Masking Cascaded Self-Attentions for Few-Shot Font-Generation Transformer

Jing Ma, Xiang Xiang*(✉), and Yan He

National Key Lab of Multi-Spectral Information Intelligent Processing Technology,
School of Artificial Intelligence and Automation,
Huazhong University of Science and Technology, Wuhan, China
xex@hust.edu.cn

Abstract. Few-shot Font Generation (FFG) is a practical technology widely used in designing artistic characters, handwriting imitation, and identification, etc., which aims to generate realistic font images with a few reference samples. Typically, convolutional neural networks (CNNs) are employed to learn the representations of style and content from the font images. However, owing to the locality of convolutional operations, CNNs are not good at capturing the global structure of fonts, which leads to the generated images with blurry components and distorted space layouts. To address this problem, we consider cascading self-attention modules to exploit long-range dependencies for font generation and propose a transformer-based approach called FGTr. Following the style-content disentanglement paradigm, FGTr contains two different transformer encoders to extract the style and content sequences. A multi-layer transformer decoder is adopted to merge the two sequences and generate target images. In order to smoothen the transition of patch edges, we utilize a Local Self-Attention Mask (LSAM) to restrict the attention scope of each patch to a fixed-size sliding window, which plugs into the Transformer with no extra parameters. We also propose an Auxiliary Generation Module (AGM) in favor of generating glyphs closer to the real. Extensive experiments demonstrate the effectiveness and superiority of our method compared with state-of-the-art CNN-based models.

1 Introduction

Few-shot Font Generation (FFG) have received increasing attention in recent years due to the commercial and artistic value of automated font design and completion, especially for some glyph-rich scripts such as Chinese and Japanese. FFG methods are typically trained offline on a large number of fonts and characters, and then applied to generate new images for unseen styles or contents in the few-shot setting. With the development of deep learning, FFG methods aid expert designers in manually rendering the glyphs for each character, making the process of font design more efficient and convenient.

* Also with Peng Cheng Laboratory, Shenzhen, China.

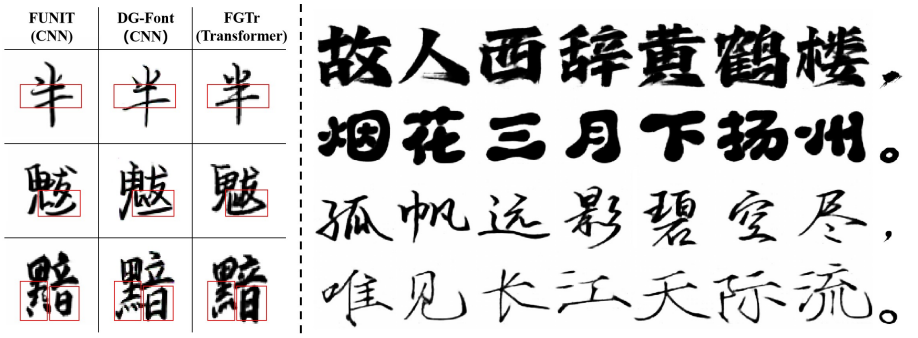

Fig. 1. (Left) Generation from CNN-based and Transformer-based FFG models. A glyph is defined by the spatial relationships of its radicals, components, and strokes, where any relative positional change alters the character's meaning and font style. The translational invariance of convolution makes it challenging to distinguish different characters with the same components. The widely used pooling operation makes it difficult for CNN's low-level local features to be preserved in high-level features. FGTr divides an image into 8×8 small patches, performs local encoding separately, and adds position embedding to endow spatial information. Cascaded attention modules allow local features in lower layers to be passed to higher layers. (Right) An ancient Chinese poem generated by our Font Generation Transformer (FGTr).

Early approaches [39,4] model FFG as an image-to-image task [16] and map font images from source to target domains, but it cannot handle unseen fonts that are not included in the training set. Afterward, SA-VAE [36] and EMD [45] disentangle the representations of content and style, and mix the content code of basis fonts with the style features of reference fonts to generate new glyphs. However, decomposing and reconstructing content from various styles is non-trivial. Following this way, component/stroke-supervised methods [32,42,26,22] are proposed to boost the characterization of local regions. Unfortunately, explicit component/stroke annotations limit the scalability of style transfer across language systems. To overcome this issue, recent methods [33,38,30] apply attention mechanisms to improve adaptability and generate realistic stylized glyphs.

