Expert Systems With Applications 242 (2024) 122817

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Learning font-style space using style-guided discriminator for few-shot font generation

Ammar Ul Hassan, Irfanullah Memon, Jaeyoung Choi [*]

School of Computer Science and Engineering, Soongsil University, Seoul, South Korea

ARTICLE INFO

Keywords:
Font generation
Generative adversarial networks
Style transfer
Similarity learning
Metric learning

ABSTRACT

Font synthesis has become a quite active field of research since manual font design requires extensive domain tradecraft expertise that is labor intensive and time consuming, especially for languages with large number of characters such as Chinese and Korean. While remarkably successful, existing methods for font synthesis have major shortcomings in their requirement to fine-tune the network for unobserved font styles, which requires large reference images. Additionally, the recent few-shot font synthesis methods are either designed for specific language systems, or they operate on low-resolution images, which limits their usage. For learning distinct font styles, most methods rely on multitask discriminators that use style labels. Such dependency on style labels ignores the semantic distances between different font styles. In this paper, we tackle this font synthesis problem by learning the font style in the embedding space. To this end, we propose a font style-guided discriminator that simultaneously acts as a style encoder and adversarial network as it learns to separate font styles in the embedding space, in which the semantic distances directly correspond to the measure of font style similarity while simultaneously guiding the generator. We also design the network architecture and training procedure that mitigates the problems discussed above. Thanks to this approach, the proposed method outperforms existing state-of-the-art few-shot font generation methods both qualitatively and quantitatively.

1. Introduction

Font style is critical in helping readers to interpret implicit message in the text. Traditional manual font design requires extensive domain tradecraft expertise that is labor intensive and time consuming. Additionally, some languages have an incredibly large number of characters (e.g., Chinese > 50K, Korean > 11K); thus, the labor and cost burdens continue to grow with the expansion of language characters.

Based on the advancements in generative adversarial networks (GANs) (Goodfellow et al., 2020), recent font generation (FG) methods (Gao, Guo, Lian, Tang, & Xiao, 2019; Jiang, Lian, Tang, & Xiao, 2017, 2019; Ko, Hassan, Suk, & Choi, 2021; Muhammad, Lee, & Choi, 2023; Tian, 2017) have addressed this automatic font synthesis as an image-to-image translation (I2I) problem by utilizing conditional GANs (Isola, Zhu, Zhou, & Efros, 2017; Mirza & Osindero, 2014). These approaches rely on font-style labels (i.e., one-hot vectors) to distinguish different styles during training. However, for generating unseen font styles during inference, additional fine-tuning of the unseen reference characters is required, which only increases computational expense and time consumption. To address these limitations, more recent methods have addressed FG as a few-shot image generation problem (Cha et al., 2020; Li, Taniguchi, Lu, & Konomi, 2021; Park, Chun, Cha, Lee, & Shim, 2020, 2021; Xie, Chen, Sun, & Lu, 2021). These methods leverage multi-task discriminator (Liu et al., 2019) trained to classify each font style in an adversarial manner. Specifically, they use font-style labels and learn a mapping between font styles using multi-task discriminator. Although these methods have shown promising results, there still remains some limitations. For example, some of these methods are designed only for specific languages (Cha et al., 2020; Hassan, Memon, & Choi, 2023; Li et al., 2021; Park et al., 2020, 2021) or trained on low-resolution images (Azadi et al., 2018; Gao et al., 2019; Xie et al., 2021), which hinder their practical application. Additionally, label-guided multi-task discriminator methods do not consider the semantic distances between font styles nor they regularize the style encoders directly.

This study proposes a simple and effective method of overcoming these limitations. Inspired by the metric learning literature, our key idea involves learning the font-style similarity in the embedding space while supervising the synthesis process instead of learning font style labels. Specifically, rather than classifying the font styles using auxiliary domain classifiers (Odena, Olah, & Shlens, 2017) or a multi-task discriminator (Liu et al., 2019) during training, our approach groups characters of the same font style closer to each other and apart from the

* Corresponding author.
E-mail addresses: ammar.instantsoft@gmail.com (A.U. Hassan), iumemon@gmail.com (I. Memon), choi@ssu.ac.kr (J. Choi).

https://doi.org/10.1016/j.eswa.2023.122817
Received 24 February 2023; Received in revised form 28 October 2023; Accepted 30 November 2023
Available online 4 December 2023

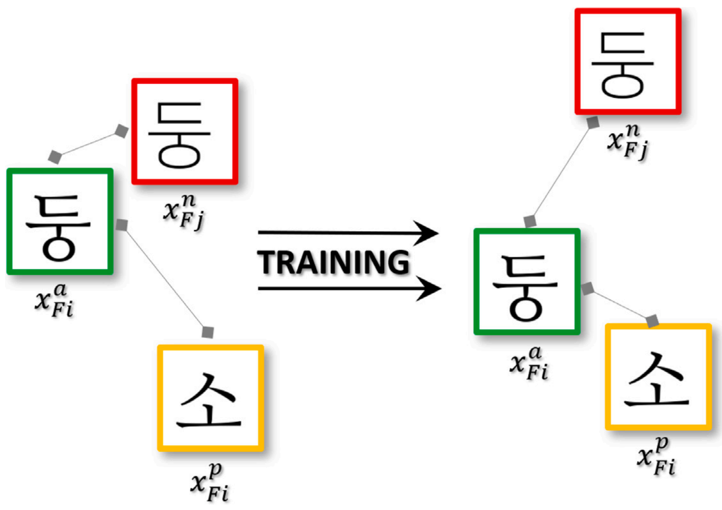


Fig. 1. Proposed font style embedding space. Our proposed method learns to minimize the distance between an anchor glyph x^a and a positive glyph x^p with the same font style F_i, but with different glyph content, and maximizes the distance between anchor glyph x^a and negative glyph x^n with the same glyph content but different font styles F_i and F_j, respectively.

others in the font-style embedding space as depicted in Fig. 1. Additionally, we fully exploit the discriminators ability to internally learn the font-style space, unlike the existing font style-specific discriminators that learn discrete space.

In this paper, we propose a novel few-shot FG (FFG) model that learns the font-style space using metric learning. Specifically, we introduce a font style-guided discriminator that has three output branches: one for extracting style features for guiding the generator, second for a typical GAN logit, and finally the third for learning font-style embedding space. Notably, we depart from the separate style encoder architecture and exploit our discriminator by combining a style encoder and a discriminator in a single module. Thus, the proposed model is lighter as a result of the elimination of one module and achieves better performance owing to the discriminator's finer font-style representation space. This strategy enables the generation of an unobserved font set with only a few samples and no architectural or additional training constraints. We qualitatively and quantitatively validated our proposed model on Chinese Hanja and Korean Hangul characters, demonstrating that it outperforms state-of-the-art (SOTA) FFG baselines. Finally, through ablation studies, we highlight the beneficial effects of the individual components.

2. Related work

2.1. I2I translation

I2I translation aims to learn a mapping function between a source and target domain while preserving the content of the source domain. Supervised I2I translation relies on paired training datasets in which each source-domain image has a corresponding image in the target domain (Isola et al., 2017; Zhu, Zhang, et al., 2017). In contrast, unsupervised I2I translation learns the source and target domain mapping without a paired dataset but with additional constraints, such as cycle consistency (Liu, Breuel, & Kautz, 2017; Zhu, Park, Isola, & Efros, 2017).

