XMP-Font: Self-Supervised Cross-Modality Pre-training for Few-Shot Font Generation

Wei Liu[*] Fangyue Liu[*] Fei Ding Qian He Zili Yi[†]
ByteDance Ltd, Beijing, China
liujikun63@gmail.com liufangyue99@hotmail.com dingfei.212@bytedance.com
heqian@bytedance.com yizili14@gmail.com

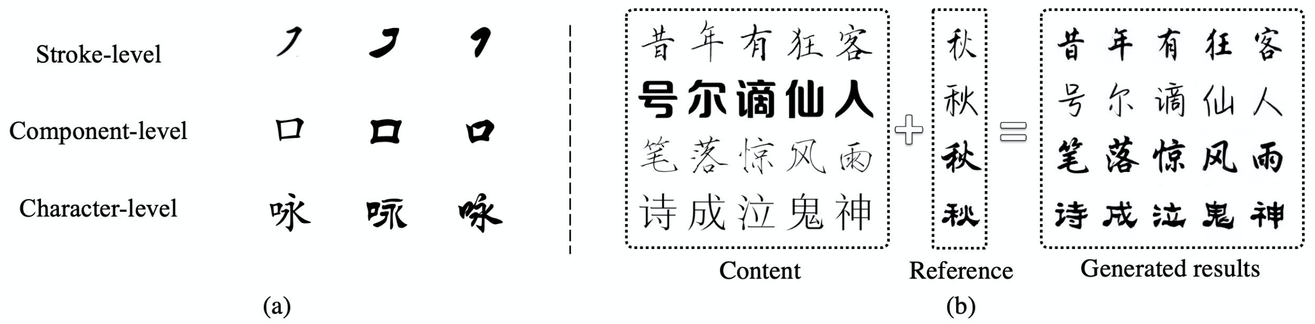

Figure 1. (a) Illustrations of different Chinese font styles. Typically, a font style involves unique morphological structures of a character at multiple scales. For example, stroke-level styles involve stroke-level features such as weight, hollowness and serif-ness. Component-level styles implies how strokes are oriented and joined to form a component. Character-level styles engage character-level features including the component layout, inter-component spacing and "connected-stroke". (b) Exemplar few-shot font generation results of our method. Please note that each row presents a disparate font style by using only one glyph as reference. In addition, our model generates faithful and consistent results regardless of the type of the source font or the reference variant.

Abstract

Generating a new font library is a very labor-intensive and time-consuming job for glyph-rich scripts. Few-shot font generation is thus required, as it requires only a few glyph references without fine-tuning during test. Existing methods follow the style-content disentanglement paradigm and expect novel fonts to be produced by combining the style codes of the reference glyphs and the content representations of the source. However, these few-shot font generation methods either fail to capture content-independent style representations, or employ localized component-wise style representations, which is insufficient to model many Chinese font styles that involve hyper-component features such as inter-component spacing and "connected-stroke". To resolve these drawbacks and make the style representations more reliable, we propose a self-supervised cross-modality pre-training strategy and a cross-modality transformer-based encoder that is conditioned jointly on the glyph image and the corresponding stroke labels. The cross-modality encoder is pre-trained in a self-supervised manner to allow effective capture of cross- and intra-modality correlations, which facilitates the content-style disentanglement and modeling style representations of all scales (stroke-level, component-level and character-level). The pre-trained encoder is then applied to the downstream font generation task without fine-tuning. Experimental comparisons of our method with state-of-the-art methods demonstrate our method successfully transfers styles of all scales. In addition, it only requires one reference glyph and achieves the lowest rate of bad cases in the few-shot font generation task (28% lower than the second best).

1. Introduction

The few-shot font generation task (FFG) aims to produce a new font library using only a few glyphs as reference, without additional fine-tuning of the model at the testing stage. FFG is especially a desirable task when designing a new font library for glyph-rich scripts such as Chinese

[*] These authors contributed equally and should be considered co-first authors.

[†] Corresponding author.

(the total number of characters exceeds 80,000), as the traditional manual font design process is very laborious. FFG is also desired when the target style glyphs are too rare to collect (e.g., historical handwriting).

Since font styles are highly complex and fine-grained, a simple analysis of the low-level textures of a few reference examples is impossible to perform successful style transfer as in [11, 16, 24, 25, 28]. A common paradigm used for FFG is to disentangle font-specific style and content information from the given glyphs, and synthesize a new glyph by combining the style embeddings extracted from the reference set and the content representations of the source glyph [2,3,9,22,29,30,33,35,41,45]. Early attempts of this stream [9,45] employ the universal style representations, using a simple convolutional encoder to extract style embeddings directly from the reference glyph images. However, the universal style representations show limited capabilities in capturing reliable and content-independent style representations due to limited awareness of the character structures and correlations between different regions of the input glyph. More advanced architectures such as DM-Font [3], LF-Font [29], MX-Font [30] propose to use structure-aware style representations and learn the localized component-wise style representations.

To make the localized style representation possible, these methods either condition the style encoders jointly upon the glyph image and the corresponding component labels or introduce component-label-guided losses to train the style encoder. The structure-aware localized style representations remarkably improve the reliability of the style representations. However, as mentioned in [29], learning component-wise styles solely is insufficient for component-rich glyphs like Chinese characters that have over 200 different types of components. It is hard to cover all component types with a few reference glyphs during test. To relieve this problem, LF-Font [29] simplifies the component-wise styles by a product of component factor and style factor, inspired by low-rank matrix factorization. MX-Font [30] extracts multiple style features not explicitly conditioned on component labels, but automatically by multiple experts to represent different local concepts, thus enabling the model to be generalized to a character with unseen components.