Heretofore, CNNs have been used in the vast majority of FFG methods. Fundamentally, due to the local receptive field of the convolution, CNN-based FFG models are inherently unsuited for capturing component and character-level statistics unless a sufficient number of layers are passed through. However, it could cause the loss of feature resolution and fine-grained details, in addition to the optimization difficulty [22]. Moreover, the spatial invariance of the convolution limits CNNs to encode spatially heterogeneous visual patterns. That leads to unstable representations, especially for content, causing generated glyphs to keep blurry components and distorted space layouts, as shown in Fig. 1. An *et al.* [1] recently discovered the content leak phenomenon in CNN-based style transfer methods. The image content is not well preserved in the stylized images after several rounds of stylization process. But this biased representation issue

could be alleviated by adopting the attention mechanism, as demonstrated by the benefits of recently proposed attention-based FFG approaches [33,38,30].

Transformer-based architectures have so far been widely applied in vision tasks [8,12]. Self-attention mechanism supersedes the convolutional operation and manifests superiority in building rich relationships between different parts of images [34]. One crucial role of learning better contextual dependencies is that Transformer incorporates more global information than CNNs at lower layers and maintains more uniform representations [35]. So that Transformer captures and propagates quantitative features in various scales from lower to higher layers, which is beneficial for the exactitude of style and content representations, as shown in Fig. 1. Furthermore, Transformer is capable of preserving input spatial information and is efficient in generating glyphs with graceful spatial layouts.

In this work, we aim to challenge the FFG task and propose a framework called **F**ont **G**eneration **Tr**ansformer (FGTr). In contrast to generic architectures for computer vision (*e.g.*, Vision Transformer [8] and MAE [12]), we design two transformer encoders to extract style and content sequences and a transformer decoder to generate target font images. In the head of the decoder, we introduce a merging block to fuse the concatenated sequences from two encoders. We skip-connect the outputs from layers in style encoder to the layers in decoder, these intermediate features are used to guide the stylization process. Since font generation is an intensive predictive task, we adopt a hierarchical architecture by merging image patches in the encoder and reconstructing patches in the decoder. In addition, the boundary of patches is sensitive to visual perception, especially for font images composed of only black and white pixels. So we utilize Local Self-Attention Mask (LSAM) to enhance the attention of nearby patches. To alleviate the impact of random sampling in the online generation, we propose an Auxiliary Generation Module (AGM) to select generated glyphs with the highest confidence and similarity. In summary, our main contributions are as follows:

- We propose a Transformer-based framework for FFG, named FGTr, which provides richer representations than CNNs and is capable of generating high-quality glyphs. FGTr is the first one-stage font generation Transformer that follows an explicit style-content disentanglement paradigm.
- We propose LSAM to improve the attention to nearby patches and introduce AGM to enhance the online generation, which measures the confidence and similarity in the outcome for generated image selection.
- Experimental results verify that our method outperforms existing state-of-the-art FFG methods by a large margin.

2 Related Work

2.1 Image Style Transfer

Image style transfer is the task of converting the reference style to the source images. Gatys, *et al.* [11] use explicit image representations of content structures and style textures derived from CNNs and produce new images by combining the

content and style. A pile of works [19,23,14,5,9,15,20] aims to accelerate the generation speed and solve the problem of arbitrary style transfer. Recently, some few-shot methods [2,21,24,14] focus on disentangling the style and content with an encoder-transfer-decoder framework and translating the source image to arbitrary styles with only a few reference samples provided. Based on CNNs, [7,25,31] introduce self-attention mechanism to extract detailed texture of style images, alleviate the distortion of content structures, and balance local and global patterns. To eliminate the content leak phenomenon [1], Deng, *et al.* [6] propose a Transformer-based approach to capture long-range dependencies of patches. Although these methods help to generate high-quality images, they can not be directly applied to font generation. Different from ordinary artistic pictures, font images are susceptive to subtle perturbations. Thus, we design a training and generation strategy to synthesize unseen glyphs in the few-shot setting.