Recent methods have focused on multidomain and multimodal I2I translations using unified architectures (Choi et al., 2018; Choi, Uh, Yoo, & Ha, 2020). Even newer methods have focused on the few-shot image generation problem in which a few images of an unseen class are given, and the network generates the seen content in the style of unseen classes (Liu et al., 2019; Saito, Saenko, & Liu, 2020). These methods use domain labels to employ the auxiliary classifier (Choi et al., 2018) or a multitask discriminator (Liu et al., 2019; Saito et al., 2020) for multidomain translation.

2.2. Many-shot FG methods

Contemporary FG methods extend supervised I2I translation (Isola et al., 2017) and multidomain translation (Choi et al., 2018) architectures. For example, Tian et al. (Tian, 2017) utilized pix2pix (Isola et al., 2017) and an AC-GAN discriminator (Odena et al., 2017) for FG using a paired dataset that maps a fixed source font to a target font. DCFont (Jiang et al., 2017) adds a font feature extraction network to generate Chinese handwriting. Some methods translate character skeletons to a target style (Jiang et al., 2019; Ko et al., 2021), and others focus on stylish and colorful FG using one-stage architectures (Gao et al., 2019). However, all of these methods require additional fine-tuning steps using a vast number of reference characters.

Our proposed method generates an unseen font set without any additional fine-tuning steps on large number of reference glyphs to create a new font style. Several methods have addressed other font-synthesis tasks. For example, UFFG (Hassan, Ahmed, & Choi, 2021) utilized an AC-GAN discriminator for font family generation in an unpaired setting, and FTransGAN (Li et al., 2021) was recently proposed for Chinese-to-Latin and *vice versa* font-style transfers.

2.3. FFG methods

The goal of the FFG task is to generate glyphs in an unseen reference style using very few reference characters without additional fine-tuning. Recent methods (Cha et al., 2020; Park et al., 2020) utilize the compositionality of languages (e.g., Chinese, Korean, and Thai), but they can only be applied to specific writing systems, owing to the models explicit dependencies on prior knowledge and component labels. MX-Font (Park et al., 2021) uses a multihead encoder to extract local font features via component supervision; however, it is not generalized for languages lacking those components, such as Roman characters. CG-GAN (Kong et al., 2022) leverages a Component-Aware Module (CAM) to finely separate content and style at the component level, resulting in improved results over existing methods. Similar to the MX-Font, CG-GAN also shares the limitation of not being readily adaptable to languages that lack the specific components used in the training data, such as Roman characters. More recently, XMP-Font (Liu, Liu, Ding, He, & Yi, 2022) introduces a self-supervised cross-modality pre-training strategy and a cross-modality transformer-based encoder for improved few-shot font generation, however the model does not support unseen stroke labels due to its explicit reliance on stroke labels for conditioning style and content representations.

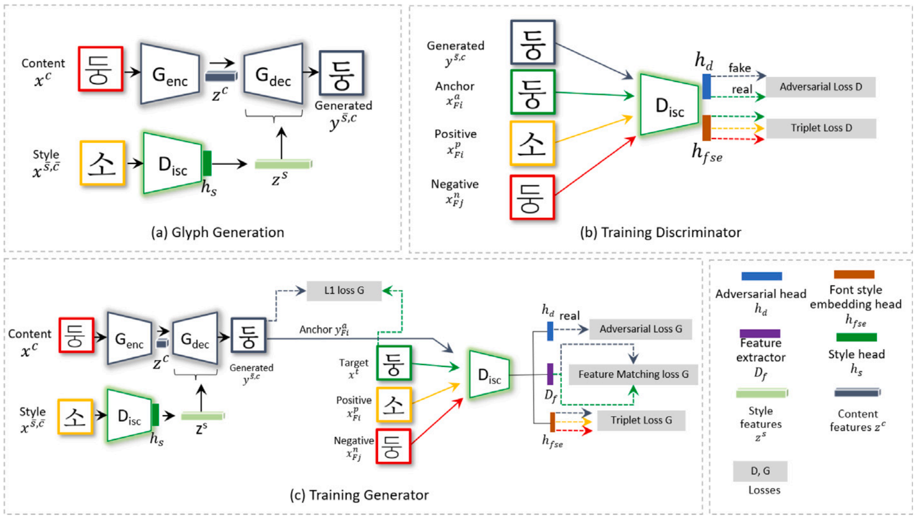


Fig. 2. Proposed few-shot font generation framework: (a) overall glyph generation process; (b) discriminator D_{isc} learns to distinguish between real and fake images via adversarial loss and learns the font style space via triplet loss D; (c) generator (G_{enc}, G_{dec}) preserves the input content via L1 loss and adopts style features z_s via triplet loss G.

Based on TUNIT (Baek, Choi, Uh, Yoo, & Shim, 2021), DGFont (Xie et al., 2021) accomplishes FG using an unpaired I2I translation (Zhu, Park, et al., 2017). However, building a paired training dataset in our scenario is inexpensive and does not limit its practical use. Furthermore, DG-Font operates on low-resolution images (80 × 80), which limits its practical usage. FSFont (Tang et al., 2022) uses cross-attention to learn fine-grained local styles from reference glyphs, achieving improved style consistency in generated content glyphs. However, this method requires specific reference characters to generate results of high quality. In addition, if it is trained with limited data, it cannot replicate the font's specific details.

All of these methods rely on font-style classification labels trained on a set of known styles. Thus, their style space does not consider the semantic distances between classes. Therefore, we focus on learning the semantic distances between distinct font styles so that this can be extended to unseen styles during inference. We accomplish this by employing metric learning via a discriminator. However, unlike other approaches, we also use our discriminator as a style encoder to fully exploit the feature space. In contrast to existing methods in which the generator and style encoder are updated simultaneously in response to the signal from the discriminator, our method updates the style encoder (proposed discriminator) independently from the generator, resulting in improved content and style disengagement. Finally, different from the most existing GAN-based FG and image generation methods that abandon the discriminator after training, ours continues to function during inference.

3. Proposed method

Our aim is to build an FFG model that learns a font-style space internally while considering the visual similarity of its fonts so that when mapping them to the embedding space, the characters will be located near those of the same style and far away from the others.

3.1. Problem definition

Given a set of character content images C, we aim to generate a glyph $y^{\bar{s},c}$ with an unseen target style $\bar{s}$ for all $x^c \in C$ with very few reference characters $x^{\bar{s},\bar{c}} \in X_r$. We want to train decoder G_{dec} to take the content features z^c and style features z^s as inputs, and generate glyph $y^{\bar{s},c}$ with reference styles $\bar{s}$ while retaining the content of the source character x^c.

To this end, we propose a novel FFG framework consisting of two modules: a font style-guided discriminator D_{isc}, that learns the visual similarity of fonts in the style space while injecting style features z^s into the generator for font-style transfer. A generator G that comprises an encoder G_{enc} and decoder G_{dec}. Encoder G_{enc} extracts the content features z^c of content image x^c, and decoder $G_{dec}(z^c, z^s)$ generates a glyph reflecting the content and style features. Fig. 2(a) illustrates the glyph generation process, and the training scheme of the entire model is shown in Fig. 2(b), and 2(c).