Such solutions relieve the “unseen components” issue to some extent. However, they are prone to generating bad cases when failing to generalize the unseen component styles from seen components. On the other hand, component-wise style representations are incapable of capturing character-level style features (e.g., inter-component spacing), which is an important perspective in many Chinese font libraries: see Figure 1 (a) for the Chinese font styles of all three scales.

To address these issues, we make two significant changes. First, we employ the stroke labels rather than the component labels as the atomic representation of character structure, as the stroke set used in Chinese is significantly fewer (about 28) than that of the component set (more than 200), which can be easier to cover with a few reference glyphs or generalized from seen strokes. On the other hand, to enhance the awareness of the stroke-level styles while not losing component-level or character-level style features, we propose to use the unified all-scale style representations instead of the localized component- or stroke-wise styles. This can be achieved by introducing a cross-modality transformer-based encoder that is conditioned jointly on the glyph image and the corresponding stroke labels. On one hand, the self-attention layers used in the encoder is good at capturing both local and global style features. On the other hand, the self-supervised pre-training of the cross-modality encoder inspires the learning the glyph-stroke alignments, which further facilitates the content-style disentanglement and modeling of style representations at multiple scales in the downstream training phase.

In addition to the cross-modality pre-training mechanism, we propose a LSTM-based stroke loss and a style-content decoupling network which considers spatial information conservation, to enhance the reliability of the model further. Comprehensive analyses of the experimental results demonstrate our method achieves significantly lower rate of bad cases than prior FFG methods and it can successfully generate novel glyphs based on only one reference example.

To sum up, the major contributions of the paper include:

- For the first time, we introduce the cross-modality transformer-based encoder and the mechanism of cross-modality pre-training to the FFG task. The self- and cross-attention layers in the transformer-based encoder pre-trained with self-supervised signals help capture local and global style features (stroke-level, component-level and character-level features) and learn the glyph-stroke alignments, thus enhancing the structure-awareness of style representations and facilitating the style-content disentanglement in the downstream FFG task.

- We elaborate a style-content decoupling network composed of Efficient Channel Attention (ECA) modules [31], and employ an 8×8 feature map instead of a simple average-pooled vector to represent styles or contents with the expect to conserve spatial information, which prove to be effective in increasing the reliability of the model.

- We also propose a novel stroke loss based on a pre-trained LSTM-based stroke order predictor, to enforce the correct stroke order of the generated glyph instead of the existence of stroke labels only, which proves to benefit the structure preservation and faithful generation of stroke-order-related style features (e.g., "connected-stroke").

- Experimental results see powerful generalizability of our model to unseen font domains. Our model can perform successful font style transfer with only one reference glyph.

2. Related work

2.1. Image-to-image translation

Image-to-image translation methods [5, 6, 18, 26, 27, 42, 42, 46] that learn the mapping between domains can be used for cross-domain font generation. For example, StarGAN-v2 [6] proposes to do image-to-image translation across multiple-domain in a unified framework. FUNIT [27] aims to translate an image to the given reference style while preserving the content without fine-tuning the model during test, which can be used for the FFG task. In this paper, we have our method compared with StarGAN-v2 as for generating glyphs of seen font domains, and also compared with FUNIT on both seen and unseen font domains.

2.2. Many-shot font generation methods

Early font generation methods [10, 17, 19, 37, 40] train the cross-domain translators between different font styles. Some font generation methods [10, 17, 19, 40] train a translation model first, and fine-tune the translation model with many reference glyphs of the target style. For example, hundreds of reference glyphs in the target domain are used in [19]. Despite their remarkable performances, their scenario is very limited because collecting hundreds of glyphs with a coherent style can be very expensive. In this paper, we aim to generate an unseen font library without any expensive fine-tuning or collecting a large number of reference glyphs for that style.

2.3. Cross-modality pre-training

Cross-modality pre-training [4, 21, 23, 32, 36, 43, 44] is widely used in visual-linguistic tasks such as image-text matching [4, 32, 36], visual question answering [4, 23, 34, 36], image captioning [44], etc. Cross-modality tasks require the understanding of both modalities, and the alignment and relationships between the two modalities. The pre-training enables the encoder to produce representations with fused cross-modality information, thus benefiting downstream tasks. Motivated by the concept, we introduce such mechanisms to the font generation tasks. The framework to learn vision-and-language connections is adapted to the glyph-stroke correlation learning, with the expect to increase the structure-awareness of style encoding. In cross-modality pre-training, we build a transformer model that consists of three encoders: a glyph processing module, a stroke processing encoder, and a cross-modality module. Next, to endow our model with the capability of connecting a glyph image and its related stroke labels, we pre-train the model with large amounts of glyph-stroke pairs, via self-supervised signals (reconstruction of the input data). This task helps in learning both intra-modality and cross-modality relationships.

3. Method

3.1. Overall pipeline

As shown in Figure 2 (top), the encoder takes two modalities as input: the glyph image of specific style and a sequence of stroke labels representing the corresponding character structure of the glyph. The encoder processes the two modalities separately with two single-modality modules before they are joined with a cross-modality module. In the pre-training stage, the encoder is followed by a convolutional decoder and stroke label predictor, and is designated for self-supervised representation learning (i.e., trained to reconstruct the inputs).