2.2 Few-Shot Font Generation

FFG aims to generate full-character sets with only a few reference samples. zi2zi [39] builds on pix2pix [16] to model FFG as a conditional image-to-image translation problem. Following this way, early methods [39,4,17,29,37] learn a mapping from source fonts to target with variational autoencoders or adversarial networks, but they are not able to generate unseen styles outside the training set. To address this issue, SA-VAE [36] and EMD [45] introduce two encoders to learn style and content representations, respectively. Afterward, large amounts of approaches [10,18,42,32,33] try to increase the generation quality and improve the perception. To some extent, unsupervised or character-level supervised methods generate font images with stuck strokes, blurred components, and distorted structures. ChiroGAN [10] and SCFont [18] decompose the stylization process into skeleton extraction, skeleton transformation, and stroke rendering. Calligan [42], LF-Font [32] and MX-Font [33] employ component-wise representations to synthesize complex local details. To extract robust content-independent style features, FSFont [38] adopts cross-attention to learn fine-grained styles and spatial correspondence. CG-GAN [22] proposes an attention decoder to supervise a generator at the component level. XMP-Font [26] utilizes stroke information for self-supervised cross-modality pre-training with a Transformer encoder. Pan, *et al.* [30] apply a cross-attention module to transfer the style of reference glyphs to components. Attention mechanism is more suitable than CNNs for representing local details of style and content. From this perspective, we are motivated to propose a Transformer-based framework to replace traditional CNNs for FFG.

2.3 Transformer for Vision Tasks

Transformer is a deep neural network composed of cascaded attention modules [40] and has been widely used. So far, Transformer variants designed for computer vision have been used as common models for images and videos, including object detection, semantic segmentation, and image/video generation [8,28,12]. FGTr

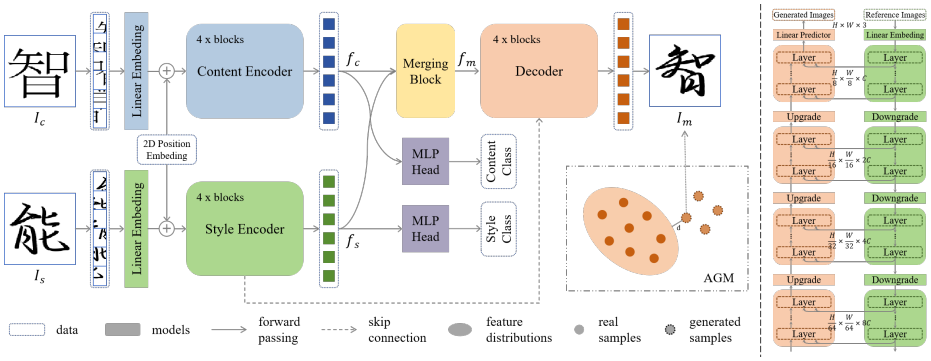

Fig. 2. The pipeline of Font-Generation Transformer (FGTr). The model utilizes two Transformer encoders to extract style and content sequences, fused by a merging block. The decoder reconstructs images under the guidance of the style encoder. The encoder and decoder have a mirror-symmetric hierarchical structure to capture different scales of representations for font images. (right) The output of each layer in style encoder serves as the key and value for the mirror layer in decoder. Hierarchical Transformer downsamples the image resolution by $1/2 \times 1/2$ and increases the feature dimension by 2 times in each block. Two MLPs are employed for classification. We introduce an Auxiliary Generation Module (AGM) for online generation to enhance the quality of generated images, which leverages the model's confidence in the generation results and the similarity with real samples in the feature space.

leverages a self-attention mechanism to boost the long-term dependencies of image representation for high-quality font generation. We retain the advantage of local focus in convolution, and propose LSAM to enhance attention for adjacent patches and edges, providing stronger local attention and spatially consistent patch processing compared to Swin [28] and ViT [8]. Similarly, FontTransformer [27] is also based on Transformer. Differently, FGTr applies a style-content disentanglement paradigm to achieve stronger generalization to unseen styles, which is widely acknowledged in FFG.

3 Methodology

In this section, we will introduce the overall framework of FGTr, and the training and online generation strategies in detail. The pipeline is illustrated in Fig. 2.

3.1 Font-Generation Transformer

Given an image $I_c \in \mathbb{R}^{H \times W \times 3}$ as the content representation of a character and an image $I_s \in \mathbb{R}^{H \times W \times 3}$ as the style reference, font generation aims to synthesize a glyph with the style of I_s and the semantic content of I_c. Similar to Vision Transformer [8], we split both images into patches and use a linear embedding layer to project the input patches into sequences in the shape of $L \times C$, where

C denotes the dimension of input space. We adopt $m = 8$ as the patch size, thus the length of the input sequence is $L = (H \times W)/(m \times m)$. When using Transformer-based models, positional embedding is added to the input sequence to obtain structural information, and we set 2-dimensional sinusoidal positional embeddings [40] with size $C/2$ for the X and Y axes, respectively:

$$z_0 = [x_p^1 E; x_p^2 E; \cdots; x_p^N E] + \text{Concat}(E_{pos-x}, E_{pos-y}), \tag{1}$$

where $E \in \mathbb{R}^{(3m^2)\times C}$ are learnable and $E_{pos-x}, E_{pos-y} \in \mathbb{R}^{N\times C/2}$ are fixed.

