Few-shot Compositional Font Generation with Dual Memory

Junbum Cha, Sanghyuk Chun, Gayoung Lee,
Bado Lee, Seonghyeon Kim, and Hwalsuk Lee

Clova AI Research, NAVER Corp.
{junbum.cha, sanghyuk.c, gayoung.lee,
bado.lee, kim.seonghyeon, hwalsuk.lee}@navercorp.com

Abstract. Generating a new font library is a very labor-intensive and time-consuming job for glyph-rich scripts. Despite the remarkable success of existing font generation methods, they have significant drawbacks; they require a large number of reference images to generate a new font set, or they fail to capture detailed styles with only a few samples. In this paper, we focus on compositional scripts, a widely used letter system in the world, where each glyph can be decomposed by several components. By utilizing the compositionality of compositional scripts, we propose a novel font generation framework, named Dual Memory-augmented Font Generation Network (DM-Font), which enables us to generate a high-quality font library with only a few samples. We employ memory components and global-context awareness in the generator to take advantage of the compositionality. In the experiments on Korean-handwriting fonts and Thai-printing fonts, we observe that our method generates a significantly better quality of samples with faithful stylization compared to the state-of-the-art generation methods quantitatively and qualitatively. Source code is available at `https://github.com/clovaai/dmfont`.

1 Introduction

Advances of web technology lead people to consume a massive amount of texts on the web. It makes designing a new font style, *e.g.*, personalized handwriting, critical. However, because traditional methods to make a font library heavily rely on expert designers by manually designing each glyph, creating a font library is extremely expensive and labor-intensive for glyph-rich scripts such as Chinese (more than 50,000 glyphs), Korean (11,172 glyphs), or Thai (11,088 glyphs) [11].

Recently, end-to-end font generation methods [1,16,17,26,5,4] have been proposed to build a font set without human experts. The methods solve image-to-image translation tasks between various font styles based on generative adversarial networks (GANs) [10]. While the methods have shown the remarkable achievement, they still require a large number of samples, *e.g.*, 775 samples [16,17] to generate a new font set. Moreover, they require additional training to create a new glyph set, *i.e.*, they need to finetune the pretrained model on the given

Fig. 1: **Few-shot font generation results.** While previous few-shot font generation methods (AGIS [8], FUNIT [23], and EMD [35]) are failed to generate unseen font, our model successfully transfer the font style and details.

new glyph subset. Thus, these finetune-based methods are rarely practical if collecting the target glyphs is extremely expensive, *e.g.*, human handwriting.

Several recent studies attempt to generate a font set without additional training with a large number of glyphs, but using only a few samples [2,31,35,8,30]. Despite their successful few-shot generation performances on training styles, existing few-shot font generation methods often fail to generate high-quality font library with unseen style few-shot samples as illustrated in Figure 1. We solve this problem using the inherent glyph characteristics in contrast to most of the previous works handling the problem in the end-to-end data-driven manner without any human prior. A few researchers have considered characteristics of glyphs to improve font generation methods [31,17], but their approaches are either still requiring more than 700 samples [17], or only designed for memory efficiency [31].

In this paper, we focus on a famous family of scripts, called *compositional scripts*, which are composed of a combination of sub-glyphs or components. For example, the Korean script has 11,172 valid glyphs with only 68 components. One can build a full font library by designing only 68 sub-glyphs and combine them by the pre-defined rule. However, this rule-based method has a significant limitation; a sub-glyph changes its shape and position diversely depending on the combination, as shown in Figure 2. Hence, even if a user has a complete sub-glyphs, generating a full font set is impossible without the combination rule of components. Due to the limitations, compositional scripts have been manually designed for each glyph despite its compositionality [11].

Our framework for the few-shot font generation, Dual Memory-augmented Font Generation Network (DM-Font), utilizes the compositionality supervision in the weakly-supervised manner, *i.e.*, no component-wise bounding box or mask is required but only component labels are required, resulting on more efficient and effective generation. We employ the dual memory structure (*persistent memory* and *dynamic memory*) to efficiently capture the global glyph structure and the local component-wise styles, respectively. This strategy enables us to generate a new high-quality font library with only a few samples, *e.g.*, 28 samples and 44 samples for Korean and Thai, respectively. In the experiments, the generated Korean and Thai fonts show both quantitatively better visual quality in various metrics and qualitatively being preferred in the user study.

2 Related Works

2.1 Few-shot image-to-image translation

Image-to-image (I2I) translation [15,36,6,19,7] aims to learn the mapping between different domains. This mapping preserves the content in the source domain while changing the style as the target domain. Mainstream I2I translation methods assume an abundance of target training samples which is impractical. To deal with more realistic scenarios where the target samples are scarce, few-shot I2I translation works appeared recently [23]. These methods can be directly applied to the font generation task as a translation task between the reference font and target font. We compare our method with FUNIT [23].

