

Few-shot Font Generation with Weakly Supervised Localized Representations

Song Park*, Sanghyuk Chun*, Junbum Cha, Bado Lee, Hyunjung Shim

Abstract—Automatic few-shot font generation aims to solve a well-defined, real-world problem because manual font designs are expensive and sensitive to the expertise of designers. Existing methods learn to disentangle style and content elements by developing a universal style representation for each font style. However, this approach limits the model in representing diverse local styles, because it is unsuitable for complicated letter systems, for example, 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 that learns localized styles, namely component-wise style representations, instead of universal styles. The proposed style representations enable the synthesis of complex local details in text designs. However, learning component-wise styles solely from a few reference glyphs is infeasible when a target script has a large number of components, for example, over 200 for Chinese. To reduce the number of required reference glyphs, we represent component-wise styles by a product of component and style factors, inspired by low-rank matrix factorization. Owing to the combination of strong representation and a compact factorization strategy, our method shows remarkably better few-shot font generation results (with only eight reference glyphs) than other state-of-the-art methods. Moreover, strong locality supervision, for example, location of each component, skeleton, or strokes, was not utilized. The source code is available at <https://github.com/clovaai/lffont> and <https://github.com/clovaai/fewshot-font-generation>.

Index Terms—Few-shot font generation, font generation, few-shot generation, image-to-image translation, computer vision



1 INTRODUCTION

Text is a critical resource of information on the web and publications. Fonts are paints of text-based content design thus fonts have a significant impact on the overall user experience and satisfaction with text-based content; such as logo designs, handouts, magazines, movie posters, and web pages. For example, “Gotham” font families are widely adopted for materials that should be objective but convincing (*e.g.*, election handouts, or memorials). On the other hand, “Comic Sans” is used for humorous presentations.

However, font design is a labor-intensive and time-consuming work, requiring elaborate handwork by proficient experts, especially for glyph-rich scripts such as Korean and Chinese. For example, when creating a new Korean font library; designers must manually create every single character per font style, where there are a total of 11,172 possible Korean characters (or at least 2,350 widely used characters), while maintaining a coherent font style [1]. For this reason, various font generation methods [2], [3], [4], [5], [6] have been investigated to address an automatic font generation problem, which generates a font with a coherent style of the given reference glyph images; the number of reference glyphs varies with the application scenario.

In this study, we addressed a practical font generation scenario: a few-shot font generation problem, in glyph-rich language systems [5], [6], [7], [8], [9], [10], [11], [12],



Fig. 1. **Example font styles of the same character.** Font styles are defined locally by diverse characteristics, such as local strokes or size of components.

generating a new font library with notably few references (8, herein). No additional training procedure (*e.g.*, fine-tuning the model on the reference characters) was performed. We aimed to generate high-quality, diverse styles in the few-shot font generation scenario. The few-shot generation scenario consists of the training and generation stages. During model training, we rely on paired data that are easily accessible by public font libraries. In contrast, at the generation stage, we use only few-shot examples as unseen style references and require no additional model fine-tuning. This scenario is particularly effective in the following application scenarios: 1) when the target-style glyphs are expensive to collect (*e.g.*, historical handwriting), but there is a large database for existing fonts; or 2) computing resources are limited to run additional fine-tuning (*e.g.*, on mobile devices). A popular strategy to tackle this problem is to separate style and content representations from the given glyph images [5], [7], [8], [10]. These methods generate a full-font library by combining target-style representation and source-content representations.

A major challenge in font generation tasks is that the font style is often defined locally, *e.g.*, local strokes, serif-ness, or size of sub-characters as shown in Figure 1. Therefore, a few-shot font generation method should be capable of extracting complex local features from very few reference glyphs.

- Song Park, and Hyunjung Shim are with the School of Integrated Technology, Yonsei University, South Korea.
- Sanghyuk Chun is with NAVER AI Lab.
- Sanghyuk Chun, Junbum Cha, and Bado Lee are with the NAVER CLOVA
- Song Park and Sanghyuk Chun contributed equally to this work.
- Hyunjung Shim is a correspondence author (kateshim@yonsei.ac.kr).

However, previous few-shot font generation methods [5], [7], [8] learned and extracted a *universal* style representation for each style, which is limited in representing diverse local styles. This is particularly problematic when generating fonts for glyph-rich scripts, such as Chinese or Korean. For example, every Chinese character can be divided into a number of sub-characters or components, with their own meanings, as shown in Figure 2. Hence, the visual quality of a Chinese character tends to be highly sensitive to local damage or a distinctive local component-wise style. As a result, it is difficult to represent highly complex and diverse local styles using only a universal style representation.

Cha *et al.* [6] reported that many previous methods often fail to transfer unseen styles for the few-shot Korean generation. To alleviate this problem, they proposed a novel architecture named DM-Font. DM-Font extracts component-wise local features for all components and utilizes them for generation. Despite its notable generation quality, DM-Font is restricted to *complete compositional scripts*, such as Korean and Thai [6], [13]. While each Korean character can be decomposed into a fixed number of components and positions, more complex scripts (*e.g.*, Chinese) can be decomposed into varying components and positions. Consequently, we empirically observe that DM-Font fails to disentangle complex glyph structures and diverse local styles in the Chinese generation task. Furthermore, DM-Font requires that all components are shown in the reference set at least once to construct their memories. In our experiments, DM-Font tends to require more reference images than others; the fonts generated by DM-Font show worse visual quality than other methods using the same number of reference sets (see Figure 10). These drawbacks limit the applicability of DM-Font 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) that utilizes compositionality, a language-specific characteristic, and a weak supervised framework for few-shot font generation. We focus on compositional scripts whose character is decomposed into a number of sub-characters or components, as illustrated in Figure 2. Here, the component labels are weak supervision; the components in the given glyph are known but their locations are unknown. With the component labels of the given glyph as weak supervision, LF-Font learns 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 handling Chinese compositionality. We show that without pixel-level guidance of each component, the proposed method successfully deals with localized representations, resulting in generalizability to novel styles with only very few references compared to previous state-of-the-art font generation methods.

We define the localized style representation as a character-wise style feature that considers both a complex character structure and local styles. Because handling all characters in the glyph-rich script (*e.g.* > 50,000 for Chinese script) is infeasible, we denote the *localized style representation* as a combination of component-wise local style representations to reduce the number of required style features (*e.g.*,

Chinese script has a few hundred components) (§ 3.3). However, this strategy can have an inherent limitation: the reference set must cover the entire 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 a *factorization module*, which factorizes a localized style feature to a component factor and a style factor (§ 3.4). Consequently, our method can generate the whole vocabulary without having the entire components in the reference style, or utilizing strong locality supervision, for example, the location of each component or skeleton.

We demonstrate the effectiveness of the proposed LF-Font on the Chinese and Korean few-shot font generation scenarios when the number of references is extremely small (*i.e.*, 8) (§ 4). Our method significantly outperforms five state-of-the-art few-shot font generation methods with various evaluation metrics. Careful ablation studies on our design choice show that the proposed localized style representation and factorization modules are an effective choice to tackle our target problem successively.

This work is an extensive version of our AAAI 2021 [11]. Compared to the AAAI 2021 work, this paper includes the following additional contributions: (a) reformulation of the original localized features using a weakly supervised learning problem (§1, §3), and related discussions (§2); (b) additional analyses and ablation studies to demonstrate the effectiveness of the proposed weakly supervised localized style representations; (c) additional comparisons by varying the reference size from one-shot to a many-shot; (d) extension to few-shot Korean generation; (e) removing component conditions in test-time, resulting in showing superior few-shot generation performances on unseen languages (*e.g.*, Chinese to Korean generation); and (f) showing the effectiveness of few-shot font generation methods to the character recognition systems.

2 RELATED WORKS

Font generation as image-to-image translation and style transfer. Image-to-image (I2I) translation [14], [15] aims to learn a mapping between source and target domains while preserving the contents in the source domain, for example, day to night. Recent I2I translation methods are extended to learn a mapping between multiple diverse domains [16], [17], [18], [19] (*i.e.*, multi-domain translation); thus, they can be naturally adopted into the font generation problem. For example, [2] attempted to solve the font generation task via paired I2I translation by mapping a fixed “source” font to the target font. Inspired by this approach, several recent papers [2], [4], [20], [21], [22] addressed the font generation task by presuming a large set of references, *e.g.*, 775 [4], and additional finetuning for generating each font. Our scenario focuses on the few-shot font generation task which requires only a few reference glyphs without any finetuning.

Font generation as a content-style disentanglement. Our application scenario is also related to content-style disentanglement approaches, such as style transfer. Style transfer methods use a pre-trained model that can represent content and style representations separately by their purpose, *e.g.*, artistic styles such as texture or painting styles [23], [24], [25] or photorealistic styles such as diverse lighting or

materials [26], [27], [28]. However, style transfer methods are designed to capture the “style” as global artistic textures or photorealistic colors of the given images, while the font style is defined locally. For this reason, the naïve extension of existing style transfer approaches to the font generation task is not effective. The unique characteristics of the font domain motivate the design of a font domain-specific approach for few-shot font generation tasks.

Similarly, other content-style disentanglement approaches, such as image-to-image translation tasks, suffer from the same issue. For example, most of image-to-image translation methods to extract style information from the references [9], [19], [29], [30], [31] relies on AdaIN [24], designed for style transfer. However, as we observed in our experiments with FUNIT [9], a universal style extractor based on AdaIN cannot capture the complex local styles of glyphs. In this paper, we propose a component-wise localized style representation to mitigate the issue by utilizing component labels as a weak supervision.

Attribute-conditioned generation and font generation tasks. Attribute-conditioned (AC) generation methods aim to generate an image by given attribute conditions, such as “brown hair color”, “female” [32], [33], [34], [35]. Since a glyph is conditioned by many components, font generation tasks can be viewed as AC generation tasks. However, there are significant differences between AC generation tasks and font generation tasks. First, a glyph is uniquely defined for each font design, while an attribute-conditioned image can be mapped to various images by different viewpoint, background or lighting [32]. For example, a facial image can be mapped to other facial images while keeping identity by changing identity independent information such as viewpoint or background. On the other hand, if we change a component condition of the given glyph, the meaning of the glyph will be no longer preserved. Second, in the font domain, it is easy to obtain glyph images with the same content but different styles, while in general image domains, *e.g.*, facial images, it is impossible to collect all possible pairs for the given attributes. In this paper, we focus on font-specific domain knowledge such as compositionality to capture complex local styles of glyph images.

Few-shot font generation. The few-shot font generation task aims to generate new glyphs with very few style references without additional fine-tuning. The mainstream of few-shot font generation attempts to disentangle content and style representations, specialized in font generation tasks. For example, AGIS-Net [5] proposed a font-specialized local texture discriminator and local texture refinement loss. Unlike other methods, DM-Font [6] disassembles glyphs into stylized components and reassembles them into new glyphs by utilizing a strong compositionality prior. DG-Font [36] utilizes deformable convolution [37] to unsupervised font generation tasks, *i.e.*, assuming there is no paired glyphs across different styles. In our scenario, we aim to generate standard true type font libraries hence we assume that there exist a number of paired glyph images across different styles by rendering images from the existing font libraries.