In the second stage, the cross-modality encoder is frozen and used for the downstream task: see Figure 2 (bottom). In this stage, we follow the style-content disentanglement paradigm and synthesize novel fonts by combining the style features of the reference glyphs and the content embeddings of the source glyphs. The cross-modality encoder is appended with a decoupling network that aims to decouple the style and content representations from the fused cross-modality representations, which is further followed by a convolutional decoder that is designated to generate novel fonts by taking the style representations of the reference and the content representations of the source as input.

In the following sections, more detailed descriptions of the sub-modules and training methodologies are presented.

3.2. Cross-modality encoder

As shown in Figure 2, the cross-modality encoder is made up of two input embedding modules, two single-modality modules and the cross-modality module. The input embedding module converts the input data (glyph and stroke labels) into embeddings sequences. Then, the single-modality module processes each modality separately before they are joined with the cross-modality module. Next, we describe the sub-modules of this cross-modality encoder in detail.

Input embedding module The input embedding module converts the input data (i.e., a glyph image and a stroke label sequence) into two separate embedding sequences (glyph embeddings and stroke embeddings). The stroke embedding sequence consists of 28 stroke embeddings that ar-

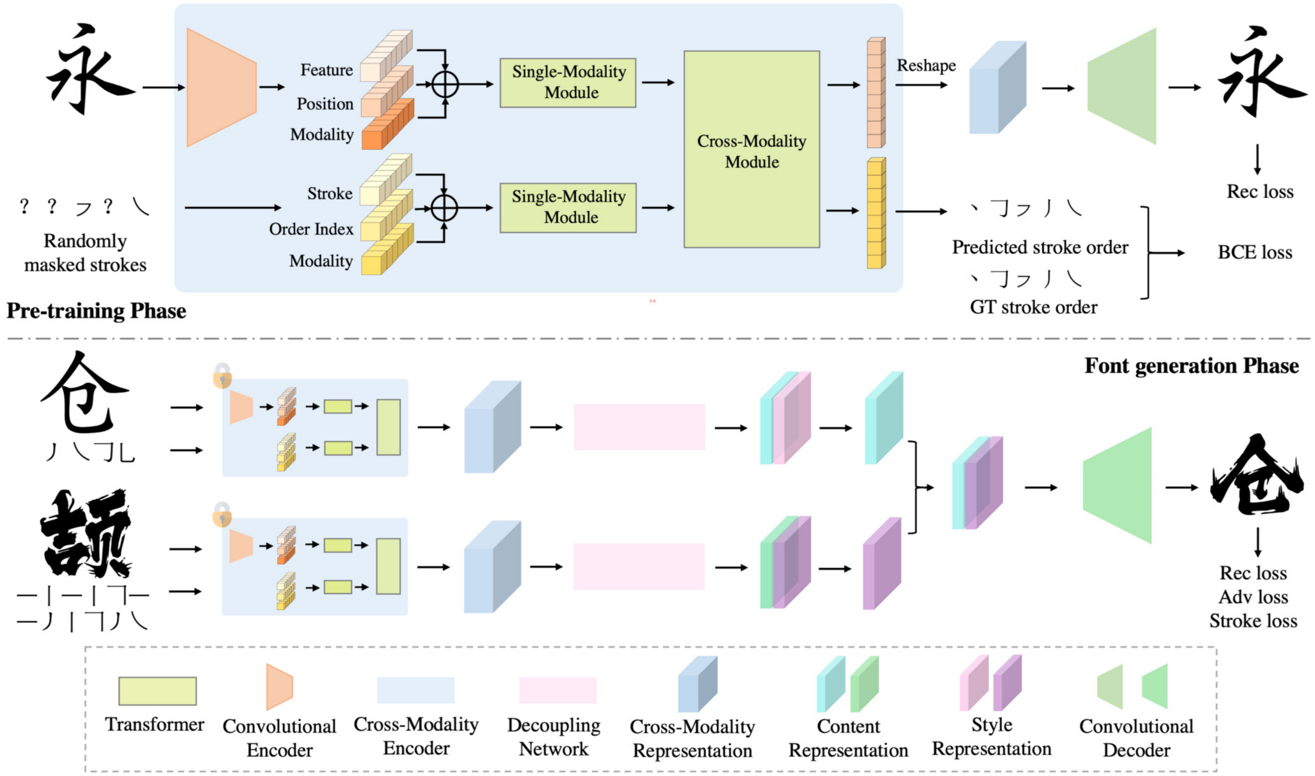

Figure 2. The framework of the proposed XMP-Font model. Our approach consists of the pre-trining phase and the downstream font generation phase. The cross-modality encoder is shared in both phases.

ranged in the stroke order. Note that 28 is the maximum number of strokes forming a commonly-used Chinese character, and the stroke order of each Chinese character is unambiguously determined as strokes are typically arranged based on their spatial coordinates, i.e., from left to right and from top to bottom. The stroke order inputs contain the spatial information, which is also beneficial to improve the final result, especially for the deformation of structure and correct generation of "connected-stroke". Each stroke embedding is a sum of three different embeddings: the stroke label embedding, position embedding and modality type embedding. In detail, the label embedding is a 512-dimensional (512-d) vector mapped from the stroke label with an embedding sub-layer. Similarly, the position index (from 0 to 29) of a stroke is projected to a 512-d position embedding with a position embedding sub-layer, and the modality label (0 for stroke and 1 for glyph image) is projected to a 512-d modality type embedding with a modality type embedding sub-layer. Thus, we obtain a 30-embedding sequence for the stroke modality.