As an independent line of research, style transfer methods [9,20,14,25,21,32] have been proposed to transfer styles of an unseen reference while preserving the original content. Unlike I2I translation tasks, style transfer methods cannot be directly transformed to font generation tasks, because they usually define the style as the set of textures and colors. However, in font generation tasks, the style of font is usually defined as discriminative local property of the font. Hence, our work does not concern style transfer methods as our baseline.

2.2 Automatic font generation

Automatic font generation task is an I2I translation between different font domains, *i.e.*, styles. We categorize the automatic font generation methods into two classes, which are many-shot and few-shot methods, according to way to generate a new font set. Many-shot methods [1,16,26,5,4,17] directly finetune the model on the target font set with a large number of samples, *e.g.*, 775. It is impractical in many real-world scenarios when collecting new glyphs is costly, *e.g.*, handwriting.

In contrast, few-shot font generation methods [35,30,2,8,31] does not require additional finetuning and a large number of reference images. However, the existing few-shot methods have significant drawbacks. For example, some methods generate a whole font set at single forward path [2,30]. Hence, they require a huge model capacity and cannot be applied to glyph-rich scripts but scripts with only a few glyphs, *e.g.*, Latin alphabet. On the other hand, EMD [35] and AGIS-Net [8] can be applied to any general scripts, but they show worse synthesizing quality to unseen style fonts, as observed in our experimental results. SA-VAE [31], a Chinese-specific method, keeps the model size small by compressing one-hot character-wise embeddings based on compositionality of Chinese script. Compared with SA-VAE, ours handles the features as component-wise, not character-wise. It brings huge advantages in not only reducing feature dimension but also in performances as shown in our experimental results.

3 Preliminary: Complete Compositional Scripts

Compositional script is a widely-used glyph-rich script, where each glyph can be decomposed by several components as shown in Fig. 2. These scripts account

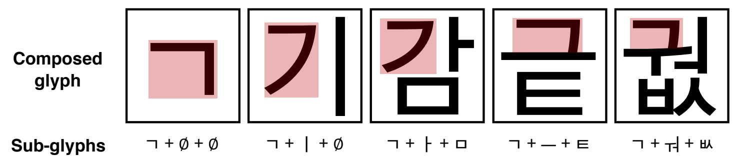


Fig. 2: **Examples of compositionality of Korean script.** Even if we choose the same sub-glyph, *e.g.*, "ㄱ", the shape and position of each sub-glyph are varying depending on the combination, as shown in red boxes.

for 24 of the top 30 popular scripts, including Chinese, Hindi, Arabic, Korean, Thai. A compositional script is either *complete* or not, where each glyph in *complete compositional scripts* can be decomposed to fixed number sub-glyphs. For example, every Korean glyph can be decomposed by three sub-glyphs (See Fig. 2). Similarly, a Thai character has four components. Furthermore, complete compositional letters have specific sub-glyph sets for each *component type*. For example, the Korean alphabet has three component types where each component type has 19, 21, 28 sub-glyphs. By combining them, Korean letter has $19 \times 21 \times 28 = 11,172$ valid characters. Note that the minimum number of glyphs to get the entire sub-glyph set is 28. Similarly, Thai letter can represent $44 \times 7 \times 9 \times 4 = 11,088$ characters, and 44 characters are required to cover whole sub-glyphs.

Some compositional scripts are not complete. For example, each character of the Chinese letter can be decomposed into a diverse number of sub-glyphs. Although we mainly validate our method on Korean and Thai scripts, our method can be easily extended to other compositional scripts.

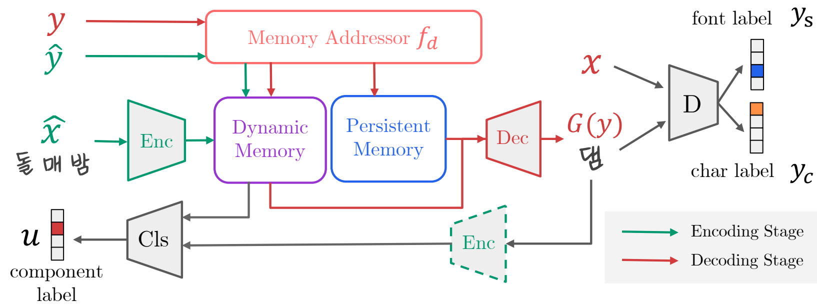
4 Dual Memory-augmented Font Generation Network

In this section, we introduce a novel architecture, Dual Memory-augmented Font Generation Network (DM-Font), which utilizes the compositionality of a script by the augmented dual memory structure. DM-Font disentangles global composition information and local styles, and writes them into persistent and dynamic memory, respectively. It enables to make a high-quality full glyph library only with very few references, *e.g.*, 28 samples for Korean, 44 samples for Thai.

