized style encoding.** We compare two universal-style encoding strategies to our localized style encoding strategy. First, we train a universal style encoder, which extracts a universal style from the references. EMD, AGIS-Net, and FUNIT employ this scheme. We also develop alternative universal-style encoding strategy with a component-wise style encoder $E_{s,u}$. This alternative encoding utilizes $E_{s,u}$ to extract component-wise features from references; however, the extracted features are directly used without considering the target character. On the other hand, our localized style encoder encodes the character-wise localized style representations using $E_{s,u}$ and factorization modules.

We conduct the ablation study to investigate the effects of different style encoding strategies and summarize the results in Table 3 (the same evaluation setting as Table 2). In Table 3, we observe that the universal style encoding without $E_{s,u}$ shows comparable style-aware performances (33.6%) to AGIS-Net (33.3%) or FUNIT (38.0%). We further confirm that universal styles by adding the component-wise style encoder $E_{s,u}$ is useful to increase the style-aware metric (33.6% $\rightarrow$ 52.8%), and our reorganized localized style representation improves the style-aware metric (33.6% $\rightarrow$ 72.8%). The generated samples for each ablation are shown in Figure 5. From these results, we conclude that the proposed localized style representation enables the model to capture diverse local styles, while the universal style encoding fails in fine and local styles.

| Style representation $f_s$ | Acc (S) ↑ | Acc (C) ↑ | Acc (Hmean) ↑ |
|---|---|---|---|
| AGIS-Net | 33.3 | **99.7** | 49.9 |
| FUNIT | 38.0 | 96.8 | 54.5 |
| Universal without $E_{s,u}$ | 33.6 | 97.2 | 49.9 |
| Universal with $E_{s,u}$ | 52.8 | 95.9 | 68.1 |
| Localized with $E_{s,u}$ | **72.8** | 97.1 | **83.2** |

Table 3: Three different style encoding strategies are evaluated. The universal style encoding without the component-wise style encoder $E_{s,u}$ is defined for each style. The universal style with $E_{s,u}$ is computed by the average of the reference component-wise styles. The bottom row shows Ours.

|  | Few-shot |  |  | Many-shot |  |  |
|---|---|---|---|---|---|---|
| Accuracies | S | C | H | S | C | H |
| DM-Font | 11.1 | 53.0 | 18.4 | 51.8 | 15.0 | 23.2 |
| LF-Font without $E_c$ | 36.3 | 15.4 | 21.7 | 37.8 | 5.1 | 8.9 |
| LF-Font | **72.8** | **97.1** | **83.2** | **74.7** | **96.5** | **84.2** |

Table 4: We evaluate DM-Font, LF-Font without content encoder $E_c$, and LF-Font, in the few-shot (8 references) and many-shot (256 references) scenarios. We report the style-aware accuracy (S), content-aware accuracy (C), and their harmonic mean (H). Note that DM-Font is similar to LF-Font without $E_c$, but the *persistent memory* is used.

**Content encoding.** Although the localized style encoding brings impressive improvements in transferring a target style, our localized style encoding strategy has a drawback; it will extract the same feature for characters whose components are identical, but the locations vary. To solve this problem, we employ the content encoder $E_c$ enforced to capture structural information. Here, we examine various content-encoding strategies: LF-Font without content-encoding, DM-Font (*persistent memory* for content encoding), and LF-Font. When developing LF-font without the content encoder $E_c$, the target glyph is generated with the localized style features alone. DM-Font replaces the content encoder with *persistent memory*, a learned codebook defined for each component. Note that DM-Font cannot generate unseen reference components; thus we replace unseen component features to the source style features. For removing unexpected effects from this source style replacement strategy, we reported many-shot (256 references) results in addition to few-shot (8 references) results.

In Table 4, we observe that the content encoder notably enhances overall performance (21.7% → 83.2% in few-shot harmonic mean accuracy). Since there is no content information, the style encoder of LF-Font without $E_c$ should encode both style and content information of each component. However, as the style encoder is optimized for modeling local characteristics, it is limited to handle global structures *e.g.*, the positional relationship of components. Besides, because a combination of a component set can be mapped to diverse characters as Figure 3, solely learning localized style features without global structures cannot reconstruct the correct character even though it can capture detailed local styles. Qualitative examples for LF-Font without the content encoder are in Figure 5.