3.2. Font style-guided discriminator

The proposed discriminator D_{isc} is designed with three output branches above shared layers. Specifically, given an image x, the proposed discriminator returns a vector as the output. The style head h_s outputs the style features z_s to guide generator G, and discriminator head h_d outputs a typical GAN logit based on x being real or fake. The font-style embedding head h_{fse} outputs a style embedding $f(x)$ to learn the style space internally.

3.2.1. Learning the font-style space

Given a reference image representing the style of the target font, the style encoder should extract the style features of the given image. Current methods (Azadi et al., 2018; Cha et al., 2020; Gao et al., 2019; Jiang et al., 2017, 2019; Ko et al., 2021; Park et al., 2020; Tian, 2017; Xie et al., 2021) only use the feedback from the style conditional discriminators trained with a set of known styles to guide the generator in synthesizing the target style. Additionally, style encoders are not directly regularized to learn the font style. In contrast to these approaches, which learn styles using style classification in an adversarial manner, we train our discriminator D_{isc} to learn the font-style similarity in the embedding space. Moreover, instead of using a separate style encoder, we use D_{isc} to inject the style representation into the generator via h_s.

To this end, we employ a triplet loss (Schroff, Kalenichenko, & Philbin, 2015) in D_{isc} that extracts style embedding $f(x)$ from a given font image x, into an embedding space R_d, so that the squared L2 distance between all font images belonging to the same font style is small, whereas that between a pair of font images belonging to different font styles is large. To learn the font-style embedding, we train D_{isc} so that a font image x^a_{Fi} (anchor), belonging to a specific font style F_i, is closer to all other font images x^p_{Fi} (positive) of the same font style F_i than to any font image x^n_{Fj} (negative) of a different font style F_j. The objective for learning the font style space in D_{isc} is given by

$$\mathcal{L}_{triplet}(D_{isc}) = \| f\left(x^a_{Fi}\right) - f\left(x^p_{Fi}\right) \|_2^2 - \left\| f\left(x^a_{Fi}\right) - f\left(x^n_{Fj}\right) \right\|_2^2 + \alpha, \quad (1)$$

where $f(x^a_{Fi})$, $f(x^p_{Fi})$, and $f(x^n_{Fj})$ are the embeddings of anchor x^a, positive x^p, and negative x^n images, respectively, extracted from D_{isc} (i.e., font style embedding head, h_{fse}), and α is a margin between positive and negative pairs. To inject the style features z_s of a given image into generator G, we extract features from the intermediate layer of D_{isc} (style head h_s), which is later used to guide generator G for font synthesis. We also analyze the learned font style space of our proposed model and compare it with a SOTA baseline to evaluate the quality of the representation in Section 5.6.

3.2.2. Font triplet selection

During training, we choose the triplets that ensure the distance between the anchor and negative is smaller than that between the anchor and the positive. To implement this hard triplet mining (Hermans, Beyer, & Leibe, 2017), we choose anchor x^a and positive x^p, ensuring that they belong to the same font style F_i, but with a different character content. However, we also choose the negative x^n, ensuring that it has the same character content as the anchor, but with a different font style F_j. The loss in Eq. (1) is conceptually visualized in Fig. 1. It enhances the representation power of D_{isc} and influences G to improve font-style transfer efficiency. We analyze the effect of the proposed triplet selection strategy to demonstrate its effectiveness in the Section 5.5.

3.2.3. Generating realistic glyphs

To generate realistic images, we impose adversarial loss on D_{isc} and G. Specifically, we employ non-saturating GAN loss (Goodfellow et al., 2020) and R1 regularization (Mescheder, Geiger, & Nowozin, 2018).

$$\mathcal{L}_{adv}(D_{isc}, G) = E_{x^t}[\log h_d(x^t)] + E_{x^c, z_s}[\log(1 - h_d(G(x^c, z_s)))], \tag{2}$$

where x^t and x^c are the ground-truth and content images, respectively, and z_s represents the style features extracted from the discriminator-style head h_s. The training scheme of the proposed discriminator d_{isc} is shown in Fig. 2(b).

3.3. Learning to synthesize font glyph

During training, we sample two glyphs from the dataset, ensuring that they have different content and style. We then use one image as the content glyph x^c and the other as reference style glyph $x^{\bar{s},\bar{c}}$. We then pass $x^{\bar{s},\bar{c}}$ through the discriminator and extract the style features $z_s = h_s(x^{\bar{s},\bar{c}})$. Our generator G learns to translate the content glyph x^c into the generated glyph $y^{\bar{s},c} = G_{dec}(G_{enc}(x^c), z_s)$, using the style features z_s that are injected using AdaIN (Huang & Belongie, 2017) in all layers of the decoder G_{dec}. To this end, we adopt the following losses to train generator G.

3.3.1. L1 loss

We employ pixel-level supervision by minimizing the pixel-by-pixel difference between generated glyph $y^{\bar{s},c}$, and the ground truth glyph x^t.

$$\mathcal{L}_{L1}(G) = E_{x^t,\ y^{\bar{s},c}}[\|x^t - y^{\bar{s},c}\|_1]. \tag{3}$$

3.3.2. Feature matching loss

We employ feature-level supervision in our generator to improve the quality of the generated glyphs. We use the discriminator D_{isc} as a feature extractor D_f to obtain the intermediate feature maps for the target glyph x^t and generated glyph $y^{\bar{s},c}$.

$$\mathcal{L}_{Feat}(G) = E_{x^t,\ y^{\bar{s},c}}[\|D_f(x^t) - D_f(y^{\bar{s},c})\|_1]. \tag{4}$$

3.4. Font style transfer via style similarity constraint

By adapting the losses from Eqs. (1)–(4), the model can now generate the glyph contents (structure) correctly. However, the generator often ignores the style features z_s from D_{isc} and synthesizes glyphs with random font styles (i.e., degenerated cases) (Baek et al., 2021). Most current I2I translation methods use class-specific (font style in our case) discriminators to determine whether the generated image is from a given target class (Baek et al., 2021; Cha et al., 2020; Li et al., 2021; Park et al., 2020, 2021; Xie et al., 2021). Instead, we impose a style similarity constraint based on the style embedding space of the discriminator. We employ the same triplet regularization as the one used to train the discriminator (Eq. (1)) to ensure that generator G utilizes the style features z_s extracted from D_{isc}.

$$\mathcal{L}_{triplet}(G) = \|f\left(y^a_{Fi}\right) - f\left(x^p_{Fi}\right)\|_2^2 - \left\|f\left(y^a_{Fi}\right) - f\left(x^n_{Fj}\right)\right\|_2^2 + \alpha. \tag{5}$$

The major difference between Eqs. (1) and (5) is that the anchor image used in Eq. (5) is the generated glyph y^a_{Fi}, whereas we use a real glyph x^a_{Fi} in Eq. (1). This loss causes the generated glyph $y^{\bar{s},c}$ to have a style similar to the reference style image $x^{\bar{s},\bar{c}}$ and dissimilar to other glyphs (negative). The training scheme of the proposed generator G is shown in Fig. 2(c).

3.5. Overall objectives

Our final objective functions for Discriminator D and Generator G are as follows:

$$\mathcal{L}_D = \min_D \mathcal{L}_{adv}(D) + \lambda_{triplet}\mathcal{L}_{triplet}(D) \tag{6}$$

$$\mathcal{L}_G = \max_G \mathcal{L}_{adv}(G) + \lambda_{L1}\mathcal{L}_{L1}(G) + \lambda_{fm}\mathcal{L}_{Feat}(G) + \lambda_{triplet}\mathcal{L}_{triplet}(G), \tag{7}$$

where λ_{L1}, $\lambda_{triplet}$, and λ_{fm} are hyperparameters for each term and we use $\lambda = 0.1$ for both L1 and triplet losses and 1.0 for feature matching loss throughout all experiments.