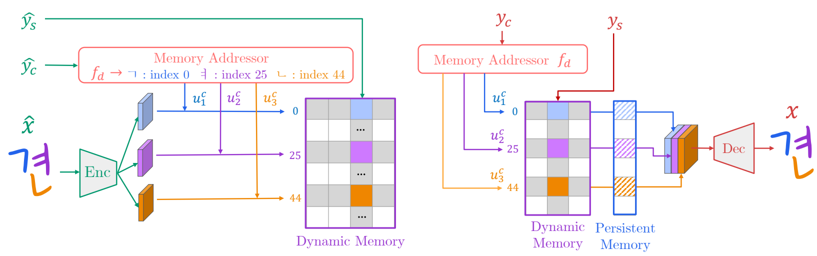
4.1 Architecture overview

We illustrate the architecture overview of DM-Font in Fig. 3a. The generation process consists of encoding and decoding stages. In the encoding stage, the reference style glyphs are encoded to the component features and stored into the dynamic memory. After the encoding, the decoder fetches the component features and generates the target glyph according to the target character label.

Encoder *Enc* disassembles a source glyph into the several component features using the pre-defined decomposition function. We adopt multi-head struc-



(a) Architecture overview.



(b) Encoding phase detail. (c) Decoding phase detail.

Fig. 3: **DM-Font overview.** (a) The model encodes the reference style glyphs and stores the component-wise features into the memory – (b). The decoder generates images with the component-wise features – (c). (b) The encoder extracts the component-wise features and stores them into the dynamic memory using the component label u_i^c and the style label $\hat{y}_s$. (c) The memory addressor loads the component features by the character label y_c and feeds them to the decoder.

ture, one head per one component type. The encoded component-wise features are written into the dynamic memory as shown in Figure 3b.

We employ two memory modules, where **persistent memory** (PM) is a component-wise learned embedding that represents the intrinsic shape of each component and the global information of the script such as the compositionality, while **dynamic memory** (DM) stores encoded component features of the given reference glyphs. Hence, PM captures the global information of sub-glyphs independent to each font style, while encoded features in DM learn unique local styles depending on each font. Note that DM simply stores and retrieves the encoded features, but PM is learned embedding trained from the data. Therefore, DM is adaptive to the reference input style samples, while PM is fixed after training. We provide detailed analysis of each memory in the experiments.

Memory addressor provides the access address of both dynamic and persistent memory based on the given character label y_c as shown in Figure 3b and Figure 3c. We use pre-defined decomposition function $f_d : y_c \mapsto \{u_i^c \mid i = 1 \dots M_c\}$ to get the component-wise address, where u_i^c is the label of i-th component of y_c, and M_c is the number of sub-glyphs for y_c. For example, the function decomposes a Korean character, "한" by f_d("한") = {"ㅎ", "ㅏ", "ㄴ"}. The function maps input character to Unicode and decomposes it by a simple rule. More details of the decomposition function are given in Appendix.

The component-wise encoded features for the reference $\hat{x}$, whose character label is $\hat{y}_c$ and style label is $\hat{y}_s$, are stored into DM during the encoding stage. In our scenario, the encoder Enc is a multi-head encoder, and $\hat{y}_c$ can be decomposed by $f_d(\hat{y}_c)$ to sub-glyph labels $\hat{u}_i^c$. Hence, the features in DM at address $(\hat{u}_i^c, \hat{y}_s)$, $DM(\hat{u}_i^c, \hat{y}_s)$ is computed by $Enc_i(\hat{x})$, where i is the index of the component type and Enc_i is the encoder output corresponding to i.

In the decoding stage, **decoder** Dec generates a target glyph with the target character y_c and the reference style y_s using the component-wise features stored into the dynamic memory DM and the persistent memory PM as the following:

$$G(y_c, y_s) = Dec\big(\big[DM(u_i^c, y_s),\, PM(u_i^c) \mid u_i^c \in f_d(y_c)\big]\big), \tag{1}$$

where $[x_0, \ldots, x_n]$ refers to the concatenation operation.

For the better generation quality, we also employ a discriminator and a component classifier. For **discriminator** D, we use a multitask discriminator [27,23] with the font condition and the character condition. The multitask discriminator has independent branches for each target class and each branch performs binary classification. Considering two types of conditions, we use two multitask discriminator, one for character classes and the other for font classes, with a shared backbone. We further use **component classifier** Cls to ensure the model to fully utilize the compositionality. The component classifier provides additional supervision to the generator that stabilizes the training.

Moreover, we introduce the global-context awareness and local-style preservation to the generator, called **compositional generator**. Specifically, self-attention blocks [3,33] are used in the encoder to facilitate relational reasoning between components, and the hourglass block [29,22] is attached to the decoder to aware global-context while preserving locality. In the experiment section, we analyze the impact of the architectural improvements on the final performance. We provide the architecture and the implementation details in Appendix.

DM-Font learns the compositionality in the weakly-supervised manner; it does not require any exact component location, *e.g.*, component-wise bounding boxes, but only component labels are required. Hence, DM-Font is not restricted to the font generation only, but can be applied to any generation task with compositionality, *e.g.*, attribute conditioned generation tasks. Extending DM-Font to attribute labeled datasets, *e.g.*, CelebA [24], will be an interesting topic.