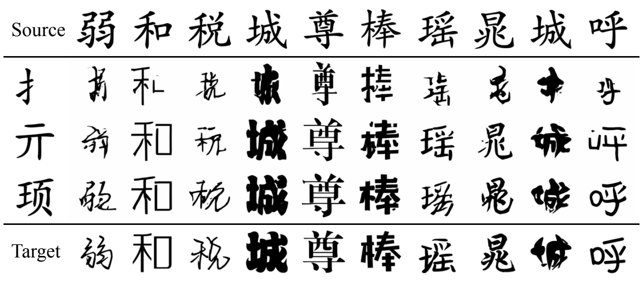

Figure 6: One-shot generation results. The reference characters and resultant images are visualized. The top and bottom rows show the source and target images, and the leftmost column shows the single reference used to generate the images in the same row.

Similar to the content encoder, the persistent memory strategy proposed by DM-Font, moderately improves the content performance (15.4% → 53.0%) but shows worse stylization due to the source style replacement strategy. Furthermore, both LF-Font without $E_c$ and DM-Font suffers from the content performance drop in the many-shot setting. This is because, their style encoders suffer from encoding the complex structures.

**One-shot generation.** We also visualize the extreme case, the one-shot generation by LF-Font, in Figure 6. We observe that when the reference glyph is too simple to extract solid component-wise features (the second row in Figure 6), the generated images show poor visual quality. Note that this might be the problem of style factors, not $E_c$, because the same content factors and content features (from $E_c$) are used for successfully generating other samples. Hence, we can conclude that the reference selection is critical to the visual quality and that the reference having rich local details is advantageous for high-quality generation.

## Conclusion

Our novel few-shot font generation method, named LF-Font, produces complex glyphs that preserve the local detail styles by introducing character-wisely defined style representations. Furthermore, we propose the factorization modules to reconstruct the entire character-wise style representations from a few reference images. It enables us to reorganize the seen character-wise style representations to the unseen character-wise style representations by disentangling character-wise style representations into style and component factors. In the experiments, LF-Font outperforms state-of-the-art few-shot font generation methods in various evaluation metrics, particularly in *style-aware* benchmarks. Our extensive analysis of our design choice supports that our framework effectively disentangles content and style representations, resulting in the high-quality generated samples with only a few references, *e.g.*, 8.

## Acknowledgements

We thank Junsuk Choe, Dongyoon Han, Chan Young Park for the feedback. NAVER Smart Machine Learning (NSML) (Kim et al. 2018) has been used for experiments. This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the MSIP (NRF-2019R1A2C2006123, 2020R1A4A1016619), and by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2020-0-01361, Artificial Intelligence Graduate School Program (YONSEI UNIVERSITY)). This work was also supported by the Korea Medical Device Development Fund grant funded by the Korea government (the Ministry of Science and ICT, the Ministry of Trade, Industry and Energy, the Ministry of Health & Welfare, Republic of Korea, the Ministry of Food and Drug Safety) (Project Number: 202011D06 )

## References

Azadi, S.; Fisher, M.; Kim, V. G.; Wang, Z.; Shechtman, E.; and Darrell, T. 2018. Multi-Content GAN for Few-Shot Font Style Transfer. In *CVPR*.

Cai, J.-F.; Candès, E. J.; and Shen, Z. 2010. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization* .

Candès, E. J.; and Recht, B. 2009. Exact matrix completion via convex optimization. *Foundations of Computational mathematics* .

Cha, J.; Chun, S.; Lee, G.; Lee, B.; Kim, S.; and Lee, H. 2020a. Few-shot Compositional Font Generation with Dual Memory. In *ECCV*.

Cha, J.; Chun, S.; Lee, G.; Lee, B.; Kim, S.; and Lee, H. 2020b. Toward High-quality Few-shot Font Generation with Dual Memory. *AI for Content Creation Workshop. CVPR Workshop* .

Choi, Y.; Choi, M.; Kim, M.; Ha, J.-W.; Kim, S.; and Choo, J. 2018. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*.

Choi, Y.; Uh, Y.; Yoo, J.; and Ha, J.-W. 2020. StarGAN v2: Diverse image synthesis for multiple domains. In *CVPR*.

Chun, S.; Oh, S. J.; Yun, S.; Han, D.; Choe, J.; and Yoo, Y. 2019. An Empirical Evaluation on Robustness and Uncertainty of Regularization Methods. *ICML Workshop on Uncertainty and Robustness in Deep Learning* .