On the other hand, the input glyph image of size $256 \times 256 \times 3$ is mapped to a feature map of size $8 \times 8 \times 512$ with a 5-layer convolutional encoder. The feature map is further flattened to a sequence of 64 512-d embeddings, in which each embedding corresponds to a specific spatial coordinate. Similarly, the position embedding and the modality type embedding of each spatial coordinate are mapped from the x-y coordinate and the modality label respectively with separate embedding sub-layers. The glyph embedding is a sum of the position embedding, modality type embedding and feature embedding. Thus, we obtain a 64-embedding sequence for the glyph modality.

Note that the inclusion of positional information is necessary for the pre-training and font generation task, because the following transformer layers are agnostic to the absolute indices of their inputs as the order of the stroke or image embeddings is not specified.

Self- and cross-attention layers We build our single- and cross-modality processing modules mostly on the basis of self-attention layers and cross-attention layers [38]. After the input embedding module, we obtain two embedding sequences each representing a specific modality. We first apply two single-modality modules, i.e., 9 BERT [8] layers to process stroke information and 5 BERT layers for glyph processing.

Each cross-modality layer in the cross-modality module consists of two self-attention sub-layers [38], one bidirectional cross-attention sub-layer [36], and two feed-forward sub-layers. We stack these cross-modality layers in our implementation. The bi-directional cross-attention sub-layer contains two unidirectional cross-attention sub-layers: one from stroke to glyph and one from glyph to stroke. Note that the query and context vectors are the outputs of the former layer (i.e., stroke features and glyph features). The cross-attention sub-layer is used to exchange the information and align the entities between the two modalities in order to learn joint cross-modality representations. For further building internal connections, the self-attention sub-layers are then applied to the output of the cross-attention sub-layer. Lastly, the final output is produced by feed-forward sub-layers. We also add a residual connection and layer normalization after each sub-layer.

3.3. Pre-training strategy

In order to learn a better initialization which understands connections between the glyph image and its related stroke labels, we pre-train the cross-modality encoder with a pre-training task on a large font library dataset. As shown in Figure 2 (top), to ensure effective interaction between the two modalities, during the training phase, there is a probability of 0.375 that all input stroke labels are masked. In the remaining cases, each stroke has a probability of 0.5 to be masked. We attach the encoder with a stroke prediction head consisting of two fully-connected layers. The embedding sequence of the stroke modality is directly mapped to the stroke labels. In addition to where masked strokes are predicted from the non-masked strokes in the stroke label modality, our model could predict masked strokes from the glyph modality as well, so as to resolve ambiguity. For example, as shown in Figure 2 (top), it is hard to determine the masked stroke from its stroke context but the stroke choice is clear if the visual information is considered. Hence, it helps building connections from the glyph modality to the stroke modality, and we refer to this task as stroke reconstruction task. We perform the task of learning the labels of masked strokes with cross-entropy loss. We also attach the encoder with a convolutional decoder with the expect to reconstruct the glyph image. The 64-embedding sequence of the glyph modality is reshaped to an 8×8 feature map, and then decoded into an image with the decoder. Further, L1 loss contrasting the input glyph and the output image is used to ensure that there is no loss of information.

We aggregate a large aligned glyph-stroke dataset from Founder font libraries [7] in the pre-training phase, which consists of 100 different fonts. We pre-train all parameters from scratch (xavier initialization [12]). Our model is pre-trained with two losses: the glyph reconstruction loss (L1 loss) and the stroke classification loss. We add these losses with equal weights as in Eq. 1. We take Adam [20] as the optimizer with a linear-decayed learning-rate schedule and a peak learning rate at $1e-4$. We train the model for 30 epochs (i.e., roughly 4,000,000 optimization steps) with a batch size of 4.

$$L^{pre} = \sum_{i=1}^{L} BCE(\hat{s}_i, s_i) + |\hat{I} - I| \tag{1}$$

where I is the predicted glyph and $\hat{I}$ is the ground-truth glyph. $s_i, \hat{s}_i$ $(i \in \{1, 2, \ldots, L\})$ are the predicted and the ground-truth stroke labels.

3.4. Downstream task of few-shot font generation

Model architecture Once the encoder is pre-trained, we freeze the parameters and use it for the font generation task. For the downstream task, the encoder is attached to a decoupling network which is made up of 4 Efficient Channel Attention (ECA) modules [31] to adaptively rescale channel-wise features and disentangle the style and content representations. The output of the decoupling network is an $8 \times 8 \times 512$ feature map, which is split into two $8 \times 8 \times 256$ feature maps. The first split feature map is designated as the style representation and the latter is treated as the content representation. Combining the content representation of the source and style representation of the reference to generate a glyph that represent the source character with the reference style. We employ 8×8 feature map instead of a latent vector to preserve richer spatial information.