4.2 Learning

We train DM-Font from font sets $(x, y_c, y_f) \sim \mathcal{D}$, where x is a target glyph image, y_c and y_f is a character and font label, respectively. During the training, we assume that different font labels represent different styles, *i.e.*, we set $y_s = y_f$ in equation (1). Also, for the efficiency, we only encode a core component subset to compose the target glyph x into the DM instead of the full component set. For example, the Korean script has the full component set with size 68, but only 3 components are required to construct a single character.

We use **adversarial loss** to let the model generate plausible images.

$$\mathcal{L}_{adv} = \mathbb{E}_{x,y}\left[\log D_y(x)\right] + \mathbb{E}_{x,y}\left[\log(1 - D_y(G(y_c, y_f)))\right], \tag{2}$$

where G generates an image $G(y_c, y_f)$ from the given image x and target label y by equation (1). The discriminator D_y is conditional on the target label y. We employed two types of the discriminator to solve the problem. The font discriminator is a conditional discriminator on the source font index and the character discriminator aims to classify what is the given character.

$\mathbf{L_1}$ **loss** adds supervision from the ground truth target x as the following:

$$\mathcal{L}_{l1} = \mathbb{E}_{x,y}\left[\|x - G(y_c, y_f)\|_1\right]. \tag{3}$$

We also use **feature matching loss** to improve the stability of the training. The feature matching loss is constructed using the output from the l-th layer of the L-layered discriminator, $D_f^{(l)}$.

$$\mathcal{L}_{feat} = \mathbb{E}_{x,y}\left[\frac{1}{L}\sum_{l=1}^{L}\|D_f^{(l)}(x) - D_f^{(l)}(G(y_c, y_f))\|_1\right]. \tag{4}$$

Lastly, to let the model fully utilize the compositionality, we train the model with additional **component-classification loss**. For the given input x, we extract the component-wise features using the encoder Enc, and train them with cross-entropy loss (CE) using component labels $u \in f_d(y_c)$, where f_d is the component decomposition function to the given character label y_c.

$$\mathcal{L}_{cls} = \mathbb{E}_{x,y}\left[\sum_{u_i^c \in f_d(y_c)} \mathrm{CE}(Enc_i(x), u_i^c)\right] + \mathbb{E}_y\left[\sum_{u_i^c \in f_d(y_c)} \mathrm{CE}(Enc_i(G(y_c, y_f)), u_i^c)\right]. \tag{5}$$

The final objective function to optimize the generator G, the discriminator D, and the component classifier C is defined as the following:

$$\min_{G,C}\max_{D} \mathcal{L}_{adv(font)} + \mathcal{L}_{adv(char)} + \lambda_{l1}\mathcal{L}_{l1} + \lambda_{feat}\mathcal{L}_{feat} + \lambda_{cls}\mathcal{L}_{cls}, \tag{6}$$

where $\lambda_{l1}, \lambda_{feat}, \lambda_{cls}$ are control parameters to importance of each loss function. We set $\lambda_{l1} = 0.1, \lambda_{feat} = 1.0, \lambda_{cls} = 0.1$ for all experiments.

5 Experiments

5.1 Datasets

Korean-handwriting dataset. Due to its diversity and data sparsity, generating a handwritten font with only a few samples is challenging. We validate the models using 86 Korean-handwriting fonts[1] refined by the expert designer. Each font library contains $2,448$ widely-used Korean glyphs. We train the models using 80% fonts and 90% characters, and validate the models on the remaining split. We separately evaluate the models on the seen (90%) and unseen (10%) characters to measure the generalizability to the unseen characters. 30 characters are used for the reference.

Thai-printing dataset. Compared with Korean letters, Thai letters have more complex structure; Thai characters are composed of four sub-glyphs while Korean characters have three components. We demonstrate the models on 105 Thai-printing fonts[2]. The train-evaluation split strategy is same as Korean-handwriting experiments, and 44 samples are used for the few-shot generation.

Korean-unrefined dataset. We also gather unrefined Korean handwriting dataset from 88 non-experts, letting each applicant write 150 characters. This dataset is extremely diverse and not refined by expert designers different from the Korean-handwriting dataset. We use the Korean-unrefined dataset as the validation of the models trained on the Korean-handwriting dataset, *i.e.*, the Korean-unrefined dataset is not visible during the training, but only a few samples are visible for the evaluation. 30 samples are used for the generation as well as the Korean-handwriting dataset.

5.2 Comparison methods and evaluation metrics

Comparison methods. We compare our model with state-of-the-art few-shot font generation methods, including EMD [35], AGIS-Net [8], and FUNIT [23]. We exclude the methods which are Chinese-specific [31] or not applicable to glyph-rich scripts [30]. Here, we slightly modified FUNIT, originally designed for unsupervised translation, by changing its reconstruction loss to L_1 loss with ground truths and conditioning the discriminator to both contents and styles.

Evaluation metrics. Assessing a generative model is difficult because of its non-tractability. Several quantitative evaluation metrics [18,13,34,28] have attempted to measure the performance of the trained generative model with different assumptions, but it is still controversial what is the best evaluation methods for generative models. In this paper, we consider three diverse levels of evaluation metrics; pixel-level, perceptual-level and human-level evaluations.