Different from MAE [12] with one encoder, we follow the common paradigm of style-content disentanglement, and use two Transformer encoders to extract the style-specific and content-specific representations respectively, which are merged for image reconstruction in the next stage. Since characters are constructed of strokes, components, and radicals, the attributes of a font image are contained in different scales. To extract features with different receptive fields, we hierarchically construct the encoder and decoder. Refer to Fig. 3 for detailed structures.

Style and Content Encoder. An encoder consists of 4 blocks, each block is composed of a downgrade layer (replaced by linear embedding at the first block) and multiple attention layers. Given an input sequence $z \in \mathbb{R}^{L\times C}$, the downgrade layer concatenates the features of 2×2 neighboring patches, and applies a linear layer on the $4C$-dimensional concatenated features, outputting a sequence with $2C$ dimensions. So it reduces the shape of the input sequence to $z_d^0 \in \mathbb{R}^{L/4\times 2C}$, while the image resolution is $(H\times W)/(2m\times 2m)$. Each attention layer contains a Multi-head Self-Attention module (MSA) and a Multi-Layer Perceptron (MLP). The reduced sequence z_d is transformed into query (Q), key (K), and value (V), and the self-attention is computed by

$$\begin{aligned} &Q = z_d W_q, \quad K = z_d W_k, \quad V = z_d W_v, \\ &\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T \odot M}{\sqrt{d_k}})V, \end{aligned} \tag{2}$$

where $W_q, W_k, W_v \in \mathbb{R}^{2C\times d_k}$ are transformation matrices, and M is mask to calculate Hadamard product with QK^T. Multi-head self-attention is denoted by

$$\text{MSA}(z_d) = \text{Concat}(\text{Attention}_1(Q, K, V), \ldots, \text{Attention}_N(Q, K, V))W_o, \tag{3}$$

where $W_o \in \mathbb{R}^{2C\times 2C}$ is a learnable matrix, N is the number of attention head and $d_k = 2C/N$. The computation process of the attention layer in encoders is

$$\dot{z}_d^l = \text{MSA}(\text{LN}(z_d^{l-1})) + z_d^{l-1}, \quad z_d^l = \text{MLP}(\text{LN}(\dot{z}_d^l)) + \dot{z}_d^l, \quad l = 1, \ldots, L, \tag{4}$$

where layer normalization (LN) and residual connection are applied to benefit cascading layers and faster optimization.

The style and content encoders employ the same architecture and output style sequences $f_s \in \mathbb{R}^{L/64\times 8C}$ and content sequences $f_c \in \mathbb{R}^{L/64\times 8C}$. Since Transformer-based networks require sufficient data for training, instead of separating style fonts and basis fonts [41,22], we train style and content encoders

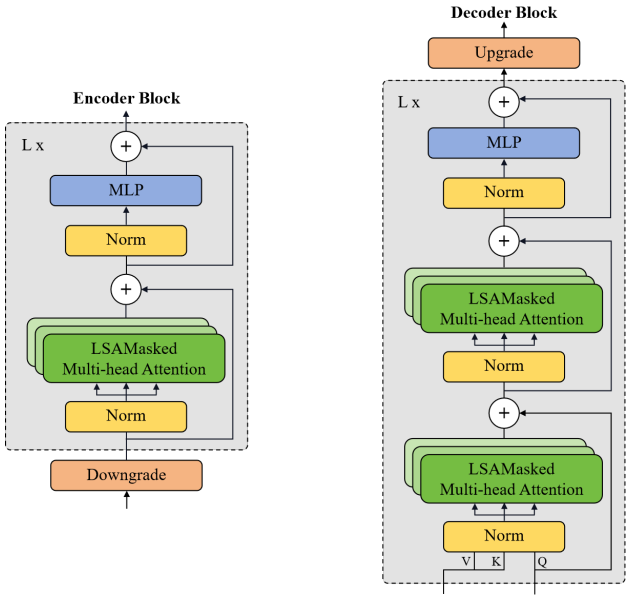

Fig. 3. Detailed structures of the encoder block and the decoder block. We apply a Local Self-Attention Mask (LSAM) in the multi-head attention layer to enhance the attention of neighboring patches, thereby smoothing the edge transition of patches.

on the whole dataset. To make the two encoders focus on different attributes of font images, we add two MLPs for style and content classification.