3.6. Training details

Our generator and discriminator are based on StyleGAN (Karras et al., 2020) with the following modifications: StyleGAN is an unconditional model, while font synthesis requires control over the desired content and style. To address this issue, we add an encoder to the StyleGAN generator (G_{enc} in Fig. 2(a)). We replace the default constant input of StyleGAN generator with the content features z_c of size 8×8 extracted from G_{enc}. The network architecture of G_{enc} is similar to discriminator except without minibatch discrimination. Notably, we also tried the FUNIT (Liu et al., 2019) content encoder but found the proposed architecture to be more stable. The architecture of discriminator is the same as StyleGAN except that our discriminator has three output heads. The style features z_s from style head h_s are passed through four fully-connected layers before being injected into the generator using AdaIN (weight modulation). We use two fully-connected layers to transform the normalized features into a style embedding $f(x)$ of size 128 from font-style embedding head h_{fse}. The channel sizes of both generator and discriminator are reduced by half. We do not use any regularization in generator. The batch size is set to 16 and all networks are initialized using Kaiming initialization.

For the rest, we follow the default settings of StyleGAN, such as R1 regularization (Mescheder et al., 2018) in D_{isc}, Adam optimizer (Kingma & Ba, 2014), learning rates, and the exponential moving average of the generator. **Note** that during training when extracting style features z_s, we freeze the D_{isc} to prevent the style space affected by other objectives. We found with our experiments that failing to freeze D_{isc} discourages it from encoding the style representation.

Table 1
Quantitative evaluation on Korean and Chinese datasets. All methods were evaluated on both seen and unseen characters on unobserved font styles (unseen during training). The best result is denoted in boldface.

	Unseen Fonts Seen Characters						Unseen Fonts Unseen Characters					
	Methods	LPIPS↓	ACC (C)↑	SSIM↑	MS-SSIM↑	FID↓	Methods	LPIPS↓	ACC (C)↑	SSIM↑	MS-SSIM↑	FID↓
Korean	FUNIT	0.1372	91.64	0.548	0.199	28.41	FUNIT	0.1416	93.82	0.564	0.213	38.41
	LF-Font	0.1161	70.30	0.673	0.306	19.85	LF-Font	0.1261	72.40	0.584	0.298	29.85
	MX-Font	0.1036	71.95	0.627	0.278	**13.11**	MX-Font	**0.1062**	61.07	0.604	0.278	**19.63**
	Ours	**0.0942**	**96.19**	**0.697**	**0.431**	16.46	Ours	0.1215	**98.28**	**0.625**	**0.413**	21.46
	Unseen Fonts Seen Characters						Unseen Fonts Unseen Characters					
Chinese	FUNIT	0.1856	91.91	0.509	0.215	28.13	FUNIT	0.1876	90.50	0.501	0.108	20.02
	LF-Font	**0.1161**	90.05	0.580	0.361	26.67	LF-Font	0.1354	87.98	**0.557**	**0.307**	38.42
	MX-Font	0.1511	96.12	0.545	0.323	24.83	MX-Font	0.1548	95.40	0.534	0.208	21.29
	Ours	0.1327	**96.27**	**0.680**	**0.561**	**18.90**	Ours	**0.0973**	**97.16**	0.531	0.243	**12.67**

3.7. Training strategy

In every iteration, we sampled a minibatch x^a, x^p, and x^n of N images from the dataset. We choose x^a and x^p so that they belong to the same font style f_i but different character content. We choose x^n so that it has a different font style f_j from x^a but same character content. To calculate the $\mathcal{L}_{triplet}(D_{isc})$, we use x^a as anchor, x^p as positive, and x^n as negative image, respectively. We reused the x^n image as the input to the generator G_{enc}. We obtained style features z_s by feeding x^p to the discriminator D_{isc} and extracting features from style head h_s. The adversarial loss for updating the discriminator D_{isc} was calculated for x^a (real) and $G(G_{enc}(x^n), z_s)$ (fake), and the adversarial loss for updating the generator G was calculated for $G(G_{enc}(x^n), z_s)$ (real). We reused the x^a image as the ground truth x^t for $\mathcal{L}_{L1}(G)$ and $\mathcal{L}_{Feat}(G)$ losses.

We mostly follow the settings of StyleGAN (Karras et al., 2020), e.g., R1 regularization (Mescheder et al., 2018) for stable training in the discriminator using $\gamma = 10$, Adam (Kingma & Ba, 2014) optimizer with 0.002 learning rate, $\beta_1 = 0.0$ and $\beta_2 = 0.99$, an exponential moving average of the generator, leaky ReLU (Maas, 2013), equalized learning rate (Karras, Aila, Laine, & Lehtinen, 2017) for all layers. We used a batch size of 16 for training our proposed model on 128×128 and 256×256 images on single NVIDIA RTX 3080 Ti GPU.

4. Experiments

In this section, we describe the evaluation setup and experimental results. We also compare our proposed model with SOTA FFG methods and analyze our design choices for additional insights.

4.1. Datasets

We evaluated our model using Chinese and Korean FG tasks due to their complex structures and large number of characters. Note that our model has no architectural constraints (not designed for specific languages), so it can be easily extended to other writing systems, such as roman characters, which are simple in overall character shape and fewer in number. We collected 110 and 101 diverse publicly available Chinese and Korean fonts, respectively, from the web. Each Chinese font contained the 1000 most commonly used characters. The dataset was randomly divided into 80% training and 20% testing. The models were evaluated separately with 800 seen characters and 200 unseen ones to measure the generalizability of the unseen.

In the Korean dataset, each font contained the 2350 most commonly used characters. Like the Chinese dataset, the Korean dataset was randomly divided into 80% training and 20% testing. The models were evaluated separately with 2000 seen and 350 unseen characters. Note that in all experiments the models were evaluated on unobserved fonts (fonts not seen during training).

4.2. Baselines and evaluation metrics

We compared our proposed model to the following SOTA FFG models: (1) FUNIT (Li et al., 2021), (2) LF-Font (Park et al., 2020), and MX-Font (Park et al., 2021). For FUNIT, we used a modified variant designed for the FFG task (Cha et al., 2020). We used the official implementations[1] of these baselines. For a fair comparison, we used a fixed-source font during inference for all methods and tested them in a few-shot setting. We used five reference glyphs per style during inference in all experiments.

We evaluated the models using pixel, perceptual, and content classification metrics. For the pixel-level evaluation, we used the structural similarity index (SSIM) and multi-scale (MS) SSIM. To assess the similarity between the two glyphs while considering perceptual similarity, we used LPIPS (Zhang, Isola, Efros, Shechtman, & Wang, 2018). We also trained a character classifier with supervision. ResNet50 (He, Zhang, Ren, & Sun, 2016) was used as the backbone, and we used the mean Fréchet inception distance (FID) (Parmar, Zhang, & Zhu, 2021) and the top accuracy (ACC (C)) from the classifier for perceptual and content classifications, respectively.