Pixel-level evaluation metrics assess the pixel structural similarity between the ground truth image and the generated image. We employ the structural similarity index (SSIM) and multi-scale structural similarity index (MS-SSIM).

However, pixel-level metrics often disagree with human perceptions. Thus, we also evaluate the models with **perceptual-level evaluation metrics**. We

[1] We collect public fonts from `http://uhbeefont.com/`.

[2] `https://github.com/jeffmcneill/thai-font-collection`.

Table 1: **Quantatitive evaluation on the Korean-handwriting dataset.** We evaluate the methods on the seen and unseen character sets. Higher is better, except perceptual distance (PD) and mFID.

	Pixel-level		Content-aware			Style-aware		
	SSIM	MS-SSIM	Acc(%)	PD	mFID	Acc(%)	PD	mFID
Evaluation on the **seen** character set during training								
EMD [35]	0.691	0.361	80.4	0.084	138.2	5.1	0.089	134.4
FUNIT [23]	0.686	0.369	94.5	0.030	42.9	5.1	0.087	146.7
AGIS-Net [8]	0.694	0.399	**98.7**	**0.018**	23.9	8.2	0.088	141.1
DM-Font (ours)	**0.704**	**0.457**	98.1	**0.018**	**22.1**	**64.1**	**0.038**	**34.6**
Evaluation on the **unseen** character set during training								
EMD [35]	0.696	0.362	76.4	0.095	155.3	5.2	0.089	139.6
FUNIT [23]	0.690	0.372	93.3	0.034	48.4	5.6	0.087	149.5
AGIS-Net [8]	0.699	0.398	98.3	0.019	25.9	7.5	0.089	146.1
DM-Font (ours)	**0.707**	**0.455**	**98.5**	**0.018**	**20.8**	**62.6**	**0.039**	**40.5**

trained four ResNet-50 [12] models on the Korean-handwriting dataset and Thai-printing dataset to classify style and character label. Unlike the generation task, the whole fonts and characters are used for the training. More detailed classifier training settings are in Appendix. We denote a metric is *context-aware* if the metric is performed using the content classifier, and *style-aware* is defined similarly. Note that these classifiers are independent to the font generation models, but only used for the evaluation. We report the top-1 accuracy, perceptual distance (PD) [18,34], and mean FID (mFID) [23] using the classifiers. PD is computed by L_2 distance of the features between generated glyph and GT glyph, and mFID is a conditional FID [13] by averaging FID for each target class.

Finally, we conduct a user study on the Korean-unrefined dataset for measuring **human-level evaluation metric**. We ask users about three types of preference: content preference, style preference, and user preference considering both content and style. The questionnaire is made of 90 questions, 30 for each preference. Each question shows 40 glyphs, consisting of 32 glyphs generated by four models and 8 GT glyphs. The order of choices is shuffled for anonymity. We collect total 3,420 responses from 38 Korean natives. More details of user study are provided in Appendix.

5.3 Main results

Quantitative evaluation. The main results on Korean-handwriting and Thai-printing datasets are reported in Table 1 and Table 2, respectively. We also report the evaluation results on the Korean-unrefined dataset in Appendix. We follow the dataset split introduced in Section 5.1. In the experiments, DM-Font remarkably outperforms the comparison methods in most of evaluation metrics, especially on style-aware benchmarks. Baseline methods show slightly worse content-

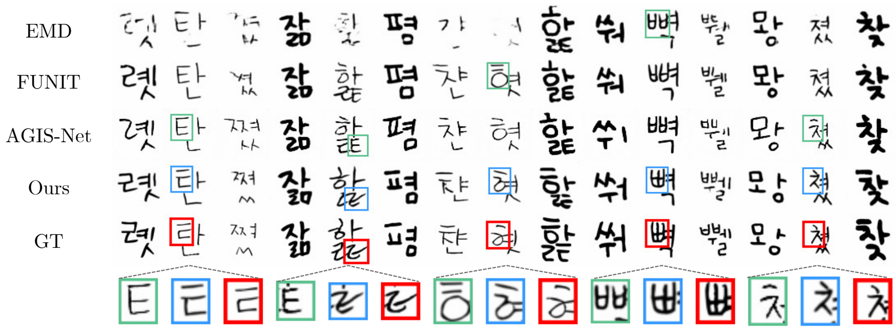
Table 2: **Quantatitive evaluation on the Thai-printing dataset.** We evaluate the methods on the seen and unseen character sets. Higher is better, except perceptual distance (PD) and mFID.