From the aligned glyph-stroke dataset [7], we only use 30 font libraries and 6741 characters from each library in this phase, which are all covered in the pre-training. We train all parameters of the decoupling network and the glyph decoder while holding the encoder parameters frozen. The font generation model is trained with three losses: adversarial loss, reconstruction loss and stroke loss. The adversarial loss encourages generation of valid glyph images by using a discriminator discriminating the generated from ground-truth glyphs [13]. The reconstruction loss is the L1 difference between the generated glyph and the exact ground-truth (a glyph of target style and source character). As for the stroke loss, we pre-train an LSTM-based [15] stroke predictor, which is able to predict the stroke labels sequentially in the correct order given a glyph image as input: see Figure 3 for details. Then we use the predictor to compute the stroke loss. Instead of using the predicted labels directly, we employ the activations of the second last LSTM layer and compute the feature differences between the generated and the ground-truth glyph. We add these losses with equal weights as in Eq. 2. We take Adam [20] as the optimizer with a linear-decayed learning-rate schedule and a peak learning rate at $1e-4$. Like in the pre-training phase, we train the model for 30 epochs (i.e., roughly 5,000,000 optimization steps) with a batch size of 4.

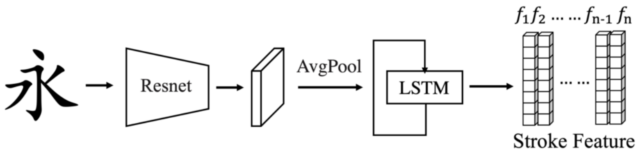

Figure 3. The LSTM-based stroke predictor, which is pre-trained for stroke order prediction before used to compute the stroke loss.

$$L^{fg} = Loss_{adv} + |\hat{I} - I| + |LSTM(\hat{I}) - LSTM(I)| \qquad (2)$$

where I, $\hat{I}$ are the predicted and the ground-truth glyphs. $LSTM(*)$ is the activations of the second last layer of LSTM-based stroke order predictor, while $Loss_{adv}$ is as the same as WGAN-GP [1].

4. Experimental Results

Our model is implemented with PyTorch 1.7 and is trained on a NVIDIA Tesla V100. The pre-training takes 2-3 days and the second-phase training costs 6 days.

We evaluate the state-of-the-art FFG methods and ours on seen and unseen font domains to measure the generalizability of the models. Our method is compared with five font generation methods on the FFG benchmark, in both the qualitative and quantitative settings. Experimental results demonstrate that XMP-Font achieves the lowest failure rate on both seen and unseen domains. The ablation and analysis study helps understand the role and effects of pre-training strategy, the use of stroke loss and other techniques.

4.1. Comparisons

We compared our XMP-Font with two image-to-image translation methods (StarGAN v2 [6] and FUNIT [27]) and three FFG methods (LF-Font [29], MX-Font [30] and DG-Font [41]). StarGAN v2 and FUNIT are not directly proposed for the font generation task, but the universal image-to-image translation paradigm can be applied to the font generation task as well. While [6] only supports translation of glyphs across seen domains, [27] can be applicable to translations between unseen domains.

To show the generalizability to the unseen style domains, we propose to do the evaluations in the following FFG scenario; training a FFG model on 100 font style domains [7], and evaluating the model on both seen and unseen style domains by using only one glyph image as reference. As the stroke labels is independent to font style, stroke labels for all 6741 characters are provided.

Model	Few Shot	FID ⇓	PSNR ⇑	SSIM ⇑	L1 ⇓	Users ⇑
			Seen Fonts			
FUNIT	Yes	147.19	8.98	0.7069	29.38	0.2667
LF-Font	Yes	58.90	9.78	0.7312	25.05	0.3725
DG-Font	Yes	73.49	9.73	0.7433	25.26	0.5759
MX-Font	Yes	66.04	9.10	0.6963	30.05	0.7220
Stargan	No	35.24	9.82	0.7336	26.64	0.7974
Ours	Yes	**31.14**	**12.94**	**0.7972**	**19.29**	**0.9249**
			Unseen Fonts			
FUNIT	Yes	173.30	8.45	0.6805	32.18	0.1060
LF-Font	Yes	86.33	9.35	0.7058	27.63	0.3185
DG-Font	Yes	53.04	9.33	0.7209	26.83	0.5220
MX-Font	Yes	135.43	8.77	0.6810	26.17	0.5906
Ours	Yes	**36.80**	**12.05**	**0.7903**	**18.78**	**0.8748**

Table 1. Quantitative evaluations of our XMP-Font and competitors. The reported values are the average of the whole datasets, where only one reference images per style is used for font generation in each experiment.

Due to the style of the font domain is defined by appearance features of multiple scales, measuring the visual quality with a unified metric is a challenging problem. As mentioned in MX-Font [30], the multiplicity of the font styles raises the issue when multiple "ground-truths" are satisfying and only one "ground-truth" glyph is present in the evaluation dataset. Thus, in addition to ground-truth-based metrics (SSIM [39], PSNR and L1), we also use evaluation metrics that does not require paired ground truths (FID [14]).

Other than the objective metrics, we conduct a user study for quantifying the subjective quality. The participants are asked to pick the acceptable cases considering the success of style transfer and correctness of the character structure. Failure of either the content or the style is considered unsuccessful. We randomly select 10 seen font styles and 10 unseen font styles, and 30 characters of each style are generated with each model. Therefore, 3300 samples are generated ($10 \times 30 \times 6 = 1800$ for seen domains and $10 \times 30 \times 5 = 1500$ samples for unseen domains as StarGAN-v2 does not work for unseen domains). The generated samples of the same style and the same character (by different models though) are put together and shown to a participant at a time. The src glyph and a few glyphs of target styles are also shown to the participant in the meantime to facilitate the rating. After all trials finish, the ratings of all participants are collected and analyzed, as presented in Table 1. We observe that XMP-Font outperforms other methods in both seen and unseen font generation scenario for most evaluation metrics. In the unseen-domain scenario, ours exceeds others in all metrics by large margin. Especially, our method achieves a remarkably 28% higher success rate on unseen font domains over the second best.