Despite notable improvements over the past few years, previous few-shot font generation methods have significant drawbacks. They are 1) infeasible to generate complex

glyph-rich scripts [38]; 2) fail to capture the local diverse styles [5], [7], [8], [9], [10]; or 3) loss of complex content structures [6], [13]. Our method employs localized style features trained by weak component supervision to capture the local diverse styles. To prevent the generated glyphs from losing the complex content structure, we used a content encoder to preserve the content structure. As a result, the samples generated by our method show high visual quality for complex glyph-rich scripts, *e.g.*, Chinese.

Many-shot font generation methods. Although we only focused on the few-shot font generation problem, several papers have addressed the Chinese font generation task with numerous references or additional fine-tuning. SCFont [4] and ChiroGAN [20] extracted a skeleton or stroke from the source glyphs and translated it into the target style. They required a large number of references for generating glyphs with a new style using the I2I framework, *e.g.*, 775 [4]. Instead of expensive skeleton or stroke annotations, different approaches [6], [7], [21], [22] utilize the compositionality to reduce the expensive search space in the character space to smaller component space. For example, RD-GAN [21] employs an additional LSTM architecture [39] to capture the compositionality of the given glyph. However, RD-GAN is designed to reconstruct unseen characters in a fixed style; thus, it cannot be applied to our few-shot generation scenario, which aims to generate characters with unseen styles. CalliGAN [22] encodes the styles by one-hot vectors; thus, it requires additional fine-tuning to create an unseen style during the training. ChiroGAN [20] aims to solve unpaired font generation tasks as unpaired image-to-image translation tasks [15]. However, in our scenario, glyph images can be easily rendered from an existing font library, as building a paired training dataset is cheap and does not limit practical usage. Similarly, StrokeGAN [40] employs one-bit stroke encoding to capture the key mode information of Chinese characters on unpaired font generation tasks. Both StrokeGAN and our method utilize the compositionality (or stroke information), but StrokeGAN only uses the primal strokes while ours utilizes the component labels where strokes and components correspond to “character bytes” and “tokenized words” in natural language processing. We did not compare our method to many-shot font generation methods because their methods are not applicable to our scenario: there exists very few references (*e.g.*, 8 in our experiments) and no finetuning procedure is allowed.

Weakly supervised object recognition. In this study, we utilized weak component-level supervision to learn localized features. Our weak supervision is image-level multi-labels without pixel-level annotations, that is, the exact position of each component is unknown. A similar scenario is widely adopted in many weakly supervised vision recognition tasks; such as weakly supervised object localization [41], [42], weakly supervised object detection [43], or weakly supervised semantic segmentation [44], [45]. These weakly supervised vision recognition techniques have shown that with only image-level weak supervision, they can achieve a reasonable localization ability, for example, locating the target objects in the image. However, these techniques are specifically designed for object recognition tasks by introducing a task-specific module design and training strategy.

	Style s	Character c	Components U_c
夏	s_1	夏	{一, 日, 夂}
冬	s_1	冬	{夂, 冫}
冬	s_2	冬	{夂, 冫}
呵	s_2	呵	{口, 口, 一, 丿}

Fig. 2. **Annotation examples.** The character label c , style label $s \in \{s_1, s_2\}$, and the component label set U_c are shown.

{口, 冫, 人} → 员 呐 另 呗

Fig. 3. **Characters from the same component set.** Examples show that a component set is mapped to diverse characters.

Hence, it is difficult to apply their schemes directly to the few-shot font generation problem. For example, object localization techniques commonly suffer from performance bias; the estimated localization map captures only the most discriminative regions of the object. To resolve this bias, existing methods have been developed to expand the localization map. For this purpose, adversarial complementary learning (ACoL) [46] for weakly supervised object localization proposes a two-head architecture — where one head acts as an adversary, and attempts to erase the high-score region produced by the other. The attention-dropout layer (ADL) [47] for the same problem is proposed to erase the highly attended region by channel attention. Both techniques propose a task-specific module for expanding the localization maps and require a generic-purpose vision backbone; namely the ImageNet-pretrained network. Hence, despite the advances of previous methods in weakly supervised object recognition, a new methodology is required to solve a few-shot generation task for a specific visual domain, that is, the font domain.

3 FEW-SHOT FONT GENERATION WITH WEAKLY SUPERVISED LOCALIZED REPRESENTATIONS

We propose a novel few-shot font generation framework, few-shot font generation with localized style representations and factorization (LF-Font). In this section, we introduce component-wise localized features trained by language-specific weak component labels.

3.1 Compositionality: a language-specific image-level weak supervision

A major challenge in the font generation task for a glyph-rich script, such as Chinese (> 50K glyphs) or Korean (\approx 11K glyphs) is the large number of characters to generate. Many letter systems have a language-specific property, *compositionality*; a character can be decomposed into a number of sub-characters or components. It is worth noting that among the top 30 popular letter systems, 24 have compositionality, e.g., Chinese, Hindi, Arabic, Japanese, Korean, and Thai (— See Appendix for examples). Utilizing the predefined decomposition rule, all characters can be represented by only a small number of components, for example, 68 components for Korean and about a few hundred components (371 in

our experiments) for Chinese¹. In Figure 2, we illustrate the example component labels and other annotations of Chinese characters. As seen in the bottom example of this figure, some Chinese characters contain duplicated components 2 (“口” is duplicated in this example). We used all duplicated components as the input for the proposed method.

The component labels are weak supervision; while the components in the given glyph are known, their locations are unknown. Furthermore, a component label set can be mapped to multiple characters. Figure 3 indicates that by combining three components, four different characters can be rendered, where the detailed shape of the component depends on the component location. For example, “口” (red component in the figure) shows different width and height ratios, and variable sizes upon different locations. We designed our model to capture local component features and to correctly combine them to express the structure of the target character. In the next subsections, we describe how LF-Font can deal with both localized component-wise features and the global structure of the target character.

3.2 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 $U_c = [u_1^c, \dots, u_m^c]$, where m is the number of components in character c . Here, each character c can be decomposed into components U_c using the predefined decomposition rule, as shown in Figure 2. In our Chinese generation tasks; the number of styles $|\mathcal{S}| = 482$, the number of characters $|\mathcal{C}| = 19,514$, and the number of components $|\mathcal{U}| = 371$. In other words, all 19,514 characters can be represented by a combination of 371 components. Our problem definition is not limited to Chinese, but is easily extended to other languages, as shown in §4.6.

The goal of the few-shot font generation task is to generate a glyph $x_{\tilde{s},c}$ with unseen target styles \tilde{s} for all $c \in \mathcal{C}$ with very few references $x_{\tilde{s},\tilde{c}} \in \mathcal{X}_r$, 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 $f_{\tilde{s}} \in \mathbb{R}^d$ from \mathcal{X}_r and the content representation $f_c \in \mathbb{R}^d$ as inputs. Then synthesizing a glyph x having the reference styles \tilde{s} , but representing a source character c . Formally, a few-shot font generation task can be represented as follows:

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

where s_0 is the source style label.

3.3 Localized style representations

Previous methods assume that the style representation f_s is universal for each style s . However, the universal style assumption can overlook complex local styles, resulting in poor performance for unseen styles as pointed out by [6]. Here, we design the style encoder E_s to encode a character-wise style. This strategy is useful when a style is defined locally and diversely as Chinese characters. However, the

1. We use the character decomposition data from Wikimedia Commons for Chinese decomposition. https://commons.wikimedia.org/wiki/Commons:Chinese_characters_decomposition

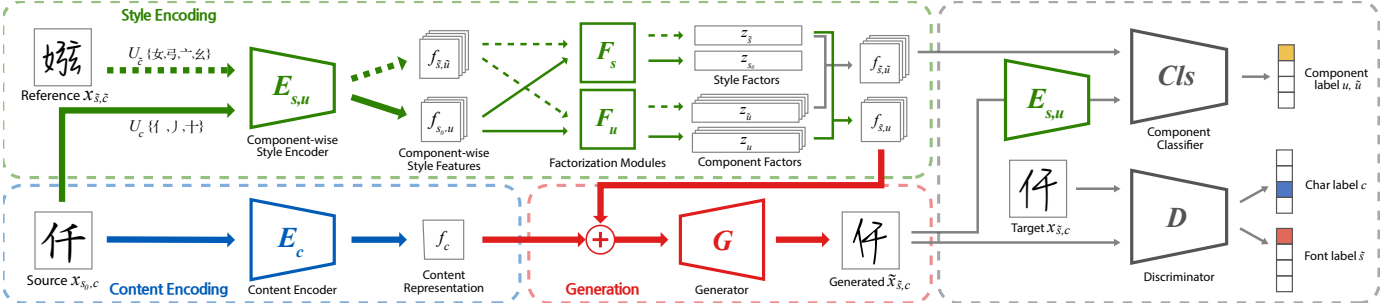


Fig. 4. **Overview of LF-Font.** LF-Font consists of four parts; the content-encoding E_c , the style-encoding $E_{s,u}, F_s, F_u$, the generation G , and the shared modules D, Cls for training. E_c encodes the source glyph to the content representation f_c . The source (solid line) and reference (dashed line) images are encoded to component-wise style features $f_{s,u}$, and further factorized into style and component factors z_s, z_u . z_s, z_u are combined to the character-wise style representation $f_{s,c}$ of the target glyph. The generator G synthesizes the target glyph from the f_c and $f_{s,c}$.

large 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 component-wise styles to minimize redundancy in character-level representations. For this, we utilize the component set U_c instead of the character label c , where $|U| \ll |\mathcal{C}|$ (371 and 19,514 in our experiments). 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 $E_{s,u}$. Here, the component labels are image-level weak supervision; we only know that there exists the given components but we do not know where are they. Our image-level component weak supervision is similar to the weak supervision adapted by weakly-supervised object localization (WSOL), weakly-supervised object detection (WSOD), or weakly-supervised semantic segmentation (WSSS). Our goal is to utilize weak image-level component labels to learn localized style representations without expensive pixel-level guidance of each component. From this motivation, 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:

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

where $x_{\tilde{s},\tilde{c}_u}$ is a glyph image from the reference set \mathcal{X}_r , whose character is \tilde{c}_u , which contains component u . However, this approach still has a significant drawback. Because we require the entire set of component-wise style representations ($|U| = 371$ for Chinese) to reconstruct the whole character set, the component labels of a reference set \mathcal{X}_r should cover all components. In the Chinese font generation scenario, at least 229 reference characters with a coherent style are required to derive complete component-wise style representations. In the following section, we introduce our solution to reduce the required number of reference images.

