Toward High-quality Few-shot Font Generation with Dual Memory

Junbum Cha Sanghyuk Chun Gayoung Lee
Bado Lee Seonghyeon Kim 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 fail to capture detailed styles with 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 Font Generation Network (DM-Font), which enables us to generate a high-quality font library with only a few samples. We employ dual memory components in the generator to take advantage of the compositionality. In the experiments on Korean-handwriting 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 in quantitatively and qualitatively.

1. Introduction

Advances of web technology lead people to consume more and more text content on the web instead of their handwriting. Meanwhile, designing a new font style, such as personalized handwriting, is getting important for a better user experience. However, because traditional methods to make a font library heavily rely on expert designers by manually design each glyph, creating a font library is extremely expensive for glyph-rich scripts such as Chinese (more than $50,000$ glyphs) and Korean ($11,172$ glyphs).

Several recent studies attempt to generate a font set using only a few samples [4, 5, 1, 3]. Despite their successful few-shot generation performances on in-distributed styles, existing few-shot font generation methods often fail to generate high-quality font library using unseen style few-shot samples. Example failure modes of existing few-shot methods are reported in Figure 1 and the experiment section. We solve this problem by introducing an inductive bias based on inherent glyph characteristics into the model architecture and optimization objective.

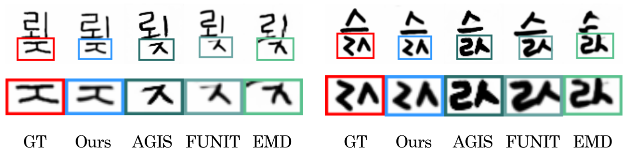


Figure 1: **Few-shot font generation results.** While previous few-shot font generation methods (AGIS [1], FUNIT [2], and EMD [5]) are failed to generate unseen font, our model successfully transfer the font style and details.

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. These compositional scripts account for 24 of the top 30 popular scripts, including Chinese, Hindi, Arabic, Korean and so on. Our framework for the few-shot font generation tasks explicitly utilizes the compositionality to more efficient and effective font generation. The proposed model, named Dual Memory Font Generation Network (DM-Font), learns the global combination recipe and the local component-wise styles from data. Unlike previous methods, we let DM-Font directly utilize local component-wise information from compositionality. In particular, we employ the dual-memory structure (*persistent memory* and *dynamic memory*) to efficiently capture the global glyph structure and the local component-wise styles. This strategy enables us to generate a new high-quality font library with only a few samples, *e.g.*, 28 samples for Korean. We validate our framework quantitatively and qualitatively using the Korean handwritten fonts. Our experiment results show both quantitatively better visual quality in various metrics and qualitatively being preferred in the user study.

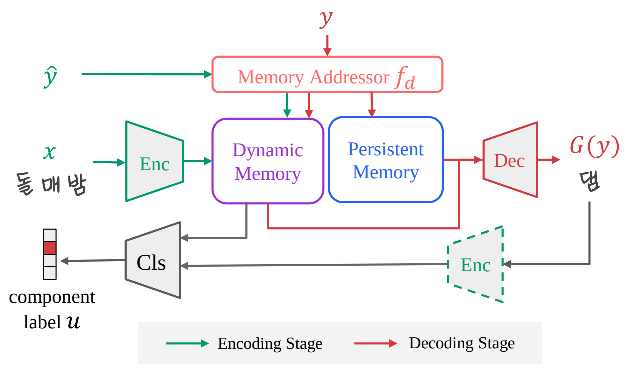


Figure 2: **Architecture overview of DM-Font**. In the encoding stage, the memory addressor places the encoded reference features into the dynamic memory using their corresponding labels. For decoding, the features saved in the dynamic and persistent memories are loaded and fed to the decoder to generate the target image.

2. Dual Memory Font Generation Network

In this section, we introduce a novel architecture, Dual Memory 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, then writes them into persistent and dynamic memory, respectively. It enables to make a high-quality full glyph library only with very few references.

2.1. Architecture Overview

We illustrate the architecture overview of DM-Font in Fig. 2. **Encoder** Enc disassembles a source glyph into the several components and encodes to the component features. The encoded component-wise features are written into *dynamic memory*.

We employ two memory modules, where **persistent memory** 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** stores encoded component features of the given reference glyphs. Hence, persistent memory captures the global information of sub-glyphs independent of each font style, while encoded features in dynamic memory learn unique local styles depending on each font.

Memory addressor provides the access address of both dynamic and persistent memory based on the given character label y_c. We use pre-defined decomposition function $f_d : y_c \mapsto \{u_i^c \mid i = 1 \ldots M_c\}$ 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.

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 dynamic memory 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 dynamic memory at address $(\hat{u}_i^c, \hat{y}_s)$, $DM(\hat{u}_i^c, \hat{y}_s)$ can be computed by $Enc_i(\hat{x})$, where i is the index of the component type of $\hat{u}_i^c$, and Enc_i is the encoder output corresponding to i.