Merging Block and Decoder. The Merging Block (MGB) is designed to fuse the style and content sequences, and its output is used to represent the character of I_c with the I_s style. MGB consists of several self-attention layers, each having the same attention structure as in the encoder. The difference is that the style and content sequences are concatenated and fed into MGB, whose output maintains the same shape as f_c, which is denoted as

$$f_m = \text{MGB}(\text{Concat}(f_s, f_c)). \tag{5}$$

The decoder utilizes f_m and the sequences of each layer output from the style encoder as conditions to reconstruct an image with the same size as I_s. The decoder has a hierarchical structure that is symmetrical to the encoder, as shown in Fig. 2. Each block contains an upgrade layer and multiple attention layers. Given an input sequence $z \in \mathbb{R}^{L\times C}$, the upgrade layer reconstructs each patch into a group of 2×2 patches, while reducing the feature dimension to $C/2$ with a linear layer $z_u^0 \in \mathbb{R}^{4L\times C/2}$. Each attention layer contains two MSAs and one MLP. We use the style sequence z_d at the mirrored position in the encoder to generate the key (K) and value (V) for the first MSA, and the output of the previous layer as the query (Q):

$$Q = z_u W_q, \quad K = z_d W_k, \quad V = z_d W_v. \tag{6}$$

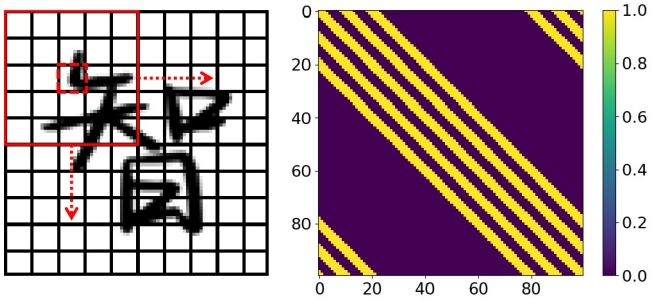

Fig. 4. (left) A font image is divided into patches with $m = 8$. A patch and its surrounding patches that compute attention with it are highlighted in red. (right) Visualization of Local Self-Attention Mask, which operates on a sequence of length $L = 100$, and marked the attention scope with 1.0.

The computation process of each attention layer in the decoder is as follows:

$$\begin{aligned} \ddot{z}_u^l &= \mathrm{MSA}(\mathrm{LN}(z_u^{l-1}),\ \mathrm{LN}(z_d^{L-l+1})) + z_u^{l-1}, \\ \dot{z}_u^l &= \mathrm{MSA}(\mathrm{LN}(\ddot{z}_u^l) + \ddot{z}_u^l, \\ z_u^l &= \mathrm{MLP}(\mathrm{LN}(\dot{z}_u^l)) + \dot{z}_u^l, \qquad l = 1, \ldots, L, \end{aligned} \tag{7}$$

Output images are mapped to $[-1, 1]$ by a Tanh function.

Local Self-Attention Mask. In font images, the smooth transition of patch edges in the neighborhood has an impact on human visual perception. The discontinuity between adjacent pixels at the patch edge will cause the truncation of strokes and distortion of glyphs, as shown in Fig. 1. The original full attention computes the similarity between a patch and all patches to obtain the weighted sum of global patches. When each patch pays attention to the global information, it tends to reduce the importance of the surrounding patches, resulting in unnatural transitions at the patch edges. To address this issue, we adopt the Local Self-Attention Mask (LSAM) to force each patch to only pay attention to patches within a certain range around it. Specifically, LSAM restricts the attention scope of each patch to a fixed-size sliding window.

Given a sequence of length L, the matrix QK^T in Eqn. 2 has a shape of $L \times L$. We set the attention scope of LSAM to 5×5, and the mask $M = (m_{ij})_{L \times L}$ is shown in Fig. 4. LSAM is generated offline solely based on the length of input sequence, and as part of the program initialization, it remains unchanged during the model training, while not being stored on the disk as network parameters.