3.4 Completing missing localized style representations by factorization modules

Our localized style representation is defined by 1) different styles and 2) a character-wise manner, *i.e.*, a localized style representation $f_{s,c}$ is defined by a character c and a style s . As we defined in Eq (1), when synthesizing a new character with style s and character c' , we need a feature $f_{s,c'}$. Since during test-time, we do not have c' with style s , we decompose $f_{s,c}$ into $\sum_u f_{s,u}$ where u is the component of c as illustrated in Eq (2). Our component-wise decomposition has two benefits: 1) every Chinese character can be decomposed into the pre-defined components, therefore we can define $f_{s,c}$ for any character without observing character c with style s , 2) the number of representations to learn is dramatically decreased because the number of characters (about 20K in our experiments) is much larger than the number of components (371 in our experiments).

In our scenario, only partial components are observable from the reference set, whereas the other components are not accessible by $E_{s,u}$. However, our formulation in Eq (2) needs a full access to the whole components for a novel style. Hence, the localized style feature $f_{s,c}$ for a style s and a character c with unseen components cannot be computed, and therefore, G cannot generate a glyph with c . For example, to generate a new font library, one has to prepare at least 229 characters whose components can cover the whole 371 components, while our goal is few-shot font generation with very few references (*e.g.*, 8).

To tackle this problem, we formulate the few-shot font generation problem as a reconstruction problem. Here, we treat each component-style combination as an entry of the matrix and we assume that some component-style entries are missed. Our goal is to reconstruct a novel combination of content-style with previously observed data entries. We employ the factorization modules motivated by the low-rank matrix factorization (MF) that assumes the data matrix X has low rank k , *i.e.*, $X = A^T B$ where $\text{rank}(A) = \text{rank}(B) = k \ll \text{rank}(X)$. With this assumption, the value of the entry (i, j) can be computed by $a_i^T b_j$. However, applying conventional MF algorithms to every novel style is inefficient and computationally inefficient. Inspired by classical matrix completion approaches [48], [49], 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 the factors. Formally, we decompose $f_{s,u}$ into z_s and z_u as follows:

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

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 . Notably, [10], [50] also uses a factorization strategy for font generation; however, they directly apply 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 [48], or alternative algorithms [49] are infeasible in our scenario by repeatedly applying matrix factorization d times to obtain a d -dimensional feature $f_{s,u}$. Instead, we propose a style and component factorization module F_s and F_u that extract factors $z_s, z_u \in \mathbb{R}^{k \times d}$ from the given feature $f_{s,u} \in \mathbb{R}^d$ as follows:

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

We used a linear weight $W = [w_1; \dots; w_k] \in \mathbb{R}^{k \times d}$ and 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. For example, without any constraint, a style factor of s extracted by u , $F_s(f_{s,u})$, and a style factor of s extracted by u' , $F_s(f_{s,u'})$, will have different values, while we assume that each style factor with the same style is identical — Eq (3). Hence, we add a consistency loss that enforces the factors to have the same values for the same content or style. Intuitively, it can be obtained by minimizing all pair-wise distances of the factors, *i.e.*, $\min \sum_{u,u'} \|F_s(f_{s,u}) - F_s(f_{s,u'})\|_2^2$. Since this equation is identical to minimize the sum of distances between each factor and their average, we train the factorization modules F_s and F_u by minimizing the consistency loss $\mathcal{L}_{consist}$ as follows:

$$\begin{aligned} \mathcal{L}_{consist} &= \sum_{s \in \mathcal{S}} \sum_{u \in \mathcal{U}} \|F_s(f_{s,u}) - \mu_s\|_2^2 + \|F_u(f_{s,u}) - \mu_u\|_2^2, \\ \mu_s &= \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} F_s(f_{s,u}), \quad \mu_u = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} F_u(f_{s,u}). \end{aligned} \quad (5)$$

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 $f_{s,c} = \sum_{u \in \mathcal{U}_c} f_{s,u}$ even for the unseen component u in the reference set.

3.5 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 $z_{\tilde{s}}$ from the reference

glyphs, and by extracting z_u and f_c from the known source glyphs. Then, we combine z_c and $z_{\tilde{s}}$ to generate the localized style feature $f_{\tilde{s},u}$, as described in §3.4. Finally, we generate a glyph x using Equation (2). Formally, LF-Font consists of three sub-modules, as illustrated in Figure 4.

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 $E_{s,u}$, and the style and content factorization modules F_s and F_c . $E_{s,u}$ is simply defined by a conditional encoder as previous generative models [19], [31], where a component label u is used for the condition label, and encodes a glyph image x into several component-wise style features $f_{s,u}$.

More specifically, the component-wise style encoder $E_{s,u}$ consists of five modules: convolution, residual, component-conditional, global-context [51], and convolutional block attention (CBAM) [52]. The component-conditional module learns a set of channel-wise biases where each bias value is in charge of each component. The component-wise style encoder reflects the given component condition by adding the corresponding channel-wise bias learned by the component-conditional module to the intermediate features.

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

Our style encoding requires the explicit component labels due to the component-wise style encoder $E_{s,u}$. Utilizing character labels as weak supervision during training is affordable because we use true-type fonts easily accessible from web for training our models. However, assuming that character labels are available even in test-time can limit the applicability of LF-Font when the given reference glyph images are unlabeled (*e.g.*, historical handwriting). Furthermore, it hinders LF-Font to generate characters with unseen components during the training (*e.g.*, generating glyph images for different language systems).

To mitigate the issues, we introduce an auxiliary classifier that predicts the character label of the given glyph image. Our prediction-based inference strategy has two benefits; 1) we can remove the dependency of character labels in test-time. It can be beneficial when the reference characters are unlabeled, *e.g.*, historical handwriting. 2) the prediction-based strategy makes LF-Font handle characters having unseen components in the training set. For example, the original LF-Font will not be effective if the target character is from a different language system, such as Korean characters. The detailed discussion and experimental results are in §4.7.

Content encoding. Although our 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 the generation of 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 $\tilde{x}_{s,c}$ by combining the localized style representations $f_{s,c}$ from the style encoding and the global complex structural representation f_c from the encoding.

3.6 Training

Given the source glyph x and the references \mathcal{X}_r with 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 to generate glyph \tilde{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. We employ a multi-head conditional discriminator for style label s and character label c . The hinge GAN loss [53] was used.

$$\begin{aligned} \mathcal{L}_{adv}^D &= -\mathbb{E}_{(x,s,c)\sim p_{data}} \min(0, -1 + D_{s,c}(x)) \\ &\quad -\mathbb{E}_{(\tilde{x},s,c)\sim p_{gen}} \min(0, -1 - D_{s,c}(\tilde{x})) \quad (6) \\ \mathcal{L}_{adv}^G &= -\mathbb{E}_{(\tilde{x},s,c)\sim p_{gen}} D_{s,c}(\tilde{x}). \end{aligned}$$

L1 loss and feature matching loss. These objectives enforce the generated glyph \tilde{x} to reconstruct the ground truth glyph x at the pixel level and feature level.

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

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 an additional component-wise classifier Cls that classifies the component label u of the given component-wise style feature $f_{s,u}$. We optimized the cross-entropy loss (CE) as follows:

$$\mathcal{L}_{cls} = \sum_{\tilde{u} \in U_{\tilde{z}}} \text{CE}(Cls(f_{s,\tilde{u}}), \tilde{u}) + \sum_{u \in U_c} \text{CE}(Cls(f_{s,u}), u), \quad (8)$$

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

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

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

where $\lambda_{L1}, \lambda_{feat}, \lambda_{cls}, \lambda_{rep}$ are hyperparameters that control 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 used an Adam [54] optimizer with a learning rate of 0.0008 for the discriminator, and 0.0002 for the others. We trained the model in two phases for stability.

Our two-phase training is designed for a stable optimization of factorization modules. The role of the factorization modules is to reconstruct the component-wise style representations for the unseen components in the reference set. We observe that jointly training the factorization module

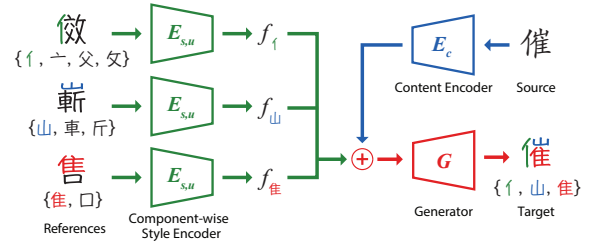


Fig. 5. **Mini-batch construction for phase 1 training.** During the first phase of the training, we constructed a mini-batch with a coherent style, where the component labels of the input glyphs cover the component labels of the target character.

and randomly initialized features at the same time can make the convergence unstable as shown in our ablation study (§4.4.6). Our two-phase training first learns component-wise style representations and then learns the factorization modules. At the first phase, we do not train the factorization modules by building a mini-batch in a way that the target character components are completely covered by the input character components (Figure 5). After training the component-wise style representations, we train the factorization modules with various styles. The detailed discussion of our two-phase training strategy is in §4.4.6.

In the first phase, we train the model without factorization modules as in Eq (2) until 800k iterations for Chinese and 200k iterations for Korean. Here, the model is trained to generate a target glyph from the component-wise style features $f_{s,u}$, extracted by the style encoder $E_{s,u}$ from the reference set \mathcal{X}_r . The content feature f_c is extracted by the content encoder E_c from the source glyph. We constructed a mini-batch with pairs of a reference set, source glyph, and target glyph. To build each pair, we randomly selected a style from the training style set and constructed 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 . The source font was fixed throughout the training period. We illustrate an example of mini-batch construction in Figure 5. Here, $\lambda_{consist}$ is set to 0.0, and the generator G and the component classifier Cls take the original component-wise style features from $E_{s,u}$, e.g., without the factorization procedure, as their input.

After a sufficient number of iterations, we began the second phase of training. We jointly trained all modules with the full objective function for 50k iterations. All the component-wise style features used in the first phase are replaced by the reconstructed component-wise style features from style and component factors as Eq (3). The mini-batch for phase 2 training was constructed differently from phase 1. Our model cannot deal with component-wise style features that are not seen in the reference set if factorization modules are not available. Because the model in the first phase deactivates the factorization modules, we construct the mini-batch by having the images in the reference set and target glyph share the same style. In the second phase, we build the mini-batch by selecting the images with various styles in the reference set, and the target glyph with one of the reference styles. During the second phase of training, we also enforced the model to reconstruct the reference images using the original component-wise style features. This was

to construct a model that can handle both the original, and reconstructed component-wise style features.

3.7 LF-FontMix: Font generation as data augmentation

Mix-based augmentations, such as Mixup [55] and CutMix [56], are widely used for the state-of-the-art image recognition models. While the simple and random mix strategy surprisingly works well, the augmented images by Mixup and CutMix are unnatural, limiting their power in complex and fine-grained datasets [57]. Character recognition systems are complex and fine-grained systems that also need careful mixing strategies as fine-grained classification tasks.