We illustrated the generated samples in Figure 4. We show the source images in the top row and the corresponding stylize transfer results in the below. In Figure 4, we

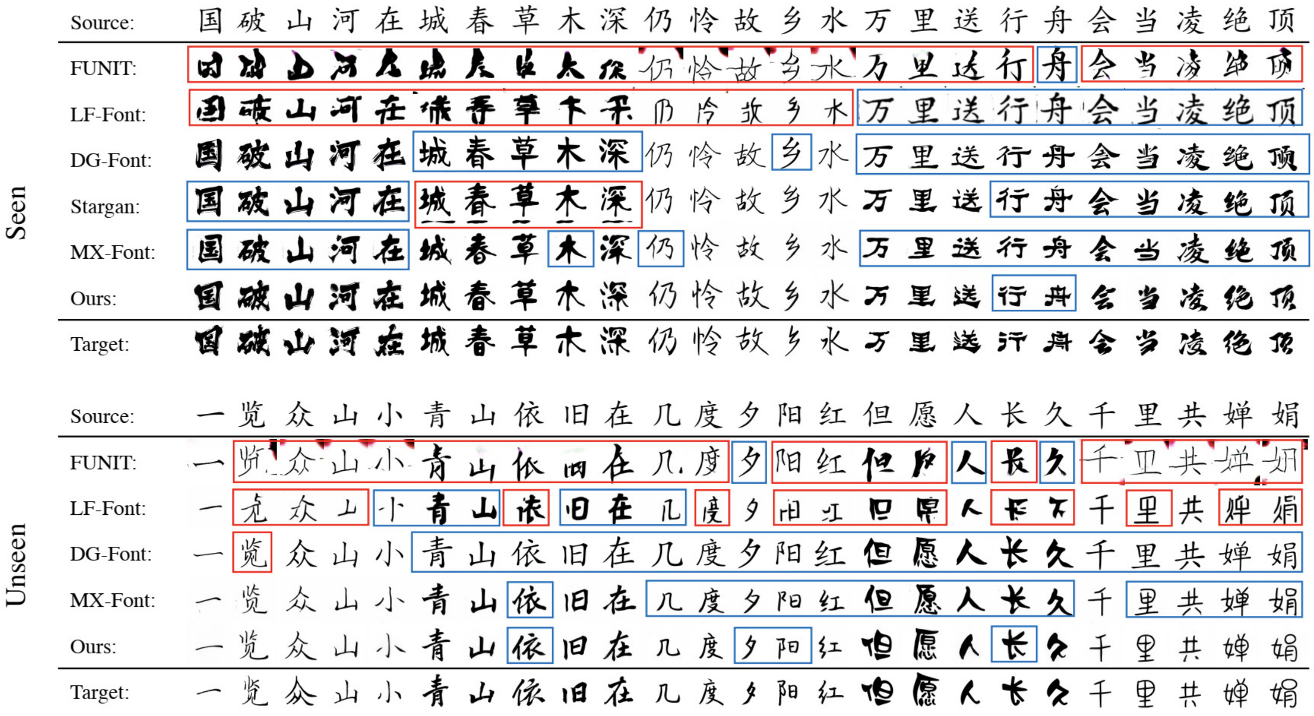
Seen
Source: 国破山河在城春草木深仍怜故乡水万里送行舟会当凌绝顶
FUNIT:
LF-Font:
DG-Font:
Stargan:
MX-Font:
Ours:
Target:
Unseen
Source: 一览众山小青山依旧在几度夕阳红但愿人长久千里共婵娟
FUNIT:
LF-Font:
DG-Font:
MX-Font:
Ours:
Target:

Figure 4. Visual comparisons of our XMP-Font with other state-of-the-art methods on famous Chinese poems. The red boxes highlight failures of structure preservation, and blue boxes highlight failures of style transfer.

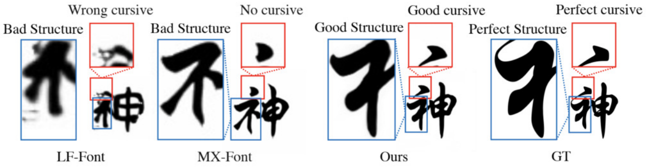

Figure 5. Comparisons of our XMP-Font with LF-Font [29] and MX-Font [30] in terms of cursive font generation. The glyph details highlighted with the blue boxes and red boxes reveal the noticeable gap between the other two models and ours.

observe that FUNIT [27] generates the worst results, as it often fails to preserve the character structure of the source (severe loss of strokes or components) and generates unrecognizable glyphs. LF-Font [29] performs well for some test styles, while its performance is unstable as they are prone to loss of strokes or distortion of components on certain style domains. At a glance, other methods including DG-Font [41], StarGAN-v2 [6], MX-Font [30] and ours seem to preserve the character structure well. However, DG-Font [41] fails to perform style transfer especially when the source style significantly differs from the target. StarGAN-v2 [6] can only do transfer of seen styles, while it occasionally generates unpleasant stroke paddings as highlighted with the red boxes in Figure 4. MX-Font and ours synthesize better detailed structures both in content and style, while there is more chance that MX-Font fails to generate fine-grained style features.