Gao, Y.; Guo, Y.; Lian, Z.; Tang, Y.; and Xiao, J. 2019. Artistic Glyph Image Synthesis via One-Stage Few-Shot Learning. *ACM Transactions on Graphics* .

Gao, Y.; and Wu, J. 2020. GAN-Based Unpaired Chinese Character Image Translation via Skeleton Transformation and Stroke Rendering. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2016. Image Style Transfer Using Convolutional Neural Networks. In *CVPR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

Heo, B.; Chun, S.; Oh, S. J.; Han, D.; Yun, S.; Uh, Y.; and Ha, J.-W. 2020. Slowing Down the Weight Norm Increase in Momentum-based Optimizers. *arXiv preprint arXiv:2006.08217* .

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*.

Huang, X.; and Belongie, S. J. 2017. Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization. In *ICCV*.

Huang, Y.; He, M.; Jin, L.; and Wang, Y. 2020. RD-GAN: Few/Zero-Shot Chinese Character Style Transfer via Radical Decomposition and Rendering. In *ECCV*.

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *CVPR*.

Jiang, Y.; Lian, Z.; Tang, Y.; and Xiao, J. 2019. SCFont: Structure-Guided Chinese Font Generation via Deep Stacked Networks. In *AAAI Conference on Artificial Intelligence*.

Kim, H.; Kim, M.; Seo, D.; Kim, J.; Park, H.; Park, S.; Jo, H.; Kim, K.; Yang, Y.; Kim, Y.; et al. 2018. NSML: Meet the MLaaS platform with a real-world case study. *arXiv preprint arXiv:1810.09957* .

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Li, Y.; Fang, C.; Yang, J.; Wang, Z.; Lu, X.; and Yang, M.-H. 2017. Universal Style Transfer via Feature Transforms. In *Advances in Neural Information Processing Systems*.

Li, Y.; Liu, M.-Y.; Li, X.; Yang, M.-H.; and Kautz, J. 2018. A Closed-form Solution to Photorealistic Image Stylization. In *ECCV*.

Liu, A. H.; Liu, Y.-C.; Yeh, Y.-Y.; and Wang, Y.-C. F. 2018. A unified feature disentangler for multi-domain image translation and manipulation. In *Advances in neural information processing systems*.

Liu, M.-Y.; Huang, X.; Mallya, A.; Karras, T.; Aila, T.; Lehtinen, J.; and Kautz, J. 2019. Few-Shot Unsupervised Image-to-Image Translation. In *ICCV*.

Luan, F.; Paris, S.; Shechtman, E.; and Bala, K. 2017. Deep Photo Style Transfer. In *CVPR*.

Srivatsan, N.; Barron, J.; Klein, D.; and Berg-Kirkpatrick, T. 2019. A Deep Factorization of Style and Structure in Fonts. In *Conference on Empirical Methods in Natural Language Processing*.

Sun, D.; Ren, T.; Li, C.; Su, H.; and Zhu, J. 2018. Learning to Write Stylized Chinese Characters by Reading a Handful of Examples. In *International Joint Conference on Artificial Intelligence*.

Tenenbaum, J. B.; and Freeman, W. T. 2000. Separating style and content with bilinear models. *Neural computation* 12(6): 1247–1283.

Tian, Y. 2017. zi2zi: Master Chinese Calligraphy with Conditional Adversarial Networks. URL https://github.com/kaonashi-tyc/zi2zi.

Wu, S.-J.; Yang, C.-Y.; and Hsu, J. Y.-j. 2020. CalliGAN: Style and Structure-aware Chinese Calligraphy Character Generator. *AI for Content Creation Workshop. CVPR Workshop* .

Yoo, J.; Uh, Y.; Chun, S.; Kang, B.; and Ha, J.-W. 2019. Photorealistic Style Transfer via Wavelet Transforms. In *ICCV*.

Yu, X.; Chen, Y.; Liu, S.; Li, T.; and Li, G. 2019. Multi-mapping image-to-image translation via learning disentanglement. In *Advances in Neural Information Processing Systems*.

Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J.; and Yoo, Y. 2019. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. In *ICCV*.

Zhang, H.; Goodfellow, I.; Metaxas, D.; and Odena, A. 2019. Self-Attention Generative Adversarial Networks. In *ICML*.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*.

Zhang, Y.; Zhang, Y.; and Cai, W. 2018. Separating Style and Content for Generalized Style Transfer. In *CVPR*.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *ICCV*.