4.3. Experimental results

4.3.1. Quantitative evaluation

Table 1 summarizes the FFG performance comparisons on the Korean and Chinese datasets. Our proposed method outperformed the baselines on most evaluation metrics. First, our proposed method outperformed all baselines based on pixel-level metrics (i.e., SSIM and MS-SSIM), except for the Chinese unseen font and unseen characters (UFUC) case, where LF-Font performed better. For the perceptual-level evaluation FID, MX-Font performed better on the Korean dataset for both seen and unseen characters, whereas our method performed better on seen and unseen Chinese characters. For LPIPS, the proposed method performed better on the Korean unseen fonts and seen characters (UFSC) and Chinese UFUC datasets. Our method also clearly outperformed the baselines in terms of content-aware metrics (Acc (C)). Our model achieved character classification accuracies of 96.19% and 98.28% for Korean and 96.27% and 97.16% for Chinese seen and unseen characters, respectively, showing that the proposed method preserves the content structure better than the baselines.

4.3.2. Qualitative evaluation

Fig. 3 depicts the qualitative comparisons on the Korean and Chinese datasets. All methods were evaluated on a variety of challenging fonts that varied in thickness, thinness, and serifs. In this experiment, a fixed source font is transformed into target font styles (illustrated in rows of ground truth (GT)) by all methods. We observe that all methods accurately generate the global font styles for both Korean and

[1] https://github.com/clovaai/fewshot-font-generation

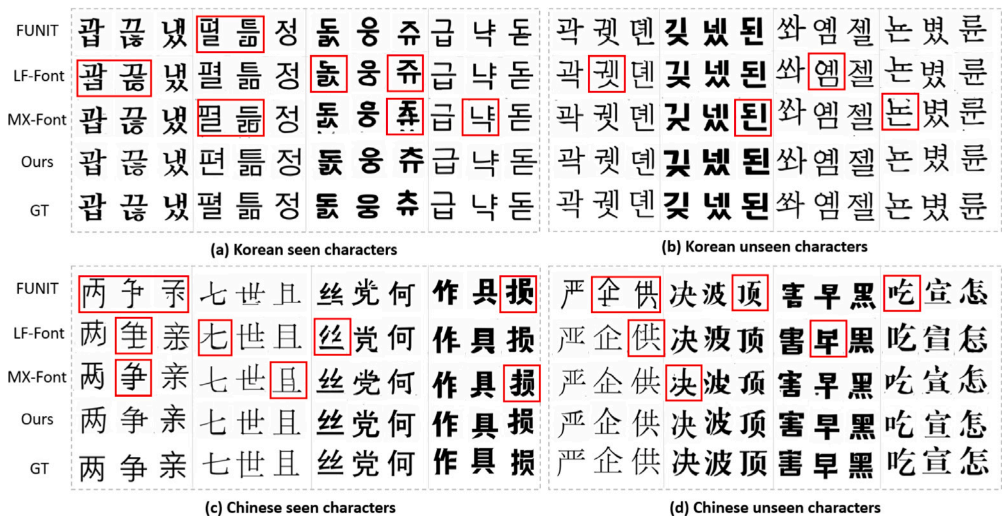

FUNIT
LF-Font
MX-Font
Ours
GT
(a) Korean seen characters
(b) Korean unseen characters
(c) Chinese seen characters
(d) Chinese unseen characters

Fig. 3. Visual comparisons of our proposed method with state-of-the-art few-shot font generation models: FUNIT (Li et al., 2021), LF-Font (Park et al., 2020), and MX-Font (Park et al., 2021) on Korean and Chinese fonts. GT represents the ground truth font glyph. All methods were evaluated in a 5-shot FG setting. Note that all presented results are based on unobserved font styles with seen or unseen characters for both languages. We highlight some of the failure cases in red boxes (zoom in for further details).

Table 2
Analyses of the proposed method. We report the quantitative comparisons on Korean unseen fonts and unseen characters. $wo_{triplet}$ refers to the setting in which we remove the Eqs. (1) and (5). $separate_{D\_senc}$ is the setting of the proposed model in which the discriminator and style encoder are trained separately. $L_{style\_recon\_G}$ refers to the setting where style reconstruction loss is substituted for generator style loss in Eq. (5).

Methods	LPIPS↓	ACC (C)↑	SSIM↑	MS-SSIM↑
Ours	0.1215	**98.28**	**0.625**	**0.413**
$wo_{triplet}$	0.1293	60.68	0.570	0.279
$separate_{D\_senc}$	**0.1051**	92.96	0.606	0.386
$L_{style\_recon\_G}$	0.1079	64.30	0.603	0.359

Chinese characters, however the generated glyphs of baseline methods are frequently broken or noisy. We highlight certain failure cases with red boxes in which the baseline methods failed to preserve the structure or transfer the style. Compared to the baseline methods, our proposed method successfully generates thick fonts, cursive fonts with strokes, and other fonts that are similar to the ground truth fonts in terms of global and local styles and content (structure). In Section 5.7 we present additional qualitative results on Chinese and Korean characters on few-shot (four reference characters) and one-shot settings.

5. Analysis of the proposed method

In this section, we analyze the key ideas of our proposed framework: First, introducing the triplet regularization for grouping each font style separately in the style embedding space. Second, employing the discriminator as a style encoder for guiding the generator during glyph generation. We also examine the effect of proposed style loss in our generator. We perform these ablations on Korean UFUC.

5.1. Effect of triplet regularization

For this experiment, we trained our proposed model without the triplet loss in Eqs. (1) and (5) to analyze the effect of it. Specifically, we removed the losses in Eqs. (1) and (5) from the proposed model which we name as $wo_{triplet}$. The results are depicted in Table 2 second row. Without triplet regularization in both G and D_{isc}, the proposed models character content accuracy (ACC (C)) drops significantly, indicating that the generated glyphs cannot preserve the content structure. In addition, the SSIM and MS-SSIM decrease.

5.2. Effect of integrating discriminator and style encoder

We trained our proposed model with a separate discriminator and style encoder in order to examine the impact of combining the two modules into a single module. For this experiment, we use the style encoder from FUNIT (Liu et al., 2019) and remove the style head h_s from the discriminator d_{isc}, naming this setting $separate_{D\_senc}$. The results are shown in the third row of Table 2. We observe a slight improvement in LPIPS in this setting, however drop in ACC (C), SSIM, and MS-SSIM owing to the effectiveness of the proposed integrated discriminator and style encoder architecture.

5.3. Effect of generator style loss

Previous methods have utilized style reconstruction loss objectives in generator to ensure that the generator utilizes style code from style encoder (Baek et al., 2021; Choi et al., 2020). This experiment is conducted to evaluate the effectiveness of the proposed style loss in Eq. (5). We replace our objective in Eq. (5) with the style reconstruction objective and name this setting $L_{style\_recon\_G}$ as depicted in fourth row of Table 2. Notably, the character content accuracy drops from 98.28% to 64.30%. In comparison to our proposed model with triplet loss in generator, SSIM and MS-SSIM also decrease.