As shown in Figure 5, the blue and red boxes highlight the failure of LF-Font [29] and MX-Font [30] in terms of the generation of stroke-level styles (e.g., selfness), component- and character-level styles (e.g., connected-stroke and inter-component spacing). The advantage of our XMP-Font is highlighted with more successful transfer of styles of all scales. XMP-Font preserves both the detailed local style and fine-grained global styles and generates the plausible and recognizable images consistently. Such a noticeable gap in visual quality explains the large performance leap of XMP-Font in the user study.

4.2. Ablation studies

Pre-training strategies To show the effectiveness of the pre-training strategy used in our approach, we did three experiments. In Experiment A, the cross-modality encoder is not pre-trained and it is directly trained for the downstream task from scratch. In Experiment B, we pre-train the encoder first and in the second-stage training we only fine-tune the encoder with smaller learning rate (10^{-6}). Experiment C is what we use in our approach, namely pre-training the cross-modality encoder first and then keeping it frozen in the second-phase training. Then, we demonstrate the validation losses (stroke loss and reconstruction loss) in Figure

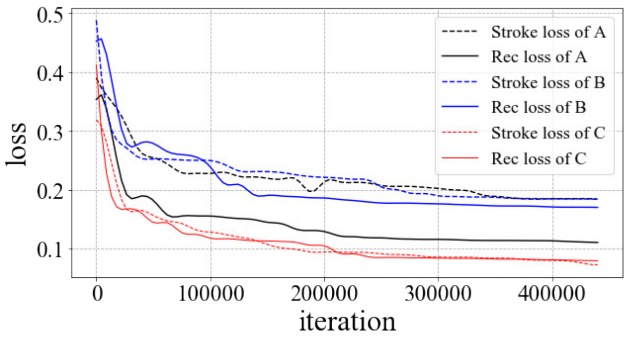

Figure 6. The validation losses over optimization steps when using different pre-training strategies. A is the experiment w/o pre-training (the encoder is trained for the downstream task from scratch). B is the experiment with pre-training first and then fine-tuning for the downstream task. C is what is used in our method, namely pre-training the cross-modality encoder first and then keeping it frozen in the second-phase training.

6. Figure 6 clearly shows that Experiment A and B fail to converge as C does, implying that the pre-training strategy is essential for our architecture.

Stroke loss and architecture design Further, to verify the effectiveness of the proposed stroke loss and the architecture of the generator, we compare the performances of different experimental settings and architecture designs on the validation benchmark under seen and unseen domain transfer scenarios. In Experiment A, we simply remove the stroke loss during the second-stage training. The results are shown in Table 2 and Figure 7. Table 2 (A) shows that the objective metrics worsen without the use of stroke loss. The visual results in Figure 7 (A) also show that the preservation of character structure becomes worse and the model is less sensitive to the stroke order.

In Experiment B, we modify the network architecture to adopt smaller glyph feature map (4 × 4 instead of 8 × 8). To make this possible, we employ a down-sampling convolutional layer upon the 8 × 8 glyph feature map before it is processed with the decoupling network. We observe from Figure 7 (B) that the smaller feature map is prone to loss of fine-grained structure information and stroke-level style features. This is also reflected by the objective metrics in Table 2 (B). However, using a 16 × 16 feature map is intractable due to limitation of GPU memory. Therefore, we choose 8 × 8 glyph feature representations in the final architecture.

In Experiment C, we ablate the decoupling network by replacing the ECA modules with a simple convolutional layer. The results in Figure 7 (C) show that content-style disentanglement worsens without the use of the decoupling network, and some results suffer loss of structure information and incorrect style features.

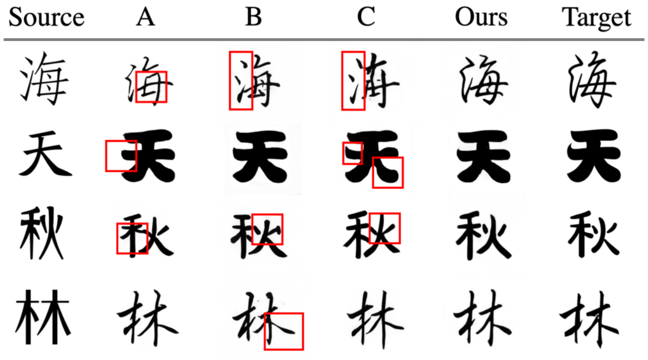

Figure 7. Qualitative analysis of the proposed techniques and architecture configurations. A is the experiment w/o stroke loss. B refers to the modified generator architecture with glyph feature size 4 × 4 instead of 8 × 8. C is the modified generator architecture w/o ECA modules [31].

	Models	A	B	C	Ours
Seen	FID	33.73	70.56	32.93	**31.14**
	PSNR	12.29	11.58	11.94	**12.95**
	SSIM	0.7933	0.7689	0.7810	**0.7972**
	L1	19.85	21.25	21.07	**19.28**
	Stroke Loss	0.3424	0.3896	0.3582	**0.2709**
Unseen	FID	41.60	56.51	45.45	**36.80**
	PSNR	11.73	10.81	11.15	**12.05**
	SSIM	0.7725	0.7653	0.7732	**0.7903**
	L1	21.70	21.59	21.63	**18.78**
	Stroke Loss	0.3692	0.4879	0.4207	**0.3272**

Table 2. Quantitative analysis of the proposed techniques. A is the experiment w/o stroke loss. B refers to the modified generator architecture with glyph feature size 4 × 4 instead of 8 × 8. C is the modified generator architecture w/o ECA modules [31].