3.2 Network Optimization

In the training stage, we construct image pairs (I_t, I_c, I_s), where I_t is the target image. Based on the attributes of I_t, we randomly select a style-consistent image from the dataset as I_s, and another character-consistent image as I_c. The

generated font image I_m should maintain the same character as I_c and the same style as I_s. Therefore, we compute the content loss $\mathcal{L}_c$ and style loss $\mathcal{L}_s$ between synthetic image I_m and target image I_t.

We normalize pixels to the range of $[-1, 1]$, and content loss is defined as

$$\mathcal{L}_c = \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{1}{N(I_t^{i,j}|I_t)} \|I_m^{i,j} - I_t^{i,j}\|_2^2, \tag{8}$$

where $N(\cdot)$ is the statistical number of black and white pixels in target image:

$$N(x|I_t) = \begin{cases} \sum_{i=1}^{H} \sum_{j=1}^{W} \mathbb{1}(I_t^{i,j} \geq 0), \text{ if } x \geq 0, \\ \sum_{i=1}^{H} \sum_{j=1}^{W} \mathbb{1}(I_t^{i,j} < 0), \text{ if } x < 0. \end{cases} \tag{9}$$

We measure the style of patches and global images utilizing the mean $\mu(\cdot)$ and variance $\sigma(\cdot)$. The style loss is

$$\begin{aligned} \mathcal{L}_s = \frac{1}{N_p} \sum_{i=1}^{N_p} & \|\mu(P_m^i) - \mu(P_t^i)\|_2^2 + \|\sigma(P_m^i) - \sigma(P_t^i)\|_2^2 \\ & + \|\mu(I_m) - \mu(I_t)\|_2^2 + \|\sigma(I_m) - \sigma(I_t)\|_2^2, \end{aligned} \tag{10}$$

where P denotes the patches of image I. We also employ identity loss [31] for richer and more robust representations. We input the same image I_i to style and content encoder respectively, and the generated output I_o should be consistent with the input I_i. Therefore, we compute identity loss between I_o and I_i as

$$\mathcal{L}_{iden} = \mathcal{L}_c(I_o, I_i) + \mathcal{L}_s(I_o, I_i). \tag{11}$$

As mentioned in Sec. 3.1, we classify the style and content sequences f_s, f_c with two MLPs. The output logits of them are used to compute Cross-Entropy loss with the style and content label l_s, l_c of target image I_t, which is defined as

$$\begin{aligned} \mathcal{L}_{ce} = & \text{CrossEntropy}(\text{softmax}(\text{MLP}(f_s)), l_s) \\ & + \text{CrossEntropy}(\text{softmax}(\text{MLP}(f_c)), l_c). \end{aligned} \tag{12}$$

Overall, the entire network is optimized by minimizing the following function:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_s + \mathcal{L}_{iden} + \lambda \mathcal{L}_{ce}, \tag{13}$$

where λ is a hyper-parameter to balance the generation and classification tasks, which is set to 1.0 in our experiments.

3.3 Auxiliary Generation Module

Existing methods for FFG suffer from the limitation of selecting appropriate style references for generating individual target glyphs. Since deep neural networks cannot fully disentangle the style and content representations, the style sequence

of an image is not completely independent of the content. Therefore, it becomes imperative to explore methods to enhance the quality of the generated images in the online generation stage.

To reduce the randomness of the generation with different inputs, we measure the quality of generated images from two aspects: 1) We treat the font images as 2D masks composed of two categories of black and white pixels, and the output of the decoder as the probabilities of classification. A higher average probability indicates the model is more confident in the outcome. 2) We count the similarity between the features extracted by the style and content encoder from the generated and real images. Generated samples with higher overall similarity are more likely to be consistent in style and content with the target images.