	Pixel-level		Content-aware			Style-aware		
	SSIM	MS-SSIM	Acc(%)	PD	mFID	Acc(%)	PD	mFID
Evaluation on the **seen** character set during training								
EMD [35]	0.773	0.640	86.3	0.115	215.4	3.2	0.087	172.0
FUNIT [23]	0.712	0.449	45.8	0.566	1133.8	4.6	0.084	167.9
AGIS-Net [8]	0.758	0.624	**87.2**	**0.091**	**165.2**	15.5	0.074	145.2
DM-Font (ours)	**0.776**	**0.697**	87.0	0.103	198.7	**50.3**	**0.037**	**69.4**
Evaluation on the **unseen** character set during training								
EMD [35]	0.770	0.636	85.0	0.123	231.0	3.4	0.087	171.6
FUNIT [23]	0.708	0.442	45.0	0.574	1149.8	4.7	0.084	166.9
AGIS-Net [8]	0.755	0.618	85.4	0.103	**188.4**	15.8	0.074	145.1
DM-Font (ours)	**0.773**	**0.693**	**87.2**	**0.101**	195.9	**50.6**	**0.037**	**69.6**

aware performances on unseen characters than seen characters, *e.g.*, AGIS-Net shows worse content-aware accuracy (98.7 → 98.3), PD (0.018 → 0.019), and mFID (23.9 → 25.9) in Table 1. In contrast, DM-Font consistently shows better generalizability to the unobserved characters during the training for both datasets. It is because our model interprets a glyph at the component level, the model easily extrapolates the unseen characters from the learned component-wise features stored in memory modules.

Our method shows significant improvements in style-aware metrics. DM-Font achieves 62.6% and 50.6% accuracy while other methods show much less accuracy, *e.g.*, about 5% for Korean unseen and Thai unseen character sets, respectively. Likewise, the model shows dramatic improvements in perceptual distance and mFID as well as the accuracy measure. In the latter section, we provide more detailed analysis that the baseline methods are overfitted to the training styles and failed to generalize to unseen styles.

Qualitative comparison. We also provide visual comparisons in Figure 4 and Figure 5, which contain various challenging fonts including thin, thick, and curvy fonts. Our method generates glyphs with consistently better visual quality than the baseline methods. EMD [35] often erases thin fonts unintentionally, which causes low content scores compared with the other baseline methods. FUNIT [23] and AGIS-Net [8] accurately generate the content of glyphs and capture global styles well including overall thickness and font sizes. However, the detailed styles of the components in their results look different from the ground truths. Moreover, some generated glyphs for unseen Thai style lose the original content (see the difference between green boxes and red boxes in Figure 4 and Figure 5 for more details). Compared with the baselines, our method generates



(a) Seen character set during training.

(b) Unseen character set during training.

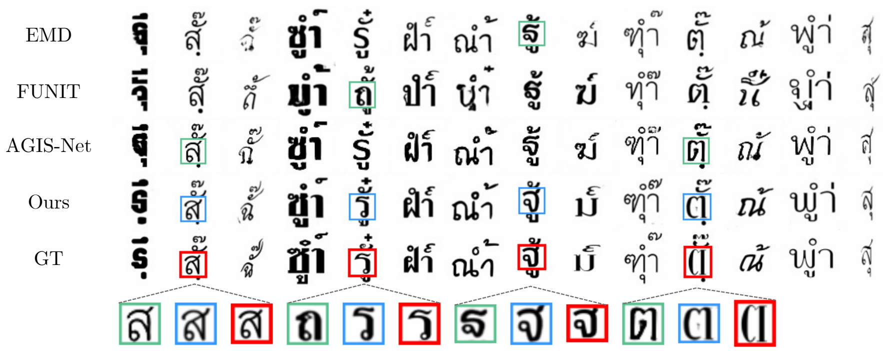
Fig. 4: **Qualitative comparison on the Korean-handwriting dataset.** Visualization of generated samples with seen and unseen characters. We show insets of baseline results (green box), ours (blue box) and ground truth (red box). Ours successfully transfers the detailed style of the target style, while baselines fail to generate glyphs with the detailed reference style.

the most plausible images in terms of global font styles and detailed component styles. These results show that our model preserves details in the components using the dual memory and reuse them to generate a new glyph.

User study. We conduct a user study to further evaluate the methods in terms of human preferences using the Korean-unrefined dataset. Example generated glyphs are illustrated in Figure 6. Users are asked to choose the most preferred generated samples in terms of content preservation, faithfulness to the reference style, and personal preference. The results are shown in Table 3, which present similar intuitions with Table 1; AGIS-Net and our method are comparable in the content evaluation, and our method is dominant in the style preference.

5.4 More analysis

(a) Seen character set during training.



(b) Unseen character set during training.

Fig. 5: **Qualitative comparison on the Thai-printing dataset.** Visualization of generated samples with seen and unseen characters. We show insets of baseline results (green box), ours (blue box) and ground truth (red box). Overall, ours faithfully transfer the target style, while other methods even often fail to preserve contents in unseen character sets.

Ablation study. We investigate the impact of our design choices by ablative studies. Table 4a shows that the overall performances are improved by adding proposed components such as dynamic memory, persistent memory, and compositional generator. We report full table in Appendix.