5. Conclusion

We proposed the XMP-Font model for few-shot font generation that can generate a novel font library with high success rate by using only one reference glyph from the target domain. Both qualitative and quantitative comparisons with existing methods verify the remarkable advantages of our approach. Our approach significantly boosts the art as it achieves a record-breaking 87.5% success rate for the few-shot font generation task on unseen font domains.

Nevertheless, a limitation of our model is that it does not support unseen stroke labels, as it explicitly conditions the style and content representations upon the stroke labels. Neither can it be generalized to unseen languages whose characters are composed of a disparate set of stroke genres.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017. 6

[2] Samaneh Azadi, Matthew Fisher, Vladimir G Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. Multi-content gan for few-shot font style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7564–7573, 2018. 2

[3] Junbum Cha, Sanghyuk Chun, Gayoung Lee, Bado Lee, Seonghyeon Kim, and Hwalsuk Lee. 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*, pages 735–751. Springer, 2020. 2

[4] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation learning. In *European conference on computer vision*, pages 104–120. Springer, 2020. 3

[5] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797, 2018. 3

[6] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8188–8197, 2020. 3, 6, 7

[7] Founder Corp. Founder Type. https://www.foundertype.com/, 2021. 5, 6

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 4

[9] 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)*, 38(6):1–12, 2019. 2

[10] Yiming Gao and Jiangqin Wu. Gan-based unpaired chinese character image translation via skeleton transformation and stroke rendering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 646–653, 2020. 3

[11] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 2

[12] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 5

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 5

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 6

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 5

[16] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017. 2

[17] Yaoxiong Huang, Mengchao He, Lianwen Jin, and Yongpan Wang. Rd-gan: few/zero-shot chinese character style transfer via radical decomposition and rendering. In *European Conference on Computer Vision*, pages 156–172. Springer, 2020. 3

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 3

[19] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. Scfont: Structure-guided chinese font generation via deep stacked networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4015–4022, 2019. 3

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[21] Chenliang Li, Bin Bi, Ming Yan, Wei Wang, Songfang Huang, Fei Huang, and Luo Si. Structurallm: Structural pre-training for form understanding. *arXiv preprint arXiv:2105.11210*, 2021. 3

[22] Chenhao Li, Yuta Taniguchi, Min Lu, and Shin'ichi Konomi. Few-shot font style transfer between different languages. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 433–442, 2021. 2

[23] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer, 2020. 3

[24] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. *arXiv preprint arXiv:1705.08086*, 2017. 2

[25] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 453–468, 2018. 2

[26] Alexander H Liu, Yen-Cheng Liu, Yu-Ying Yeh, and Yu-Chiang Frank Wang. A unified feature disentangler for multi-domain image translation and manipulation. *arXiv preprint arXiv:1809.01361*, 2018. 3

[27] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *Proceedings of the*

IEEE/CVF International Conference on Computer Vision, pages 10551–10560, 2019. 3, 6, 7

[28] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4990–4998, 2017. 2

[29] Song Park, Sanghyuk Chun, Junbum Cha, Bado Lee, and Hyunjung Shim. Few-shot font generation with localized style representations and factorization. *arXiv preprint arxiv:2009.11042*, 2020. 2, 6, 7

[30] Song Park, Sanghyuk Chun, Junbum Cha, Bado Lee, and Hyunjung Shim. Multiple heads are better than one: Few-shot font generation with multiple localized experts. *arXiv preprint arXiv:2104.00887*, 2021. 2, 6, 7

[31] Pengfei Zhu Peihua Li Wangmeng Zuo Qilong Wang, Banggu Wu and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 5, 8

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. 3

[33] Nikita Srivatsan, Jonathan T Barron, Dan Klein, and Taylor Berg-Kirkpatrick. A deep factorization of style and structure in fonts. *arXiv preprint arXiv:1910.00748*, 2019. 2

[34] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019. 3

[35] Danyang Sun, Tongzheng Ren, Chongxun Li, Hang Su, and Jun Zhu. Learning to write stylized chinese characters by reading a handful of examples. *arXiv preprint arXiv:1712.06424*, 2017. 2

[36] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019. 3, 5

[37] Yuchen Tian. zi2zi: Master chinese calligraphy with conditional adversarial networks. *Internet] https://github.com/kaonashi-tyc/zi2zi*, 2017. 3

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 4

[39] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6

[40] Shan-Jean Wu, Chih-Yuan Yang, and Jane Yung-jen Hsu. Calligan: Style and structure-aware chinese calligraphy character generator. *arXiv preprint arXiv:2005.12500*, 2020. 3

[41] Yangchen Xie, Xinyuan Chen, Li Sun, and Yue Lu. Dgfont: Deformable generative networks for unsupervised font generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5130–5140, 2021. 2, 6, 7

[42] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pages 2849–2857, 2017. 3

[43] Fei Yu, Jiji Tang, Weichong Yin, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. Ernie-vil: Knowledge enhanced vision-language representations through scene graph. *arXiv preprint arXiv:2006.16934*, 1:12, 2020. 3

[44] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5579–5588, 2021. 3

[45] Yexun Zhang, Ya Zhang, and Wenbin Cai. Separating style and content for generalized style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8447–8455, 2018. 2

[46] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 3