We propose LF-FontMix, a novel mix augmentation for character recognition tasks. Instead of generating mixed images in the pixel domain, we mix two images in the style factor space. Formally, let $z_{s,1}$ and $z_{s,2}$ be style factor of images x_1 and x_2 defined by Eq (4), respectively. Then the mixed style feature \hat{f}_s by LF-FontMix is defined as follows:

$$\hat{f}_s = \sum_{u \in U_c} \mathbf{1}^\top ((\lambda z_{s,1} + (1 - \lambda) z_{s,2}) \odot z_u), \quad (10)$$

where $\lambda \in [0, 1]$ is a combination ratio, sampled from $\text{Beta}(\alpha, \alpha)$ as previous methods [55], [56]. We generate a mixed image x_{mix} by $x_{mix}(x_1, x_2) = G(\hat{f}_s, f_c)$. Here, the target label is the same as the target character c . The character-level LF-FontMix is defined similarly as Eq (10), while the target label is mixed as previous works [55], [56]. The style mixing strategy makes complex and diverse font styles without mixing the labels. The character mixing strategy mixes images and labels at the same time, but the augmented styles can be limited. In practice, we randomly alternate style-level and character-level LF-FontMix for every iteration to take the advantages of both two strategies.

4 EXPERIMENTS

This section compares the results of LF-Font and previous methods for Chinese few-shot font generation. We first introduce the datasets and evaluation metrics (§4.1), and comparison methods (§4.2). We compare the results of LF-Font and previous methods for Chinese few-shot font generation in §4.3. Extensive analysis on each module (§4.4) showed that our design choice successfully dealt with the

	Localized style?	Contents encoder?	Restricted to generate
SA-VAE	✗	✗	unseen chars (train)
EMD	✗	✓	
AGIS-Net	✗	✓	unseen components (refs.)
FUNIT	✗	✓	
DM-Font	✓	✗	
DG-Font	✗	✓	
Ours	✓	✓	

TABLE 1. Comparison of LF-Font with other methods. 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.

few-shot font generation task, e.g., localized style representation (§4.4.1), content encoder (§4.4.2), utilizing weakly-supervised component labels to learn complex font information (§4.4.3), factorization modules (§4.4.4), the effectiveness of style representations in extreme cases (§4.4.5), and two-phase training strategy (§4.4.6). We also provide experimental results with various sizes of reference images, e.g., from 1 to 256 (§4.5) and comparisons in Korean generation tasks (§4.6). Finally, in §4.7, we present a new prediction strategy for LF-Font when reference images are unlabeled, e.g., the character labels are not accessible during test-time. In addition, we show the effectiveness of LF-Font in terms of data augmentation in §4.8.

4.1 Datasets and evaluation metrics

We collected 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 were used for the evaluation. The models are separately evaluated with 2,615 *seen characters* and 280 *unseen characters* to measure the generalizability of the unseen characters.

We evaluated the visual quality of the generated glyphs using various metrics. To measure the accuracy of the generated glyphs matching their ground truths, LPIPS [58] with ImageNet pre-trained VGG-16 was used. LPIPS is popularly used to assess the similarity between two images by considering the perceptual similarity.

We further assessed the visual quality of the generated glyphs in two aspects: content-preserving and style-adaptation [6]. We trained two classifiers to distinguish the style or content labels of the test dataset. Note that we trained the evaluators independently from our generation models, and the character and font labels for the evaluation did not overlap with the training labels. ResNet-50 [59] was employed for the backbone architecture. Compared to photorealistic images, glyph images are highly sensitive to local damage or distinctive local component-wise information. We developed evaluation classifiers by employing CutMix augmentation [56], which leads to a model that can learn localizable and robust features [60] using the AdamP optimizer [61]. We set a CutMix probability and the CutMix beta to 0.5 and 0.5, respectively. The batch size, learning rate, and number of epochs were set to 64, 0.0002, and 20, respectively. We report the accuracies of the generated glyphs using the *style-aware* and *content-aware* models, respectively. We also used each classifier as a feature extractor and computed the Fréchet inception distance (FID) [62]. In the experiments, we denote metrics computed by content and style classifiers as *content-aware* and *style-aware*, respectively.

Finally, we employ a new evaluation metric that measures how the generated images can reflect a novel font style, while LPIPS and classifier-based metrics only can measure the property indirectly. We use the trained style classifier where the style classifier predicts 482 font styles, including 15 unseen test styles. After we generate images by using 15 novel font styles, we predict font styles of generated images using the trained style classifier. We report

		LPIPS ↓	Acc (S) ↑	Acc (C) ↑	Acc (Hmean) ↑	FID (S) ↓	FID (C) ↓	FID (Hmean) ↓	p_{unseen}
Seen chars	SA-VAE (IJCAI'18)	0.310	0.2	41.0	0.3	231.8	66.7	103.6	0.01
	EMD (CVPR'18)	0.248	11.9	63.7	20.1	148.1	25.7	43.8	0.19
	AGIS-Net (TOG'19)	0.182	34.0	99.8	50.7	79.8	4.0	7.7	0.41
	FUNIT (ICCV'19)	0.217	39.0	97.1	55.7	58.5	3.6	6.8	0.47
	DM-Font (ECCV'20)	0.275	10.2	72.4	17.9	151.8	8.0	15.2	0.25
	DG-Font (CVPR'21)	0.189	46.9	98.8	63.6	54.0	2.6	5.0	0.57
	LF-Font (proposed)	0.169	75.6	96.6	84.8	40.4	2.6	4.9	0.83
Unseen chars	EMD (CVPR'18)	0.250	11.6	64.0	19.7	151.7	41.4	65.0	0.19
	AGIS-Net (TOG'19)	0.189	33.3	99.7	49.9	85.4	10.0	18.0	0.41
	FUNIT (ICCV'19)	0.216	38.0	96.8	54.5	63.2	12.3	20.6	0.46
	DM-Font (ECCV'20)	0.284	11.1	53.0	18.4	153.4	26.5	45.2	0.26
	DG-Font (CVPR'21)	0.188	46.4	98.7	63.1	57.8	9.0	15.5	0.57
	LF-Font (proposed)	0.169	72.8	97.1	83.2	44.5	8.7	14.6	0.82

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. p_{unseen} indicates the ratio of the generated images correctly distinguished to unseen styles by the style classifier. All numbers are average of 50 runs with different reference glyphs.

Reference	郝攀碱荷癡捐霄晴
Source	审弱和峰潮税博城尊精崗棒常瑶晁省城荷呼猗屋群
EMD	审弱和峰潮税博城尊精崗棒常瑶晁省城荷呼猗屋群
AGIS-Net	审弱和峰潮税博城尊精崗棒常瑶晁省城荷呼猗屋群
FUNIT	审弱和峰潮税博城尊精崗棒常瑶晁省城荷呼猗屋群
DM-Font	审弱无峰湖税博城尊晴崗棒常瑶晁省城荷呼猗屋群
DG-Font	审弱和峰潮税博城尊精崗棒常瑶晁省城荷呼猗屋群
Ours	审弱和峰潮税博城尊精崗棒常瑶晁省城荷呼猗屋群
GT	审弱和峰潮税博城尊精崗棒常瑶晁省城荷呼猗屋群

Fig. 6. Generated samples. We show characters in the reference set (refer to 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 for each style are shown in Appendix. We also highlight samples that show the apparent limitation of each method using colored boxes. Each color denotes the different failure cases discussed in § 4.3.

p_{unseen} the ratio of the images predicted as unseen styles not seen styles. If the style classifier is perfect, p_{unseen} denotes the degree of training style overfitting of the given font generation method, *e.g.*, a lower p_{unseen} denotes that a model is overfitted to training styles.

4.2 Comparison methods

We compared our model with six state-of-the-art few-shot font generation methods. In this study, we did not compare our method with many-shot font generation methods, such as SC-Font [4], ChiroGAN [20], CalliGAN [22] and RD-GAN [21], because they need a lot of reference characters and a finetuning procedure for generating each font style (*e.g.*, SC-Font needs 755 references) and they are not able to handle unseen font styles. Our goal is to generate font libraries without an additional optimization procedure using very few references (*e.g.*, 8 in our experiments). To understand the similarity or dissimilarity between methods, we categorize them by whether or not they explicitly model style representations or content representations, as shown in Table 1.

SA-VAE [7] extracts a universal style feature and utilizes a content code from the character classifier instead of the content encoder. This method cannot synthesize characters that are unseen during training.

EMD [8], AGIS-Net [5], FUNIT [9], and DG-Font [36] 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 in [6], [13]. 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 [6] is the most direct competitor to LF-Font. Both DM-Font and LF-Font utilize component-wise style features to capture local details. However, DM-Font is restricted to generating a glyph whose component is not in the reference set because it uses the learned codebook for each component instead of the content encoder. Because DM-Font generates 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.

As the original DM-Font cannot generate Chinese characters, we modified the structure of DM-Font in our Chinese few-shot generation experiments. Because Chinese characters are not decomposed into the same number of components, we modified the multi-head structure in DM-Font to a component-conditioned structure similar to that in LF-Font and used the averaged component-wise style features as an input to the decoder. We also changed its attention blocks to CBAM — and eliminated the hourglass blocks in the decoder to stabilize the training. For the Korean few-shot generation experiments, we used the official DM-Font model and trained weight.

4.3 Experimental results

Quantitative evaluation. We evaluated the visual quality of the generated images using seven models with eight reference glyphs per style. To avoid randomness by the reference selection, we repeated the experiments 50 times with different reference characters. A font generation method is required to satisfy two contradictory task objectives: it should preserve content and stylize well. As an extreme failure case, it performs an identity mapping, which shows the perfect content preserving score, but it will show a zero style transfer score. Hence, we report the harmonic mean of the 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-art methods with significant gaps, for example, 20.1pp higher harmonic mean accuracy than DG-Font, and 0.3 lower harmonic mean FID than DG-Font for the unseen characters. Our method outperforms other methods in style-aware benchmarks, whereas content-aware benchmarks are not significantly damaged. For example, FUNIT, AGIS-Net, and DG-Font 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, AGIS-Net, and DG-Font focus only on content preservation, while failing to achieve good stylization. We add discussions of lower content accuracies by LF-Font in §5. Additionally, LF-Font shows 82% unseen prediction ratio, which is much higher than DG-Font (57%), FUNIT (46%), and AGIS-Net (41%). The results support that our localized style representation approach has benefits in learning complex local styles, achieving generalizability to novel styles, while other methods tend to memorize the styles in the training set.