In the decoding stage, the **decoder** Dec generates a target glyph $G(y_c, y_s)$ with the target character y_c and the reference style y_s by reading the target component-wise features from 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 the discriminator and the component classifier. The **discriminator** adopts multitask architecture [2] for the font style and character content conditioning. We further use the **component classifier** Cls to ensure the model to fully utilize the compositionality. The component classifier provides additional supervision to stabilize the training process.

2.2. 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 style, *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 dynamic memory, instead of the full component set. For example, the Korean script has the full component set with size 68, but to construct a single character, only 3 components are required.

We employ four objectives to train DM-Font: adversarial loss $\mathcal{L}_{adv}$ for the better quality of generated images, L_1 loss $\mathcal{L}_{l1}$ to add supervision from the ground truth target glyph, feature matching loss $\mathcal{L}_{feat}$ to stabilize the training better, and component-classification loss $\mathcal{L}_{cls}$ to let the model to fully utilize the compositionality:

$$\begin{aligned} \min_{G,C} \max_{D} \; & \mathcal{L}_{adv(font)} + \mathcal{L}_{adv(char)} + \\ & \lambda_{l1}\mathcal{L}_{l1} + \lambda_{feat}\mathcal{L}_{feat} + \lambda_{cls}\mathcal{L}_{cls}, \end{aligned} \tag{2}$$

where $\lambda_{l1}, \lambda_{feat}, \lambda_{cls}$ are control parameters for the importance of each objective compared to the adversarial loss. The component-classification objective aims to correctly classify the component label of encoded features from the reference and synthesized glyphs. This additional supervision leads the model to disassemble and assemble the components accurately.

3. Experiments

In this section, we empirically validate DM-Font on Korean scripts and compare it with state-of-the-art few-shot font generation methods.

3.1. Datasets

To validate the models, we build **handwriting dataset** using 86 Korean-handwriting fonts[1] which are refined by the expert designer. We train the models using 80% font sets and 90% characters, then validate the models on the remaining 20% font split. Also, to measure the generalizability to the unseen characters, we separately evaluate the models on the seen and unseen character sets for the valid font split.

We also build **unrefined dataset** from 88 non-expert Koreans, letting each applicant write 150 characters. This dataset is extremely diverse and not refined by expert designer different from the handwriting dataset. We use the unrefined dataset as the validation of the models trained on the handwriting dataset, *i.e*., the unrefined dataset is not visible during the training. In our experiments, we measure the robustness to out-of-distributed styles for each few-shot font generation methods. We generate a target font library via 30 reference samples for both datasets.

3.2. Comparison Methods and Evaluation Metrics

We compare our model with state-of-the-art few-shot font generation methods, including EMD [5], AGIS-Net [1], and FUNIT [2]. 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. We exclude the methods which are Chinese-specific [4] or not applicable to glyph-rich scripts [3]. Every competitor uses identical 30 style reference glyphs.

For the disentangled quantitative evaluation, we train a content classifier and a style classifier using character and font labels. We use top-1 accuracy (Acc) and mean FID (mFID) from the classifiers as evaluation metrics in addition to the multi-scale structural similarity (MS-SSIM) index.

3.3. Main Results

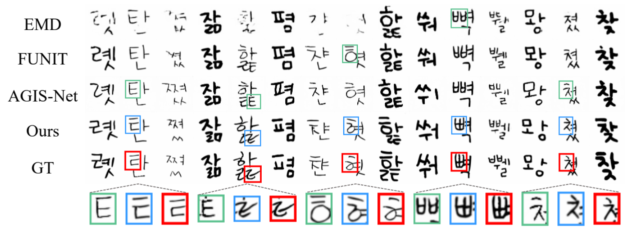
The main quantitative results on the handwriting dataset are reported in Table 1. In the experiments, DM-Font remarkably outperforms the previous methods in all evaluation metrics, especially on style-aware benchmarks. Specifically, baseline methods show slightly worse content-aware performances on unseen characters than seen characters, *e.g*., AGIS-Net shows worse content-aware accuracy (98.7 $\rightarrow$ 98.3) and mFID (23.9 $\rightarrow$ 25.9) in Table 1. In contrast, DM-Font shows better generalizability to the unobserved characters during the training. Since 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. Moreover, our method shows significant improvements in style-aware metrics. Our model achieves 62.6% accuracy while other methods show much less accuracy, *e.g*., about 5%. Likewise, DM-Font shows dramatic improvements in mFID as well as the accuracy measure.

We also provide visual comparisons in Figure 3, which contain various challenging fonts including thin, thick and

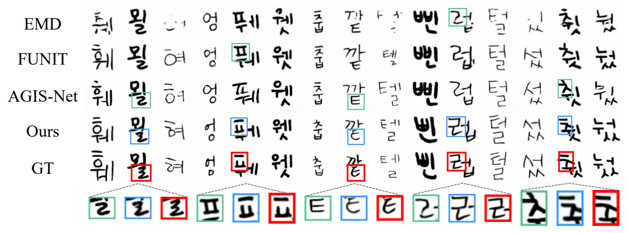
[1] We collect public fonts from `http://uhbeefont.com/`.

Table 1: **Quantitative Evaluation on the handwriting dataset.** We evaluate the methods on the seen and unseen character sets. Higher is better, except mFID.