Here, the baseline method is similar to FUNIT whose content and style accuracies are 93.9 and 5.4, respectively. The baseline suffers from the failure of style generalization as previous methods. We observe that dynamic memory and persistent memory dramatically improves style scores while preserving content scores. Finally, our architectural improvements bring the best performance.

We also explore the performance influence of each objective. As shown in Table 4b, removing L_1 loss and feature matching loss slightly degrades performances. The component-classification loss, which enforces the compositionality to the model, is the most important factor for successful training.

Table 3: **User study results on the Korean-unrefined dataset.** Each number is the preferred model output out of 3,420 responses.

	EMD [35]	FUNIT [23]	AGIS-Net [8]	DM-Font (ours)
Best content preserving	1.33%	9.17%	**48.67%**	40.83%
Best stylization	1.71%	8.14%	17.44%	**72.71%**
Most preferred	1.23%	9.74%	16.40%	**72.63%**

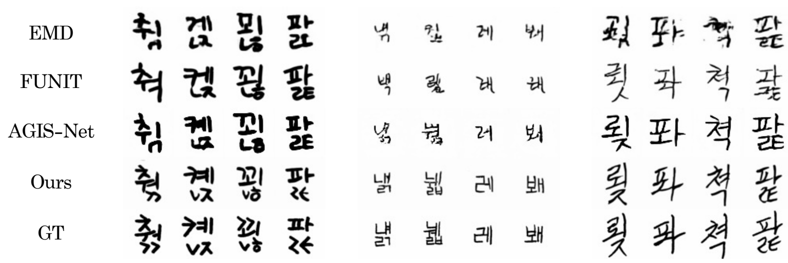


Fig. 6: Samples for the user study. The Korean-unrefined dataset is used.

Table 4: **Ablation studies on the Korean-handwriting dataset.** Each content and style score is an average of the seen and unseen accuracies. Hmean denotes the harmonic mean of content and style scores.

(a) Impact of the memory modules.

	Content	Style	Hmean
Baseline	96.6	6.5	12.2
+ Dynamic memory	**99.8**	32.0	48.5
+ Persistent memory	97.6	46.2	62.8
+ Compositional G	98.3	**63.3**	**77.0**

(b) Impact of the objective functions.

	Content	Style	Hmean
Full	**98.3**	**63.3**	**77.0**
Full $-\mathcal{L}_{l1}$	97.3	53.8	69.3
Full $-\mathcal{L}_{feat}$	97.8	51.3	67.3
Full $-\mathcal{L}_{cls}$	3.1	16.0	5.2

Style overfitting of baselines. We analyze the generated glyphs using our style classifier to investigate the style overfitting of the baseline methods. Figure 7 shows the predicted classes for each model output. We observe that the baseline methods often generate samples similar to the training samples. On the other hand, our model avoids the style overfitting by learning the compositionality of glyphs and directly reusing components of inputs. Consequently, as supported by previous quantitative and qualitative evaluations, our model is robust to the out-of-distributed font generation compared to the existing methods. We provide more analysis of the overfitting of comparison methods in the Appendix.

Component-wise style mixing. In Figure 8, we demonstrate our model can interpolate styles component-wisely. It supports that our model fully utilizes the compositionality to generate a glyph.

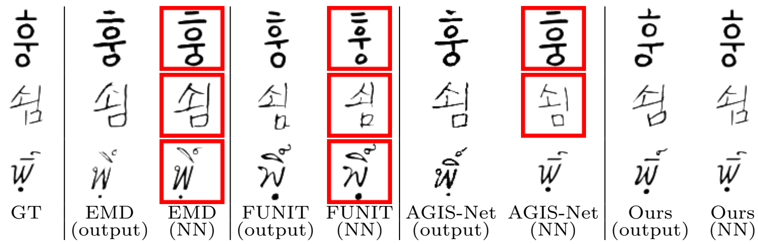


Fig. 7: **Nearest neighbor analysis.** We report the generated images by each model (output) with the given unseen reference style (GT) and the ground truth samples whose label is predicted by the style classifier (NN). Red boxed samples denote training samples. We can conclude that the baseline methods are overfitted to the training style while ours easily generalizes to unseen style.

Fig. 8: **Component-wise style mixing.** We interpolate only one component (marked by blue boxes) between two glyphs (the first column and the last column). The interpolated sub-glyphs are marked by green boxes. Our model successfully interpolates two sub-glyphs, while preserving other local styles.

6 Conclusions

Previous few-shot font generation methods often fail to generalize to unseen styles. In this paper, we propose a novel few-shot font generation framework for compositional scripts, named Dual Memory-augmented Font Generation Network (DM-Font). Our method effectively incorporates the prior knowledge of compositional script into the framework via two external memories: the dynamic memory and the persistent memory. DM-Font utilizes the compositionality supervision in the weakly-supervised manner, *i.e.*, neither component-wise bounding box nor mask used during the training. The experimental results showed that the existing methods fail in stylization on unseen fonts, while DM-Font remarkably and consistently outperforms the existing few-shot font generation methods on Korean and Thai letters. Extensive empirical evidence support that our framework lets the model fully utilize the compositionality so that the model can produce high-quality samples with only a few samples.