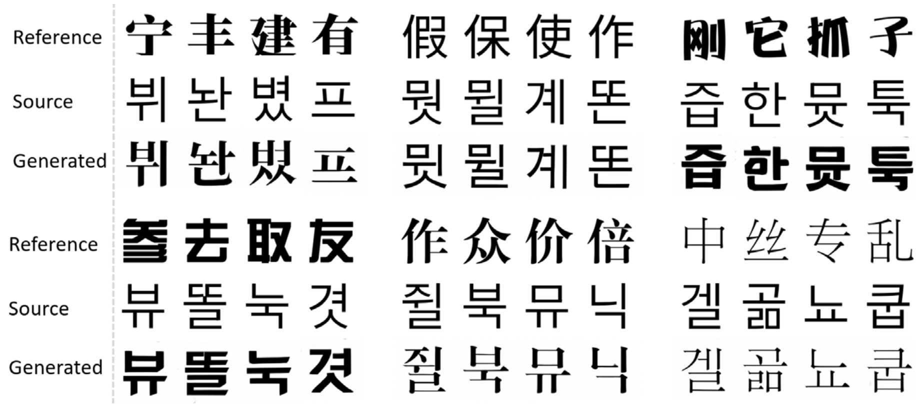
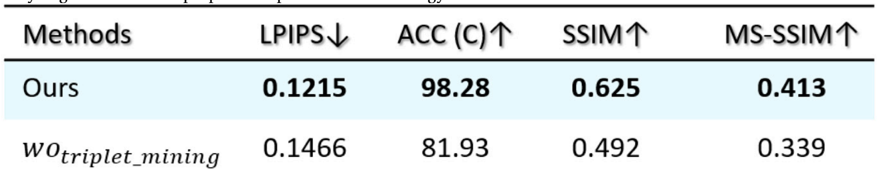


Fig. 4. Korean characters generated using our model trained on Chinese glyph images. The model successfully generates the source characters in the style of reference glyphs.

Table 3
Analyzing the choice of proposed triplet selection strategy.



Methods	LPIPS↓	ACC (C)↑	SSIM↑	MS-SSIM↑
Ours	**0.1215**	**98.28**	**0.625**	**0.413**
$wo_{triplet\_mining}$	0.1466	81.93	0.492	0.339

5.4. Cross-language evaluation

Can the network generate high-quality results on unseen language characters? To answer this question, we conducted this experiment. We trained our model on Chinese glyph images, however during inference, we fed the trained model with Korean Hangul characters, which were not seen by the model. As shown in Fig. 4, generated glyphs of our model for Korean characters are also effective. The network is able to transfer the style of Chinese reference glyphs on the unseen Korean source glyphs. This experiment demonstrates that our proposed model has powerful generalization ability, additionally the model is capable of disentangling the content and style of glyph images.

5.5. Effect of proposed triplet selection

Triplet selection is an important factor for fast convergence and stable training for triplet loss. In the paper (Schroff et al., 2015), the authors recommended to violate the triplet constraint (proposed Eqs. (1) and (5)). Specifically, selecting triplets so that the distance between the anchor and positive embeddings is larger than the distance between the anchor and negative embeddings. They utilized online triplet selection with a large batch size of 1000 for their experiments. However, due to the computational constraints, we were unable to adopt the online triplet selection strategy. We instead adopted to a simple yet very effective triplet selection strategy as discussed in Section 3.2.2. We choose the triplets such that the anchor and positive belong to the same font style and have different character content whereas the anchor and negative belong to different font styles but have the same character content. The intuition behind this selection is that the distance between the anchor and negative will be small as they have the same character content on the other hand, the difference between the anchor and positive will be large as they have different character contents even though they belong to the same font style.

In order to analyze the effect of the proposed choice of triplet selection, we trained a model without the proposed choice. Specifically, we selected negative images such that they have a random character content. Anchor and positive have same font class but different character content as before. We name this setting as $wo_{triplet\_mining}$ as depicted in Table 3 and Fig. 5. We observe that without the proposed triplet selection strategy, the objective metrics degrade. We demonstrate three problems in Fig. 5 (A), when the $wo_{triplet\_mining}$ setting fails to preserve the content also reflected in the ACC (C) in Table 3 (B), when the model generates additional strokes, and (C) when the model ignores the style code or follows the source character.

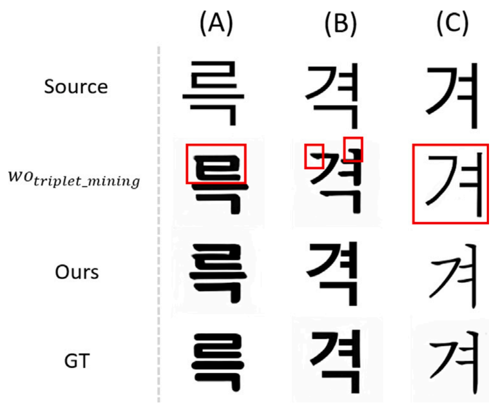


Fig. 5. Qualitative analysis of the proposed triplet selection strategy. $wo_{triplet\_mining}$ represents the setting where use random negative images. (A), (B), and (C) demonstrates some of the problems observed.

5.6. Analyzing learned style space of discriminator

We visualize the learned font style space of our proposed model and FUNIT (Liu et al., 2019) to evaluate the quality of the representation. Fig. 6 demonstrates the t-SNE maps trained on Korean dataset. For this experiment we used nine font styles highlighted with different colors as shown in Fig. 6(c). Our trained D_{isc} organizes the glyphs

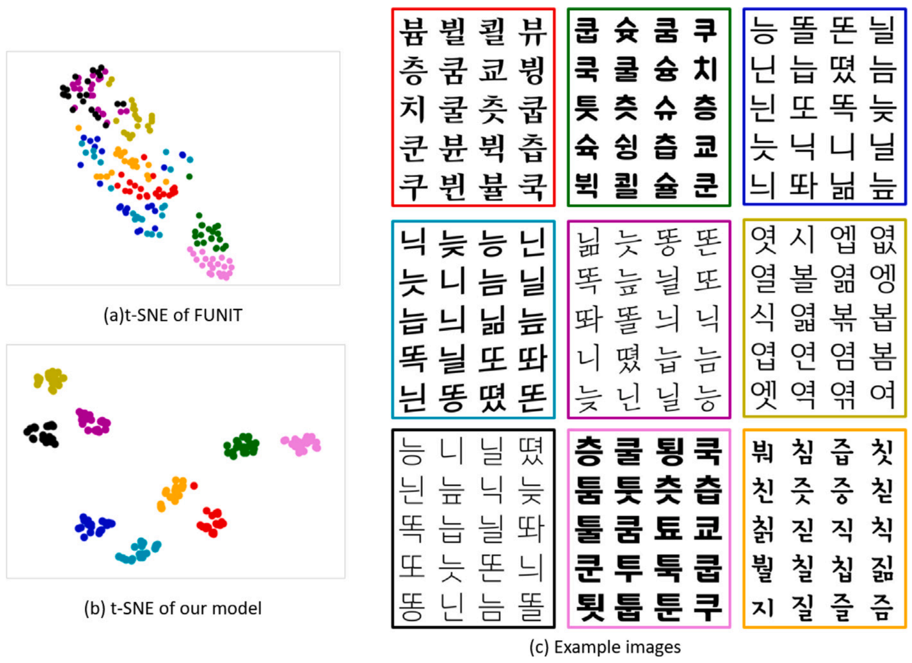


Fig. 6. t-SNE visualization of the learned font style space trained on Korean characters. 9 font styles were used with 21 characters each for this experiment. The colors correspond to the font styles in each cluster of the methods and corresponding ground truth images in (c).