Qualitative evaluation. We also qualitatively compared the generated samples using the methods in Figure 6. We observe that AGIS-Net often drops local details, such as serif-ness, and varying thickness (blue boxes). The green boxes show that FUNIT overly relies on the structure of the source images. Thus, FUNIT tends to destroy the local structures in the generated glyphs when the source and the overall structure of the target glyphs differ significantly. DG-Font produces good-looking results in general, but it sometimes ignores the reference style and leaves the source style as it is (yellow boxes). We argue that the universal style representation strategy of AGIS-Net, FUNIT, and DG-

Font causes these problems. We further provide an extensive analysis of style representations in the latter section.

DM-Font frequently fails to generate the correct characters. For example, in the red boxes of Figure 6, 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 causes DM-Font to suffer from the complex structures of glyphs as we observed in §4.4.2.

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

Style representation f_s	LPIPS ↓	Accuracies ↑			FIDs ↓		
		S	C	H	S	C	H
AGIS-Net	0.189	33.3	99.7	49.9	85.4	10.0	18.0
FUNIT	0.216	38.0	96.8	54.5	63.2	12.3	20.6
Universal without $E_{s,u}$	0.197	33.6	97.2	49.9	92.9	10.8	19.4
Universal with $E_{s,u}$	0.187	52.8	95.9	68.1	74.1	9.3	16.5
Localized with $E_{s,u}$	0.169	72.8	97.1	83.2	44.5	8.7	14.6

TABLE 3. **Impact of localized style representation.** The universal style 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.

	LPIPS ↓	Accuracies ↑			FIDs ↓		
		S	C	H	S	C	H
Few-shot							
DM-Font	0.284	11.1	53.0	18.4	152.5	26.3	44.8
LF-Font without E_c	0.255	36.3	15.4	21.7	100.8	28.3	44.2
LF-Font	0.169	72.8	97.1	83.2	44.5	8.7	14.6
Many-shot							
DM-Font	0.254	51.8	15.0	23.2	76.3	25.3	38.0
LF-Font without E_c	0.262	37.8	5.1	8.9	97.5	30.3	46.3
LF-Font	0.165	74.7	96.5	84.2	41.4	8.6	14.3

TABLE 4. **Impact of content representation.** We evaluate DM-Font, LF-Font without content encoder E_c , and LF-Font, in the few- (8 references) and many-shot (256 references) scenarios.

GT	破舌尖在血作
LF-Font (Localized with $E_{s,u}$)	破舌尖在血作
LF-Font without E_c	破舌尖在血作
Universal without $E_{s,u}$	破舌尖在血作
Universal with $E_{s,u}$	破舌尖在血作

Fig. 7. **Visual samples of style and content module analysis.** The visual samples in Table 4 and Table 3 are shown.

4.4 Module and parameter analyses

4.4.1 Localized vs. universal style representation

We compare two universal style encoding strategies to our localized style encoding strategy. First, we train a universal style encoder that extracts a universal style from the references. EMD, AGIS-Net, and FUNIT have employed this scheme. We also develop an 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 character-wise localized style representations using $E_{s,u}$ and factorization modules.

We conducted an 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 performance (33.6%) to AGIS-Net (33.3%) — or FUNIT (38.0%). Also, the universal style representations by adding the component-wise style encoder $E_{s,u}$ is useful for increasing 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 7. From these results, we conclude that the proposed localized style representation enables the model to capture diverse local styles, whereas the universal style encoding fails in fine and local styles.

4.4.2 Content encoder

Although localized style encoding brings remarkable improvements in transferring a target style, our 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 to capture the structural information. Here, we examine various content-encoding strategies: LF-Font without content-encoding (generating the target glyph with localized style features alone), DM-Font (*persistent memory* for content encoding), and LF-Font. DM-Font replaces the content encoder with *persistent memory*, which is a learned codebook defined for each component. Note that DM-Font cannot generate unseen reference components; thus, we replace the unseen component features with the source style features. To remove unexpected effects from this source style replacement strategy, we reported many-shot (256 references) results in addition to the few-shot (8 references) results.

In Table 4, we observe that the content encoder notably enhances the overall performance (21.7% \rightarrow 83.2% in few-shot harmonic mean accuracy). Because there is no content information, the style encoder of LF-Font without E_c should encode both the style and content information of each component. However, as the style encoder is optimized for modeling local characteristics, it is limited to handling global structures (*e.g.*, the positional relationship of components). In addition, because a combination of a component set can be mapped to diverse characters, as shown in 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 shown in Figure 7.

Similar to the content encoder, the persistent memory strategy proposed by DM-Font, moderately improves the content performance (15.4% \rightarrow 53.0%); but shows worse stylization owing to the source style replacement strategy. Furthermore, both LF-Font without E_c and DM-Font suffer from a content performance drop in the many-shot setting.

$\mathcal{L}_{consist}$	\mathcal{L}_{cls}	LPIPS \downarrow	Accuracies \uparrow			FIDs \downarrow		
			S	C	H	S	C	H
\times	\times	0.206	44.5	76.3	56.2	82.1	15.0	25.4
\checkmark	\times	0.195	47.2	88.6	61.6	77.9	10.7	18.8
\times	\checkmark	0.169	69.3	97.2	81.1	49.8	8.6	14.7
\checkmark	\checkmark	0.169	72.8	97.1	83.2	44.5	8.7	14.6

TABLE 5. **Impact of objective functions.** We report the results of the different combinations of consistency loss $\mathcal{L}_{consist}$ and the component-classification loss \mathcal{L}_{cls} . Our design choice is the bottom row, which shows the best overall performance.

k	LPIPS \downarrow	Accuracies \uparrow			FIDs \downarrow		
		S	C	H	S	C	H
4	0.169	71.0	98.0	82.3	46.1	8.8	14.8
6	0.168	72.0	98.0	83.0	44.6	8.6	14.5
8 †	0.169	72.8	97.1	83.2	44.5	8.7	14.6
10	0.167	71.4	97.5	82.4	45.6	8.6	14.4

TABLE 6. **Factor size study.** † used in the remaining results.

This is because their style encoders suffer from encoding the complex structures — *e.g.*, relative size or positions — of the unseen styles, as shown in Figure 6 (the yellow boxes).

4.4.3 Weakly supervised component labels

We analyzed the importance of the image-level component label as weak supervision to learn localized style representations. We utilized the component labels in two modules. First, we use the component labels for the component-wise style encoder $E_{s,u}$, whose importance has already been demonstrated in the previous section. The other module is the auxiliary component classifier Cls , which guides the local features extracted by $E_{s,u}$. We conducted a loss ablation study and demonstrated the effect of utilizing weak component-level supervision by \mathcal{L}_{cls} . Table 5 illustrates that utilizing the component supervision is a critical factor for capturing both diverse local style and global content structure; the performance gains are significant such as 47.2% \rightarrow 72.8% for the style accuracy, 88.6% \rightarrow 97.1% for the content accuracy, and 61.6% \rightarrow 83.2 for their harmonic mean, respectively. Here, using only $E_{s,u}$ still produces better style-aware performance (47.2%) than AGIS-Net (33.3%) and FUNIT (38.0%), but the content-aware performance is degraded. We speculate that this is because of insufficient component supervision. We observe that \mathcal{L}_{cls} has a particularly large impact on the style-aware and content-aware performance: 47.2% \rightarrow 72.8%, 88.6% \rightarrow 97.2%. These results demonstrate that, to improve the overall performance, employing Cls following $E_{s,u}$ plays a crucial role because the classifier provides sufficient component supervision to the model.

4.4.4 Factorization modules

The factorization modules (F_s, F_u) takes a key role in LF-Font in terms of the generalizability to novel styles. In this subsection, we show the reconstruction capability of the full feature set from the given features according to the presence of constraints and factor size.

We first report the performance of the models when the factors are unconstrained, that is, z_u and z_s are not unique for component u and style s in Table 5. The factors constrained by $\mathcal{L}_{consist}$ improve the overall performance; 69.3%



Fig. 8. **One-shot generation results.** The reference characters and the resultant images were 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.

→ 72.8% for the style accuracy, 97.2% → 97.1% for the content accuracy, and 81.1% → 83.2% for the harmonic mean. These results show that the constrained factors contribute to the performance improvements but are less effective than the presence of weak component-level supervision, \mathcal{L}_{cls} .

Table 6 shows performances by varying the factor size k from 4 to 10. We observe that a larger k enhances the harmonic mean performance (82.3% → 83.2% when we set k as 4 → 8), but the overall performance converges after $k \geq 8$. For the efficiency, we use $k = 8$ in this paper.

4.4.5 Style representation analyses

One-shot generation. We visualize the extreme case, the one-shot generation, shown in Figure 8. We observe that when the reference glyph is too simple to extract solid component-wise features (the second row in Figure 8), the generated images show poor visual quality. This may be an issue concerning style factors, not E_c , because the same content features (from E_c) are used to successfully generate other samples. Hence, the reference selection with rich local details is critical to high-quality generation.

Style and character interpolation. Figure 9a shows the style-interpolated images; we linearly interpolate only the style factors z_s extracted from the character-wise style features while remaining the component factors z_c and the content representations. We interpolate images in the character-level similarly (Figure 9b); we interpolated the component factors z_u and the content representations while remaining the style factors z_s . The style interpolation results show that LF-Font provides semantically meaningful style features thus presents well-interpolated local features. A smooth transition across two different styles and content also demonstrates that LF-Font leads to well-disentangled content-style representations while capturing diverse font-specific characteristics and the character contents.

Character label	LPIPS ↓	Accuracies ↑			FIDs ↓		
		S	C	H	S	C	H
Prediction	0.171	74.1	94.8	83.2	44.3	8.8	14.7
Ground-truth	0.169	72.8	97.1	83.2	44.5	8.7	14.6

TABLE 7. **Comparison of different character label rules on test-time.** “Prediction” denotes that LF-Font uses the predictions by an auxiliary character classifier, while “Ground-truth” uses the ground-truth character labels directly.



(a) Style interpolation (b) Character interpolation

Fig. 9. **Interpolation results.** The interpolated images by LF-Font are shown. For each figure, we mix two images from the leftmost and the rightmost images. We provide three different mixing strategies (a) mixing images in style representation only and (b) mixing images in content representations only.

Training	LPIPS ↓	Accuracies ↑			FIDs ↓		
		S	C	H	S	C	H
End-to-end	0.208	36.6	91.3	52.2	663.8	15.3	30.8
Two-phase (proposed)	0.169	72.8	97.1	83.2	44.5	8.7	14.6

TABLE 8. **Ablation of two-phase learning strategy.** We compare the proposed two-phase learning strategy and the end-to-end learning with only a single phase.

4.4.6 Two-phase training vs. end-to-end training

We show the effectiveness of the proposed two-phase training strategy compared to the end-to-end single-phase strategy. For the end-to-end training, we mix mini-batch selection policies of phase 1 and 2. Table 8 shows the results. Our two-phase training strategy shows much more stable convergence than the end-to-end strategy. As we discussed in §3.6, our two-phase training strategy helps the factorization modules to learn factorization rules on stable features not highly varying features.