References

1. zi2zi: Master chinese calligraphy with conditional adversarial networks, https://github.com/kaonashi-tyc/zi2zi 1, 3
2. Azadi, S., Fisher, M., Kim, V.G., Wang, Z., Shechtman, E., Darrell, T.: Multi-content gan for few-shot font style transfer. In: IEEE Conference on Computer Vision and Pattern Recognition (2018) 2, 3
3. Cao, Y., Xu, J., Lin, S., Wei, F., Hu, H.: GCNet: Non-local networks meet squeeze-excitation networks and beyond. In: IEEE International Conference on Computer Vision Workshops (2019) 6
4. Chang, B., Zhang, Q., Pan, S., Meng, L.: Generating handwritten chinese characters using cyclegan. In: IEEE Winter Conference on Applications of Computer Vision (2018) 1, 3
5. Chang, J., Gu, Y., Zhang, Y., Wang, Y.F.: Chinese handwriting imitation with hierarchical generative adversarial network. In: British Machine Vision Conference (2018) 1, 3
6. Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In: IEEE Conference on Computer Vision and Pattern Recognition (2018) 3
7. Choi, Y., Uh, Y., Yoo, J., Ha, J.W.: StarGAN v2: Diverse image synthesis for multiple domains. In: IEEE Conference on Computer Vision and Pattern Recognition (2020) 3
8. Gao, Y., Guo, Y., Lian, Z., Tang, Y., Xiao, J.: Artistic glyph image synthesis via one-stage few-shot learning. ACM Transactions on Graphics (2019) 2, 3, 8, 9, 10, 13
9. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (2016) 3
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems (2014) 1
11. Han, J., Lee, Y., Ahn, S.: Korean font design textbook. Ahn graphics (2009) 1, 2
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (2016) 9
13. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems (2017) 8, 9
14. Huang, X., Belongie, S.J.: Arbitrary style transfer in real-time with adaptive instance normalization. In: IEEE International Conference on Computer Vision (2017) 3
15. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition (2017) 3
16. Jiang, Y., Lian, Z., Tang, Y., Xiao, J.: DCFont: An end-to-end deep chinese font generation system. In: SIGGRAPH Asia (2017) 1, 3
17. Jiang, Y., Lian, Z., Tang, Y., Xiao, J.: SCFont: Structure-guided chinese font generation via deep stacked networks. In: AAAI Conference on Artificial Intelligence (2019) 1, 2, 3
18. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision (2016) 8, 9

19. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition (2019) 3
20. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Universal style transfer via feature transforms. In: Advances in Neural Information Processing Systems (2017) 3
21. Li, Y., Liu, M.Y., Li, X., Yang, M.H., Kautz, J.: A closed-form solution to photorealistic image stylization. In: European Conference on Computer Vision (2018) 3
22. Lin, T.Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (2017) 6
23. Liu, M.Y., Huang, X., Mallya, A., Karras, T., Aila, T., Lehtinen, J., Kautz, J.: Few-shot unsupervised image-to-image translation. In: IEEE International Conference on Computer Vision (2019) 2, 3, 6, 8, 9, 10, 13
24. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (2015) 6
25. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep photo style transfer. In: IEEE Conference on Computer Vision and Pattern Recognition (2017) 3
26. Lyu, P., Bai, X., Yao, C., Zhu, Z., Huang, T., Liu, W.: Auto-encoder guided gan for chinese calligraphy synthesis. In: International Conference on Document Analysis and Recognition (2017) 1, 3
27. Mescheder, L., Geiger, A., Nowozin, S.: Which training methods for gans do actually converge? In: International Conference on Machine Learning (2018) 6
28. Naeem, M.F., Oh, S.J., Uh, Y., Choi, Y., Yoo, J.: Reliable fidelity and diversity metrics for generative models. In: International Conference on Machine Learning (2020) 8
29. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European Conference on Computer Vision (2016) 6
30. Srivatsan, N., Barron, J., Klein, D., Berg-Kirkpatrick, T.: A deep factorization of style and structure in fonts. In: Conference on Empirical Methods in Natural Language Processing (2019) 2, 3, 8
31. Sun, D., Ren, T., Li, C., Su, H., Zhu, J.: Learning to write stylized chinese characters by reading a handful of examples. In: International Joint Conference on Artificial Intelligence (2018) 2, 3, 8
32. Yoo, J., Uh, Y., Chun, S., Kang, B., Ha, J.W.: Photorealistic style transfer via wavelet transforms. In: International Conference on Computer Vision (2019) 3
33. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: International Conference on Machine Learning (2019) 6
34. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: IEEE Conference on Computer Vision and Pattern Recognition (2018) 8, 9
35. Zhang, Y., Zhang, Y., Cai, W.: Separating style and content for generalized style transfer. In: IEEE Conference on Computer Vision and Pattern Recognition (2018) 2, 3, 8, 9, 10, 13
36. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE International Conference on Computer Vision (2017) 3