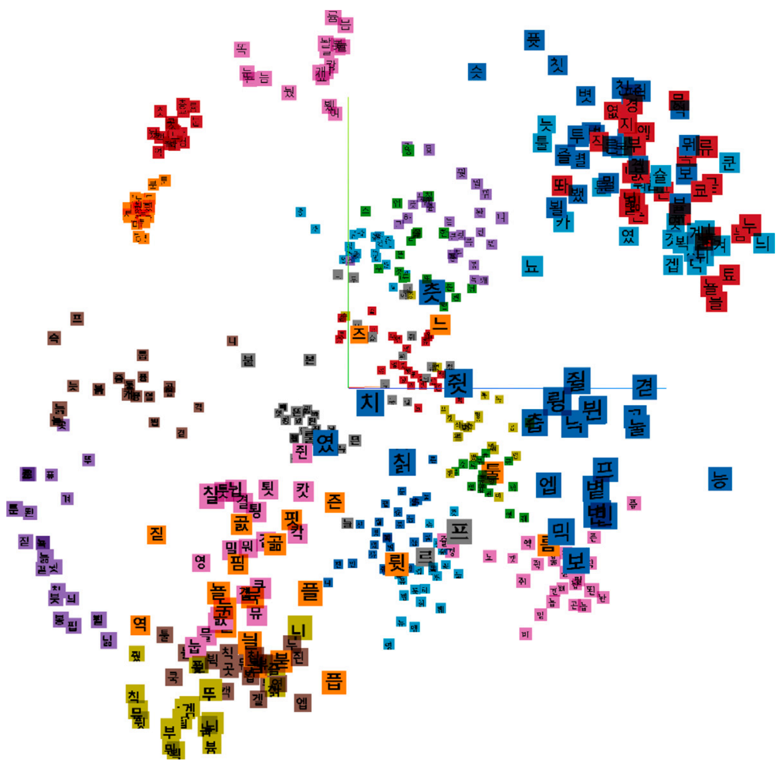


Fig. 7. 3-D representation of the style features z_s using PCA for 512 images across 24 font styles. Please zoom-in for details.

according to the their font styles and separates all images in nine clusters as shown in of Fig. 6(b). On the other hand, the style features of FUNIT are all entangled and hardly learn the meaningful fonts. The clusters are merged together hence the translation results are not proper as demonstrated earlier. From these results, we confirm that our proposed method can disentangle the style features of various font styles. Additionally, we also used PCA to visualize the style features z_s from style head h_s of our D_{isc} in a 3-D space. It can be seen from the Fig. 7 that characters of same font styles are mostly grouped together in font style embedding space.

5.7. Additional qualitative results

We provide additional glyph generation results on both Chinese and Korean datasets. Fig. 8 demonstrates results on various Chinese fonts with different styles in geometric transformation, stroke thickness, and

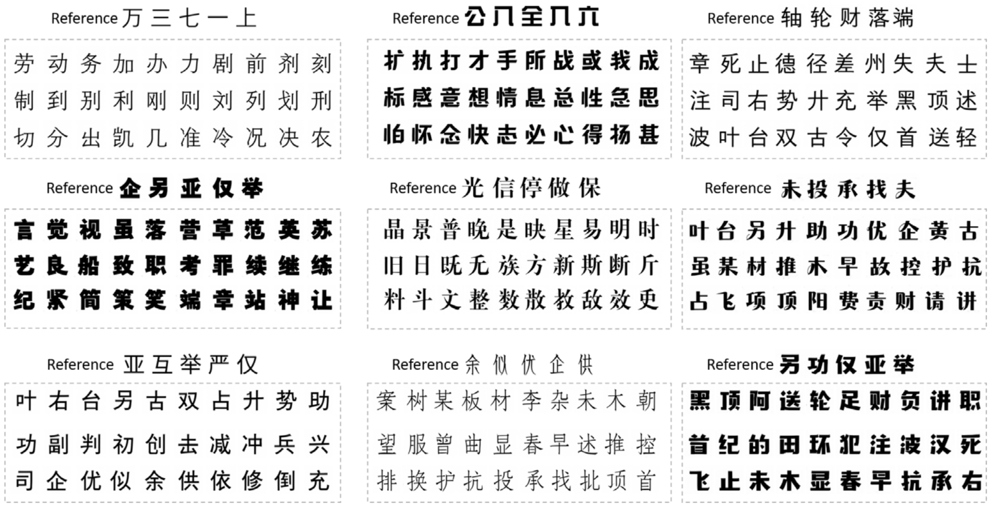


Fig. 8. Qualitative results on Chinese unseen four styles.

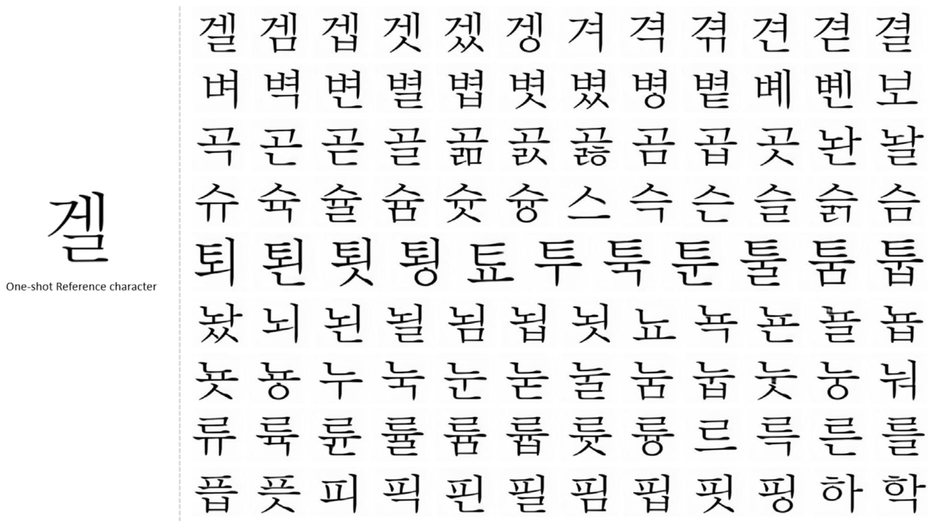


Fig. 9. Qualitative results on Korean characters. We generate all these characters in a one-shot setting i.e., with one reference style character.

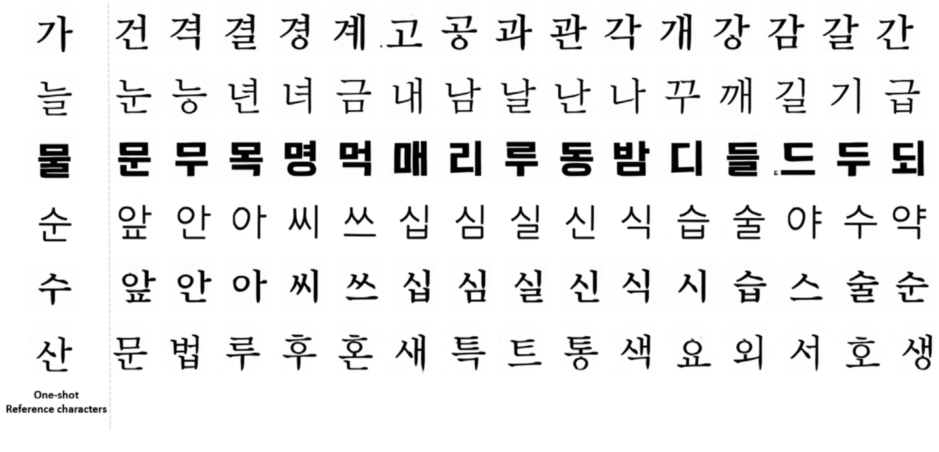


Fig. 10. Qualitative results on diverse Korean font styles and characters in one-shot setting.

tips, etc. From five reference characters, we generate thirty random characters and demonstrate the style consistency across the generated characters per reference style.