4.5 Reference size study

Figure 10 shows the performance of few-shot methods by varying the number of the references from 1 to 256. In the style-aware metrics, LF-Font performs remarkably better than all other methods. Surprisingly, LF-Font using one reference outperforms others using many references (e.g. 256.) in style-aware metrics. Despite the impressive style-aware performance, LF-Font shows less stable content-aware performances than FUNIT, AGIS-Net, and DG-Font in the low reference regime. However, we observe that the samples generated by FUNIT, AGIS-Net, and DG-Font with very few references are rarely stylized, but only maintain the source shape in Figure 11. As shown in the one-shot generation results (Figure 8), when the number of references is extremely small, LF-Font can be sensitive to the reference sampling, that is, more complex references provide rich local representations, — thus improving generation performance.

4.6 Extension to other languages

We report the Korean few-shot generation results by LF-Font, AGIS-Net [5], and DM-Font [6] with four reference

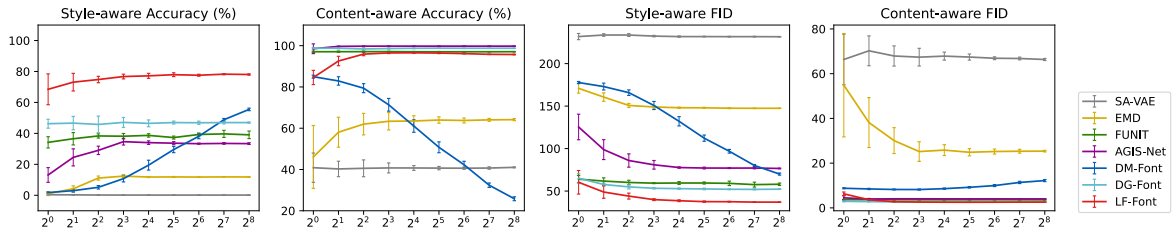


Fig. 10. **Performance changes by varying size of the reference set.** We report how the performance of each model is affected by the size of the reference set \mathcal{X}_r . The style-aware performance and content-aware performance are evaluated by generating seen characters, and each recorded two metrics: accuracy (higher is better), and FID (lower is better). Each graph shows the average performance as a line, and errors as an error bar.

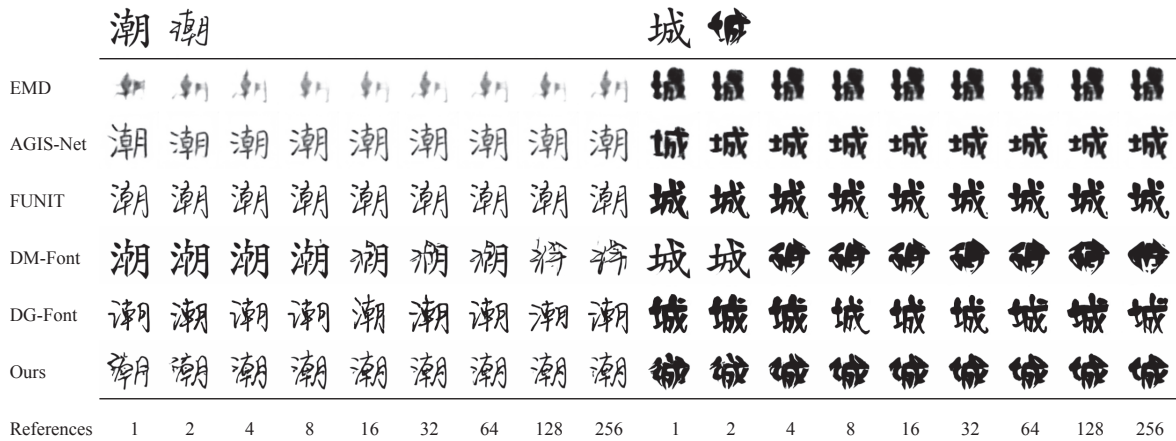


Fig. 11. **Generated samples by varying reference set size.** Each row shows the samples generated by each model. The source and target glyphs are displayed in the top row.

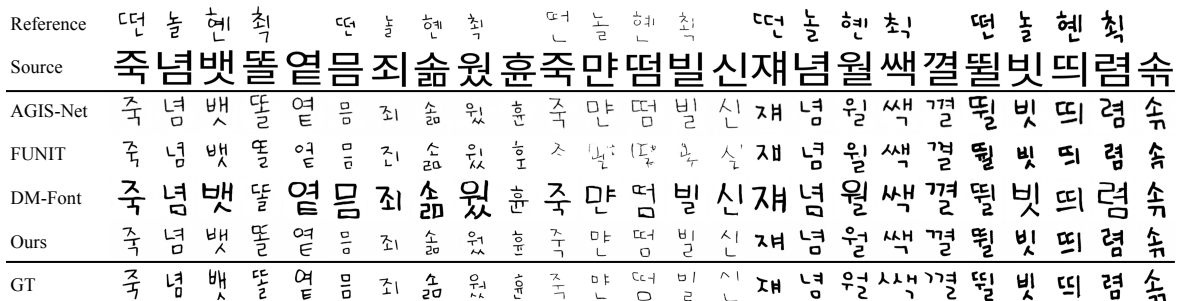


Fig. 12. **Korean few-shot generation samples.** The samples generated by each model and ground truth glyphs are shown. The samples were generated with four reference images, which are shown in the top row.

	LPIPS ↓	Acc (S) ↑	Acc (C) ↑	Acc (Hmean) ↑	FID (S) ↓	FID (C) ↓	FID (Hmean) ↓
AGIS-Net (TOG'19)	0.188	3.9	97.5	7.5	108.1	7.8	14.5
FUNIT (ICCV'19)	0.202	7.3	85.1	13.4	68.4	9.6	16.8
DM-Font (ECCV'20)	0.266	3.4	96.3	6.5	126.3	19.0	33.0
LF-Font (proposed)	0.145	41.6	98.4	58.5	47.2	4.9	8.9

TABLE 9. **Performance comparison on few-shot Korean font generation scenario.** We report LPIPS, FID and accuracy measures for AGIS-Net, DM-Font and LF-Font. All numbers are average of 10 runs with different reference glyphs.

glyphs in Table 9. We used the same training and test datasets used by Cha *et al.* [6], as well as the evaluation classifiers. Notably, Cha *et al.* used a larger reference size (28 references) and employed a special sampling strategy in which the sampled reference set covers the complete component labels. Following our main experimental protocol, we report the averages of LPIPS, FID, and accuracies of ten different runs with different reference selections to reduce

the randomness to the reference selection. In the table, we observe that LF-Font outperforms DM-Font and AGIS-Net in terms of overall metrics, particularly in *style-aware* metrics. The visual examples are illustrated in Figure 12.

4.7 Generation with pseudo-character labels

LF-Font requires explicit character labels (or component labels) even in test-time. In this subsection, we compare the

pseudo-character label-based prediction strategy (discussed in §3.5) to the ground-truth character labels. We evaluate our model in two different scenarios. First, we test the models in the **in-domain transfer scenario**, *i.e.*, training a model on Chinese images and evaluating a model on Chinese images as same as Table 6. Second, the models are evaluated on **zero-shot cross-lingual scenario**; we train a model on Chinese glyphs and evaluating the model by generating Korean images.

In-domain transfer scenario. Table 7 shows the comparison of the ground-truth character label rule and prediction-based character-pseudo label rule in Chinese-to-Chinese generation scenario. Note that we use the same LF-Font model trained on Chinese script for each character label rule. In the table, the prediction-based solution shows slightly lower content performances due to the inherent classification errors by our auxiliary character classifier. Interestingly, we observe that the prediction-based solution shows slightly better style performances than the GT solution, thus two methods show similar harmonic mean performances. It implies that the correct component condition is not necessary for better stylization, but it is required for better content-preserving. We add related discussions in §5.

Zero-shot cross-lingual scenario. We slightly modify the generation procedure of LF-Font to handle unseen language systems by omitting component conditions from the component-conditioned encoder. This enables the representation of LF-Font to have universal style. Table 11 shows that the performances on zero-shot cross-lingual (Chinese-to-Korean) generation tasks are significantly improved by our prediction-based strategy. Here, we only report accuracies because FID and LPIPS need ground-truth target characters of the given style but our test fonts do not have the target Korean characters. Note that since Korean and Chinese do not share their components, the vanilla LF-Font cannot utilize the power of localized style representations in this scenario. The Chinese-to-Korean generation results are also aligned with Table 3; our localized style representation is the key of the high-quality generation performances.

	Vanilla	CutMix	Ours (S)	Ours (C)	Ours (S,C)
Accuracy	45.4	51.6	62.4	63.9	66.8

TABLE 10. Comparison of different font augmentations. Vanilla, CutMix and LF-FontMix (Ours) are shown.

4.8 Font generation as data augmentation

LF-Font shows plausible style- or character-interpolated images for the given two glyph images (Figure 9), showing its potential as the effective augmentation policy, LF-FontMix. We compare our LF-FontMix (§3.7) to the vanilla strategy (without any augmentation) and CutMix [56]. We train character classifiers that predict 6,166 distinct characters using 5 images per character, *i.e.*, when styles are diverse and each style has very few images. The classifiers are optimized by the AdamP optimizer [61] for 90 epochs with the initial learning rate 0.0002 decayed by the cosine annealing scheduler. The batch size is 256. For all mix-based strategies, the half of mini-batch images are mixed while the remaining

half images are used as the original. The mix combination ratio λ is sampled from Beta (0.5, 0.5).

We investigate three variations of LF-FontMix; style-mix, character-mix and style-character-mix (§3.7). We compare the vanilla, CutMix, and LF-FontMix strategies in Table 10. LF-FontMix remarkably improves character recognition performances compared to the vanilla (45.4% \rightarrow 66.8%) and CutMix (51.6% \rightarrow 66.8%). We also confirm that solely adopting the style-mix or the character-mix enhances the performances. Interestingly, although style-mix does not mix labels as CutMix or character-mix, it achieves higher accuracy (62.4%) than the vanilla and CutMix by augmenting diverse font styles, while LF-FontMix (C) achieves better accuracy by mixing images and labels at the same time (63.9%). Our LF-FontMix (S,C) takes the advantages of style-mix and character-mix, achieving the best accuracy (66.8%).

5 DISCUSSION AND LIMITATIONS

In this section, we discuss the limitations of LF-Font and the possible future research directions.