	Pixel	Content		Style	
	MS-SSIM	Acc(%)	mFID	Acc(%)	mFID
Evaluation on the **seen** character set during training					
EMD [5]	0.361	80.4	138.2	5.1	134.4
FUNIT [2]	0.369	94.5	42.9	5.1	146.7
AGIS-Net [1]	0.399	**98.7**	23.9	8.2	141.1
DM-Font (ours)	**0.457**	98.1	**22.1**	**64.1**	**34.6**
Evaluation on the **unseen** character set during training					
EMD [5]	0.362	76.4	155.3	5.2	139.6
FUNIT [2]	0.372	93.3	48.4	5.6	149.5
AGIS-Net [1]	0.398	98.3	25.9	7.5	146.1
DM-Font (ours)	**0.455**	**98.5**	**20.8**	**62.6**	**40.5**



(a) Seen character set during training.



(b) Unseen character set during training.

Figure 3: **Qualitative comparison on the handwriting dataset.** We show insets of baseline results (green box), ours (blue box) and ground truth (red box). In particular, for the unseen character sets, ours successfully transfers the detailed reference style, while baselines fail to generate glyphs with the detailed reference style.

Table 2: **User study results on the unrefined dataset.** Each number is the preferred model output out of 3,420 responses.

	EMD	FUNIT	AGIS-Net	DM-Font
Content preserving	1.33%	9.17%	**48.67%**	40.83%
Stylization	1.71%	8.14%	17.44%	**72.71%**
Most preferred	1.23%	9.74%	16.40%	**72.63%**

curvy fonts. Our method generates glyphs with consistently better visual quality than the baseline methods. EMD [5] often erases thin fonts unintentionally, which causes low content scores compared to the other baseline methods. FUNIT [2] and AGIS-Net [1] accurately generate the content of glyphs and captures 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. Compared to the baselines, our method generates 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.

3.3.1 User Study

We conduct a user study to further evaluate the methods in terms of human preferences using the unrefined dataset. Some example generated glyphs are illustrated in Figure 4. Users are asked to choose the most preferred generated samples in terms of content preserving, faithfulness to the reference style, and personal preference. We collect a total of 3,420 responses from 38 users using 90 preference questions. The results are shown in Table 2, 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.

4. Conclusion

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, Dual Memory 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. The experimental results showed that the existing methods fail in stylization on unseen fonts, while DM-Font remarkably outperforms the existing few-shot font generation methods. 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.

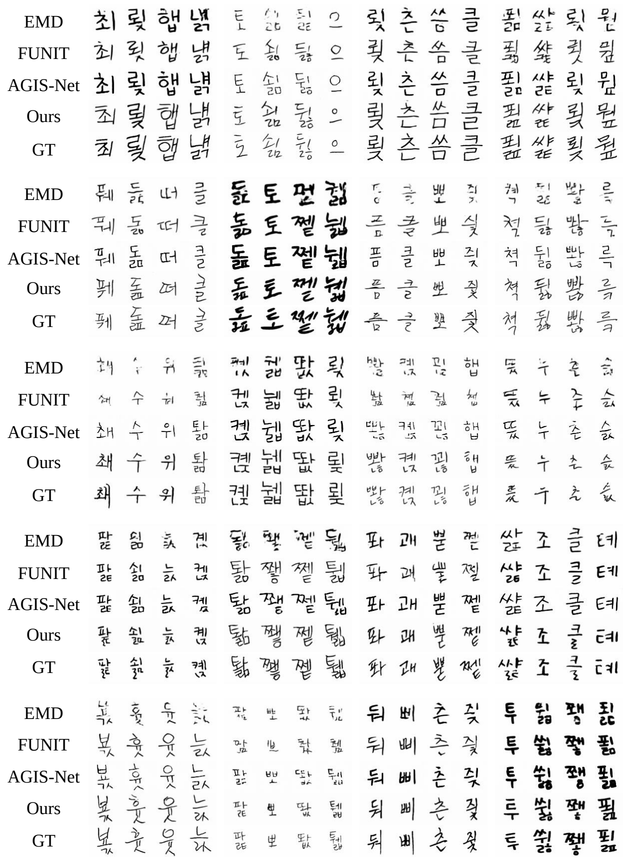


Figure 4: **Samples for the user study.** The unrefined dataset is used. Ours shows better stylization performance than baseline methods.

References

[1] Yue Gao, Yuan Guo, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. Artistic glyph image synthesis via one-stage few-shot learning. *ACM Transactions on Graphics (TOG)*, 2019. 1, 3, 4

[2] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 3, 4

[3] Akshay Srivatsan, Jonathan Barron, Dan Klein, and Taylor Berg-Kirkpatrick. A deep factorization of style and structure in fonts. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019. 1, 3

[4] Danyang Sun, Tongzheng Ren, Chongxuan Li, Hang Su, and Jun Zhu. Learning to write stylized chinese characters by reading a handful of examples. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018. 1, 3

[5] Yexun Zhang, Ya Zhang, and Wenbin Cai. Separating style and content for generalized style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 3, 4