We also validated our model on one-shot setting, i.e., given just one reference character, the model generates font set with the style of reference character. For this experiment, we generated Korean hangul characters with different components as demonstrated in Fig. 9. Additionally, we demonstrate diverse Korean font styles and characters in one-shot setting in Fig. 10. The results demonstrate that the proposed method generates high-quality results.

6. Conclusion

In this paper, we introduced a few-shot font generation method by learning the font style space. To this end, we designed a style-guided discriminator that learns font style space via metric learning. We introduced a triplet loss to learn font style space in which distances directly correspond to a measure of font similarity. This sets it apart from other existing methods that learn the font style space using discrete font labels based on multitask discriminator. Additionally, we integrate our style encoder and discriminator in a single module and demonstrate its effectiveness. Both quantitative and qualitative comparisons on Korean and Chinese characters with existing state-of-the-art methods verify the effectiveness of our approach. **Limitation**. The model is limited in its ability to represent diverse local font styles as it only learns the global style representation. **Future work** will focus on local style representation for example, using the patch level similarity learning among fonts.

CRediT authorship contribution statement

Ammar Ul Hassan: Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing, Software, Data curation, Visualization. **Irfanullah Memon:** Conceptualization, Methodology, Methodology, Validation. **Jaeyoung Choi:** Supervision, Conceptualization, Methodology, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

This work was supported by National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-20230-00321688).

References

Azadi, S., Fisher, M., Kim, V. G., Wang, Z., Shechtman, E., & Darrell, T. (2018). Multi-content gan for few-shot font style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7564–7573).

Baek, K., Choi, Y., Uh, Y., Yoo, J., & Shim, H. (2021). Rethinking the truly unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 14154–14163).

Cha, J., Chun, S., Lee, G., Lee, B., Kim, S., & Lee, H. (2020). Few-shot compositional font generation with dual memory. In *Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, Part XIX 16* (pp. 735–751). Springer.

Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., & Choo, J. (2018). Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8789–8797).

Choi, Y., Uh, Y., Yoo, J., & Ha, J.-W. (2020). Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8188–8197).

Gao, Y., Guo, Y., Lian, Z., Tang, Y., & Xiao, J. (2019). Artistic glyph image synthesis via one-stage few-shot learning. *ACM Transactions on Graphics*, *38*(6), 1–12.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144.

Hassan, A. U., Ahmed, H., & Choi, J. (2021). Unpaired font family synthesis using conditional generative adversarial networks. *Knowledge-Based Systems*, *229*, Article 107304.

Hassan, A. U., Memon, I., & Choi, J. (2023). Real-time high quality font generation with conditional font GAN. *Expert Systems with Applications*, *213*, Article 118907. http://dx.doi.org/10.1016/j.eswa.2022.118907, URL https://www.sciencedirect.com/science/article/pii/S095741742201925X.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hermans, A., Beyer, L., & Leibe, B. (2017). In defense of the triplet loss for person re-identification, CoRR abs/1703.07737. URL http://arxiv.org/abs/1703.07737.

Huang, X., & Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision* (pp. 1501–1510).

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1125–1134).

Jiang, Y., Lian, Z., Tang, Y., & Xiao, J. (2017). Dcfont: an end-to-end deep chinese font generation system. In *SIGGRAPH Asia 2017 technical briefs* (pp. 1–4).

Jiang, Y., Lian, Z., Tang, Y., & Xiao, J. (2019). Scfont: Structure-guided chinese font generation via deep stacked networks. In *Proceedings of the AAAI conference on artificial intelligence, vol. 33, no. 01* (pp. 4015–4022).

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8110–8119).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Ko, D. H., Hassan, A. U., Suk, J., & Choi, J. (2021). SKFont: skeleton-driven Korean font generator with conditional deep adversarial networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 1–13.

Kong, Y., Luo, C., Ma, W., Zhu, Q., Zhu, S., Yuan, N., et al. (2022). Look closer to supervise better: one-shot font generation via component-based discriminator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 13482–13491).

Li, C., Taniguchi, Y., Lu, M., & Konomi, S. (2021). Few-shot font style transfer between different languages. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (pp. 433–442).

Liu, M.-Y., Breuel, T., & Kautz, J. (2017). Unsupervised image-to-image translation networks. *Advances in Neural Information Processing Systems*, *30*.

Liu, M.-Y., Huang, X., Mallya, A., Karras, T., Aila, T., Lehtinen, J., et al. (2019). Few-shot unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 10551–10560).

Liu, W., Liu, F., Ding, F., He, Q., & Yi, Z. (2022). Xmp-font: self-supervised cross-modality pre-training for few-shot font generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7905–7914).

Maas, A. L. (2013). Rectifier nonlinearities improve neural network acoustic models.

Mescheder, L., Geiger, A., & Nowozin, S. (2018). Which training methods for GANs do actually converge? In *International conference on machine learning* (pp. 3481–3490). PMLR.

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

Muhammad, A. U. H., Lee, H., & Choi, J. (2023). Exploiting mixing regularization for truly unsupervised font synthesis. *Pattern Recognition Letters*, *169*, 35–42. http://dx.doi.org/10.1016/j.patrec.2023.03.019, URL https://www.sciencedirect.com/science/article/pii/S0167865523000843.

Odena, A., Olah, C., & Shlens, J. (2017). Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning* (pp. 2642–2651). PMLR.

Park, S., Chun, S., Cha, J., Lee, B., & Shim, H. (2020). Few-shot font generation with localized style representations and factorization. arXiv preprint arXiv:2009.11042.

Park, S., Chun, S., Cha, J., Lee, B., & Shim, H. (2021). Multiple heads are better than one: Few-shot font generation with multiple localized experts. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 13900–13909).

Parmar, G., Zhang, R., & Zhu, J.-Y. (2021). On buggy resizing libraries and surprising subtleties in FID calculation. arXiv preprint arXiv:2104.11222.

Saito, K., Saenko, K., & Liu, M.-Y. (2020). Coco-funit: Few-shot unsupervised image translation with a content conditioned style encoder. In *European conference on computer vision* (pp. 382–398). Springer.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. (pp. 815–823).

Tang, L., Cai, Y., Liu, J., Hong, Z., Gong, M., Fan, M., et al. (2022). Few-shot font generation by learning fine-grained local styles. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7895–7904).

Tian, Y. (2017). zi2zi: Master Chinese calligraphy with conditional adversarial networks. https://github.com/kaonashi-tyc/zi2zi.

Xie, Y., Chen, X., Sun, L., & Lu, Y. (2021). DG-font: Deformable generative networks for unsupervised font generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5130–5140).

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., & Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 586–595).

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2223–2232).

Zhu, J.-Y., Zhang, R., Pathak, D., Darrell, T., Efros, A. A., Wang, O., et al. (2017). Toward multimodal image-to-image translation. *Advances in Neural Information Processing Systems*, *30*.