Low content accuracies and failure cases on the low-shot scenario. In Table 2, LF-Font shows lower content accuracies than other methods, *e.g.*, LF-Font shows 96.6% unseen character accuracy while AGIS-Net shows 99.7% accuracy. There are two aspects why LF-Font shows lower content classification accuracies than others. First, LF-Font focused on learning various local styles. As shown in p_{unseen} in Table 2, LF-Font is capable of being generalized to novel styles while other methods heavily rely on the training styles and focus on content preserving. The qualitative results in Figure 6 show that the comparison methods overly rely on the structure of the source image. In other words, there exists the trade-off between content preserving and stylization in font generation tasks, *e.g.*, if a model generates source images without any stylization, the content accuracies will be always 100%. Hence it is important to simultaneously consider two different metrics in terms of content score and style score to evaluate font generation methods. Our LF-Font shows the best harmonic mean accuracy on both seen and unseen characters.

Nonetheless, we also observe that our method can fail to generate very novel styles with very few references. We report two cases, 1) in Figure 8, we observe that if the reference components are too simple, LF-Font fails to preserve the complex structure of the source characters. 2) in Figure 14, LF-Font fails to generate images with very novel font style, such as a font with a novel decoration (circles and butterflies in the top row) or a font with non-uniform outlines (in the bottom row). We presume that it is because the component frequency skewness of Chinese script hinders the training of our component-wise encoder; Chinese script shows long-tailed components (Figure 13).

Beyond font generation tasks. In this paper, we focus on few-shot font generation tasks by capturing complex local styles of the font domain. Although, our key idea on localized style representations can be extended to general generation tasks such as attributed-conditioned generation tasks, there are some challenges on extending LF-Font to general attributed-conditioned generation tasks.

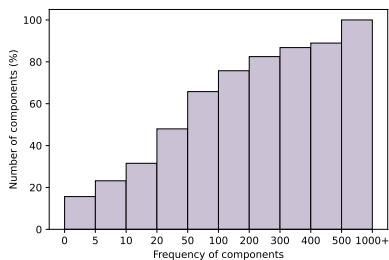


Fig. 13. **Component frequency in 19,234 Chinese characters.** Over 50% of components only appear in less than 50 characters.



Fig. 14. **Generation results of LF-Font on very novel fonts.**

	Accuracies ↑		
	S	C	H
AGIS-Net	14.1	34.0	20.0
DG-Font	46.5	41.2	43.6
FUNIT	11.8	67.2	20.1
LF-Font without $E_{s,u}$	52.1	35.2	42.0
LF-Font with pseudo character label	70.9	42.9	53.4

TABLE 11. **Zero-shot generation results.** We train font generation models in Chinese and test them on the unseen Korean target characters with Chinese references.

As we discussed in §2, there are two significant differences between font generation tasks and general attribute-conditioned generation tasks (e.g., attribute-conditional facial image generation [32], [33], [34], [35]). First, a glyph is uniquely defined for each style, while an image can be diversely mapped to different images even with the same identity. Second, it is easy to collect a glyph image with the same content but different styles. Our method heavily relies on these two font-specific properties in the training phase. Especially, in the phase 1 training, we construct a mini-batch where the input images and the target image have a coherent style, where all components of the target image can be found in the input components (Figure 5). Our mini-batch construction strategy let the model learn component-wise representations but at the same time, limits the flexibility of LF-Font to general image domains. Note that in other visual domains (e.g., facial images) constructing a mini-batch with a coherent style (e.g., the same identity) but different attributes (e.g., different gender or different hair color) are difficult, so that many attributed-conditioned generation methods formulate unpaired image generation problems while we formulate font generation as the paired scenario. Extending LF-Font to unpaired image generation tasks and general image domains will be an interesting future research direction.



Fig. 15. **Generation results on cross-lingual zero-shot scenario.** In this scenario, we generate Korean glyph images from Chinese references with models trained on Chinese script.

6 CONCLUSION

Our novel few-shot font generation method, named LF-Font, produces high-quality, complex glyphs using only a few reference images. By utilizing compositionality, a language-specific characteristics, LF-Font learns to capture local component-wise style representations from the given glyphs with the weak supervision of compositionally. That is, we only utilize the component labels, not the location of each component, skeleton, or stroke. To reduce the number of required references, we propose factorization modules, which derive the factors and then reconstruct the entire character-wise style representations from a few reference images. These factorization modules are trained on the seen character-wise style representations; but are well generalized to the unseen character-wise style representations by disentangling character-wise style representations into style and component factors. As a result, LF-Font can handle the characters unseen in the reference set, which is an important success factor when dealing with only a few reference glyphs. In the experiments, LF-Font outperformed six state-of-the-art few-shot font generation methods on both Chinese and Korean in various evaluation metrics, particularly in *style-aware* benchmarks. Our extensive analysis of our design choice supports the notion that our framework effectively disentangles content and style representations, resulting in high-quality generated samples with only a few references. Furthermore, to mitigate the inherent drawback of LF-Font, we employ an auxiliary character classifier on test-time. The proposed prediction-based inference strategy enables LF-Font to generate unseen language systems without losing localized style representations. Furthermore, we demonstrate that LF-Font can be used for data augmentation of character recognition systems.

ACKNOWLEDGMENTS

This research was supported by the Basic Science Research Program through the NRF Korea funded by the MSIP (NRF-2019R1A2C2006123, 2020R1A4A1016619), the IITP grant funded by the MSIT (2020-0-01361, Artificial Intelligence Graduate School Program (YONSEI UNIVERSITY), No.2021-0-02068 (Artificial Intelligence Innovation Hub)), and the Korea Medical Device Development Fund grant funded by the Korean government (Project Number: 202011D06).

REFERENCES

- [1] J. Han, Y. Lee, and S. Ahn, *Korean font design textbook*. Ahn graphics, 2009. 1
- [2] Y. Tian, "ziZi: Master chinese calligraphy with conditional adversarial networks," 2017. [Online]. Available: <https://github.com/kaonashi-tyc/ziZi> 1, 2
- [3] Y. Jiang, Z. Lian, Y. Tang, and J. Xiao, "DCFont: An end-to-end deep chinese font generation system," in *SIGGRAPH Asia*, 2017. 1
- [4] Y. Jiang, Z. Lian, Y. Tang, and J. Xiao, "SCFont: Structure-guided chinese font generation via deep stacked networks," in *AAAI Conference on Artificial Intelligence*, 2019. 1, 2, 3, 9
- [5] Y. Gao, Y. Guo, Z. Lian, Y. Tang, and J. Xiao, "Artistic glyph image synthesis via one-stage few-shot learning," *ACM Transactions on Graphics*, 2019. 1, 2, 3, 9, 12
- [6] J. Cha, S. Chun, G. Lee, B. Lee, S. Kim, and H. Lee, "Few-shot compositional font generation with dual memory," in *ECCV*, 2020. 1, 2, 3, 4, 8, 9, 12, 13
- [7] D. Sun, T. Ren, C. Li, H. Su, and J. Zhu, "Learning to write stylized chinese characters by reading a handful of examples," in *International Joint Conference on Artificial Intelligence*, 2018. 1, 2, 3, 9
- [8] Y. Zhang, Y. Zhang, and W. Cai, "Separating style and content for generalized style transfer," in *CVPR*, 2018. 1, 2, 3, 9
- [9] M.-Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz, "Few-shot unsupervised image-to-image translation," in *ICCV*, 2019. 1, 3, 9
- [10] N. Srivatsan, J. Barron, D. Klein, and T. Berg-Kirkpatrick, "A deep factorization of style and structure in fonts," in *Conference on Empirical Methods in Natural Language Processing*, 2019. 1, 3, 6
- [11] S. Park, S. Chun, J. Cha, B. Lee, and H. Shim, "Few-shot font generation with localized style representations and factorization," in *AAAI Conference on Artificial Intelligence*, 2021. 1, 2
- [12] S. Park, S. Chun, J. Cha, B. Lee, and H. Shim, "Multiple heads are better than one: Few-shot font generation with multiple localized experts," in *International Conference on Computer Vision (ICCV)*, 2021. 1
- [13] J. Cha, S. Chun, G. Lee, B. Lee, S. Kim, and H. Lee, "Toward high-quality few-shot font generation with dual memory," *AI for Content Creation Workshop. CVPR Workshop*, 2020. 2, 3, 9
- [14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *CVPR*, 2017. 2
- [15] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017. 2, 3
- [16] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation," in *CVPR*, 2018. 2
- [17] A. H. Liu, Y.-C. Liu, Y.-Y. Yeh, and Y.-C. F. Wang, "A unified feature disentangler for multi-domain image translation and manipulation," in *Advances in neural information processing systems*, 2018. 2
- [18] X. Yu, Y. Chen, S. Liu, T. Li, and G. Li, "Multi-mapping image-to-image translation via learning disentanglement," in *Advances in Neural Information Processing Systems*, 2019. 2
- [19] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "StarGAN v2: Diverse image synthesis for multiple domains," in *CVPR*, 2020. 2, 3, 6
- [20] Y. Gao and J. Wu, "Gan-based unpaired chinese character image translation via skeleton transformation and stroke rendering," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 2, 3, 9
- [21] Y. Huang, M. He, L. Jin, and Y. Wang, "Rd-gan: Few/zero-shot chinese character style transfer via radical decomposition and rendering," in *ECCV*, 2020. 2, 3, 9
- [22] S.-J. Wu, C.-Y. Yang, and J. Y.-j. Hsu, "Calligan: Style and structure-aware chinese calligraphy character generator," *AI for Content Creation Workshop. CVPR Workshop*, 2020. 2, 3, 9
- [23] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *CVPR*, 2016. 2
- [24] X. Huang and S. J. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *ICCV*, 2017. 2, 3
- [25] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Universal style transfer via feature transforms," in *Advances in Neural Information Processing Systems*, 2017. 2
- [26] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," in *CVPR*, 2017. 3
- [27] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, "A closed-form solution to photorealistic image stylization," in *ECCV*, 2018. 3
- [28] J. Yoo, Y. Uh, S. Chun, B. Kang, and J.-W. Ha, "Photorealistic style transfer via wavelet transforms," in *ICCV*, 2019. 3
- [29] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 172–189. 3
- [30] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410. 3
- [31] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8110–8119. 3, 6
- [32] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," in *European Conference on Computer Vision*. Springer, 2016, pp. 776–791. 3, 15
- [33] J. Choe, S. Park, K. Kim, J. Hyun Park, D. Kim, and H. Shim, "Face generation for low-shot learning using generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 1940–1948. 3, 15
- [34] Y. Lu, Y.-W. Tai, and C.-K. Tang, "Attribute-guided face generation using conditional cyclegan," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 282–297. 3, 15
- [35] Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen, "Attgan: Facial attribute editing by only changing what you want," *IEEE transactions on image processing*, vol. 28, no. 11, pp. 5464–5478, 2019. 3, 15
- [36] Y. Xie, X. Chen, L. Sun, and Y. Lu, "Dg-font: Deformable generative networks for unsupervised font generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5130–5140. 3, 9
- [37] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets v2: More deformable, better results," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9308–9316. 3
- [38] S. Azadi, M. Fisher, V. G. Kim, Z. Wang, E. Shechtman, and T. Darrell, "Multi-content gan for few-shot font style transfer," in *CVPR*, 2018. 3
- [39] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 3
- [40] J. Zeng, Q. Chen, Y. Liu, M. Wang, and Y. Yao, "Strokegan: Reducing mode collapse in chinese font generation via stroke encoding," in *proceedings of AAAI*, 2021. 3
- [41] J. Choe, S. J. Oh, S. Lee, S. Chun, Z. Akata, and H. Shim, "Evaluating weakly supervised object localization methods right," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3
- [42] J. Choe, S. J. Oh, S. Chun, S. Lee, Z. Akata, and H. Shim, "Evaluation for weakly supervised object localization: Protocol, metrics, and datasets," *arXiv preprint arXiv:2007.04178*, 2020. 3
- [43] H. Bilen and A. Vedaldi, "Weakly supervised deep detection networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2846–2854. 3
- [44] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille, "Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1742–1750. 3
- [45] J. Xu, A. G. Schwing, and R. Urtasun, "Learning to segment under various forms of weak supervision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3781–3790. 3
- [46] X. Zhang, Y. Wei, J. Feng, Y. Yang, and T. S. Huang, "Adversarial complementary learning for weakly supervised object localization," in *CVPR*, 2018, pp. 1325–1334. 4
- [47] J. Choe and H. Shim, "Attention-based dropout layer for weakly supervised object localization," in *CVPR*, 2019, pp. 2219–2228. 4
- [48] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, 2009. 5, 6
- [49] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on optimization*, 2010. 5, 6
- [50] J. B. Tenenbaum and W. T. Freeman, "Separating style and content with bilinear models," *Neural computation*, vol. 12, no. 6, pp. 1247–1283, 2000. 6

- [51] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu, "GCNet: Non-local networks meet squeeze-excitation networks and beyond," in *IEEE International Conference on Computer Vision Workshops*, 2019. 6
- [52] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *ECCV*, 2018. 6
- [53] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *ICML*, 2019. 7
- [54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015. 7
- [55] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *ICLR*, 2018. 8
- [56] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *ICCV*, 2019. 8, 14
- [57] L. Zhang, S. Huang, and W. Liu, "Intra-class part swapping for fine-grained image classification," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 3209–3218. 8
- [58] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018. 8
- [59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016. 8
- [60] S. Chun, S. J. Oh, S. Yun, D. Han, J. Choe, and Y. Yoo, "An empirical evaluation on robustness and uncertainty of regularization methods," *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2019. 8
- [61] B. Heo, S. Chun, S. J. Oh, D. Han, S. Yun, G. Kim, Y. Uh, and J.-W. Ha, "Adamp: Slowing down the slowdown for momentum optimizers on scale-invariant weights," in *International Conference on Learning Representations (ICLR)*, 2021. 8, 14
- [62] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017. 8



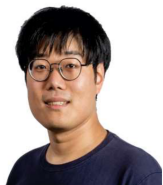
Junbum Cha is a research engineer at Clova OCR, NAVER Corp. He developed an AI go engine at the game AI team in the NHN Corp from 2017 to 2019. He received his Master's and Bachelor's degrees in computer science from Yonsei University, Seoul, Korea, in 2017 and 2015, respectively. His research interests include generative models, automated machine learning, and robust machine learning.



Bado Lee is a team leader and developer at Clova OCR, NAVER Corp. He was a developer at the Samsung Mobile Division from 2012 to 2017. He received his Master's and Bachelor's degrees in Electrical Engineering from Seoul National University, Seoul, Korea, in 2012 and 2010, respectively. His research interests focus on computer vision and image processing.



Song Park is also a Ph.D. candidate at the School of Integrated Technology, Yonsei University, South Korea. She received her B.S. degree in Integrated Technology from Yonsei University, Seoul, Korea, in 2016. Her recent research interests include computer vision and machine learning.



Sanghyuk Chun is a lead research scientist at the NAVER AI Lab. He was a research engineer at an advanced recommendation team in Kakao Corp from 2016 to 2018. He received his Master's and Bachelor's degrees in Electrical Engineering from KAIST, Daejeon, Korea, in 2016 and 2014, respectively. His research interests focus on reliable machine learning and vision-and-language.



Hyunjung Shim received her B.S. in Electrical Engineering from Yonsei University, Seoul, Korea, in 2002, and her M.S. and Ph.D. in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2004 and 2008, respectively. She was with Samsung Advanced Institute of Technology, Samsung Electronics Company, Ltd., Suwon, Korea, from 2008 to 2013. She is currently an associate professor at the School of Integrated Technology, Yonsei University. Her research interests include generative models, deep neural networks, classification/recognition algorithms, 3-D vision, inverse rendering, face modeling, and medical image analysis.

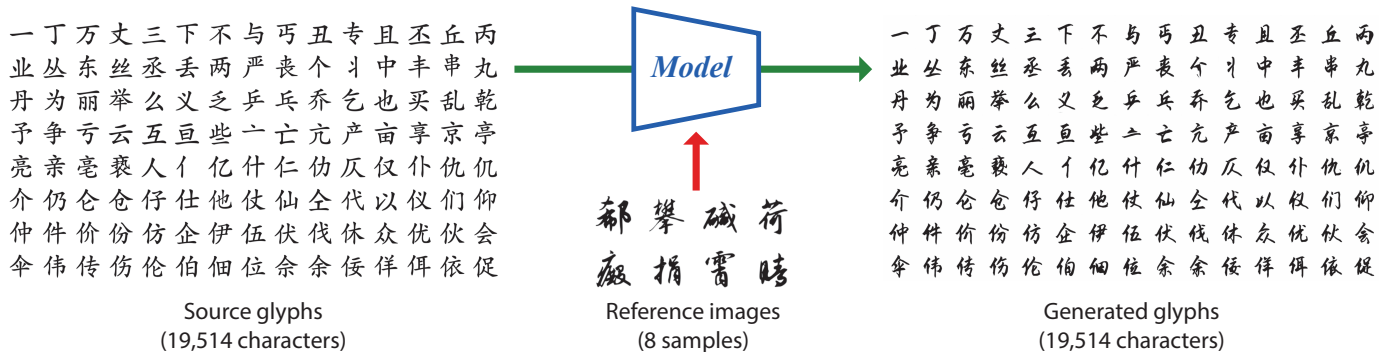


Fig. 16. Overview of few-shot font generation tasks. The few-shot font generation task aims to generate a full font library (19,514 characters in our Chinese generation scenario) with a coherent style with only a few references (eight glyphs in our experiments).

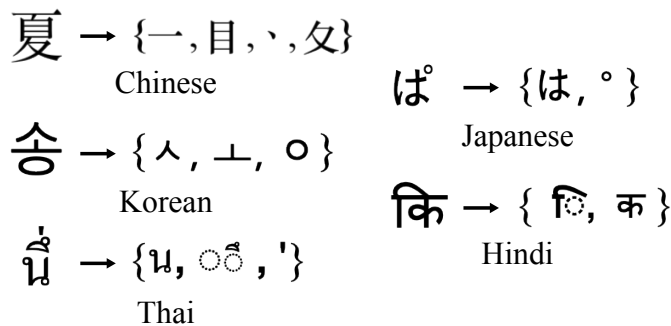


Fig. 17. Example compositionality of widely-used languages.



Fig. 18. Reference images with target styles. We visualized the eight reference samples per style used in Figure 8. Each row corresponds to the two columns of Figure 8 in the same order.

Source	秒	畝	居	筐	幘	卉	国	摇	丢	出	倍	健	典	保	何	店	峰	伸	崗	七
EMD	秒	畝	居	筐	幘	卉	国	摇	丢	出	倍	健	典	保	何	店	峰	伸	崗	七
AGIS-Net	秒	畝	居	筐	幘	卉	国	摇	丢	出	倍	健	典	保	何	店	峰	伸	崗	七
FUNIT	秒	畝	居	筐	幘	卉	国	摇	丢	出	倍	健	典	保	何	店	峰	伸	崗	七
DM-Font	秒	畝	居	筐	幘	卉	国	摇	丢	出	倍	健	典	保	何	店	峰	伸	崗	七
DG-Font	秒	畝	居	筐	幘	卉	国	摇	丢	出	倍	健	典	保	何	店	峰	伸	崗	七
LF-Font	秒	畝	居	筐	幘	卉	国	摇	丢	出	倍	健	典	保	何	店	峰	伸	崗	七
GT	秒	畝	居	筐	幘	卉	国	摇	丢	出	倍	健	典	保	何	店	峰	伸	崗	七
Source	推	常	庭	岬	体	利	叻	初	唱	付	员	崙	两	寄	弱	亩	享	兄	京	亘
EMD	推	常	庭	岬	体	利	叻	初	唱	付	员	崙	两	寄	弱	亩	享	兄	京	亘
AGIS-Net	推	常	庭	岬	体	利	叻	初	唱	付	员	崙	两	寄	弱	亩	享	兄	京	亘
FUNIT	推	常	庭	岬	体	利	叻	初	唱	付	员	崙	两	寄	弱	亩	享	兄	京	亘
DM-Font	推	常	庭	岬	体	利	叻	初	唱	付	员	崙	两	寄	弱	亩	享	兄	京	亘
DG-Font	推	常	庭	岬	体	利	叻	初	唱	付	员	崙	两	寄	弱	亩	享	兄	京	亘
LF-Font	推	常	庭	岬	体	利	叻	初	唱	付	员	崙	两	寄	弱	亩	享	兄	京	亘
GT	推	常	庭	岬	体	利	叻	初	唱	付	员	崙	两	寄	弱	亩	享	兄	京	亘
Source	呼	塌	何	合	另	冕	凤	凋	晁	击	度	役	彦	庭	糕	岌	座	况	筒	晚
EMD	呼	塌	何	合	另	冕	凤	凋	晁	击	度	役	彦	庭	糕	岌	座	况	筒	晚
AGIS-Net	呼	塌	何	合	另	冕	凤	凋	晁	击	度	役	彦	庭	糕	岌	座	况	筒	晚
FUNIT	呼	塌	何	合	另	冕	凤	凋	晁	击	度	役	彦	庭	糕	岌	座	况	筒	晚
DM-Font	呼	塌	何	合	另	冕	凤	凋	晁	击	度	役	彦	庭	糕	岌	座	况	筒	晚
DG-Font	呼	塌	何	合	另	冕	凤	凋	晁	击	度	役	彦	庭	糕	岌	座	况	筒	晚
LF-Font	呼	塌	何	合	另	冕	凤	凋	晁	击	度	役	彦	庭	糕	岌	座	况	筒	晚
GT	呼	塌	何	合	另	冕	凤	凋	晁	击	度	役	彦	庭	糕	岌	座	况	筒	晚

Fig. 19. More generation samples. We provide more generated glyphs.