Specifically, given reference samples of the target style $\{I_s\}_n$, we store the sequences extracted by the style encoder $\{f_s\}_n$. Meanwhile, we randomly select samples from the training set with the same characters as content references $\{I_c\}_n$. The model takes (I_s, I_c) as an input and generates n font images $\{I_m\}_n$. We compute the confidence score by

$$S_{conf} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} |I_m^{i,j}|, \tag{14}$$

where $I_m^{i,j} \in [-1, 1]$. We encode the features of a generated image f_{ms}, f_{mc} and measure their consistency with real samples using cosine similarity:

$$S_{sim}^s = \frac{1}{n} \sum_{i=1}^{n} \frac{f_{ms}\ f_s^i}{|f_{ms}|\ |f_s^i|}, \quad S_{sim}^c = \frac{1}{n} \sum_{i=1}^{n} \frac{f_{mc}\ f_c^i}{|f_{mc}|\ |f_c^i|}. \tag{15}$$

The similarity score is normalized by $(S_{sim}+1)/2$. We combine the above scores:

$$S_{tol} = S_{conf} \times S_{sim}^s \times S_{sim}^c. \tag{16}$$

Afterward, we select the sample with the highest score from generated images $\{I_m\}_n$ as the final output. The n is the number of reference images for a target style during online generation, i.e., the n-shot setting for FFG. It is set to $n = 10$ in our experiments, refer to Sec. 4.2.

4 Experiments

4.1 Experimental Setup

Datasets. We collect 450 Chinese fonts and built a dataset to evaluate our method in the FFG task. The dataset covers all 6,763 Chinese characters of GB/T-2312 [3]. We split the dataset into 390 fonts for training and 60 fonts for testing. Unlike previous methods [41,22] that use basis fonts (*Song*, *Kai* and *etc.*) as source fonts, we train the style and content encoder on the same font set. We also verify the generalization ability of our method to unseen language glyphs by using Japanese, Korean, and Traditional Chinese characters.

Fig. 5. We visualize the generated font images and compare our method with existing state-of-the-art approaches. We use red boxes to mark the results with erroneous spatial layout, distorted font structure and truncated strokes, and mark the mismatched styles, including stroke, component, and global level of style inconsistency, with blue boxes.

Table 1. We compare our method with existing state-of-the-art approaches on seen and unseen fonts. We highlight the best in **bold** and the second in underline.

Methods	Seen Fonts					Unseen Fonts				
	L1 ↓	RMSE ↓	SSIM ↑	LPIPS ↓	FID ↓	L1 ↓	RMSE ↓	SSIM ↑	LPIPS ↓	FID ↓
FUNIT [24]	0.1192	0.3055	0.5471	0.2018	18.83	0.1194	0.3061	0.5443	0.2063	20.21
MX-Font [33]	0.0854	0.2500	0.6619	0.1351	20.03	0.0884	0.2559	0.6574	0.1352	22.20
DG-Font [43]	0.0833	0.2544	0.6527	0.1274	21.38	0.0851	0.2577	0.6486	0.1298	21.64
CG-GAN [22]	0.1272	0.3218	0.5210	0.1956	27.67	0.1279	0.3224	0.5169	0.1989	29.14
CF-Font [41]	0.0664	0.2223	0.7244	0.1015	18.12	0.0702	0.2295	0.7150	0.1019	20.40
FGTr	**0.0522**	**0.1753**	**0.8015**	**0.0861**	**13.52**	**0.0543**	**0.1801**	**0.7916**	**0.0925**	**14.88**

Evaluation metrics. L1 and RMSE compute pixel-wise difference to measure content consistency. Lower L1 and RMSE indicate less distortion. SSIM computes structural similarity to measure style consistency. Higher SSIM refers to closer writing styles to target fonts. LPIPS [44] is used to quantify perceptual consistency in human vision. Lower LPIPS creates clearer boundaries between characters and background, continuous strokes, and regular spatial arrangements. FID [13] measures discrepancy between generation and real distributions. Lower FID leads to a higher quality and variety of generated images.

4.2 Experimental Results

Implementation details. The detailed structure of FGTr is provided in Appendix. We train FGTr using AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and the weight

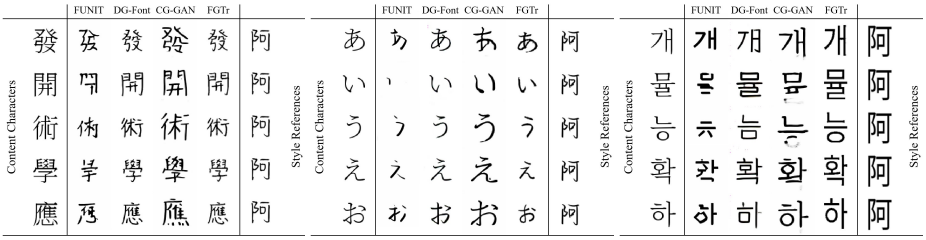

Fig. 6. Generated images of unseen Traditional Chinese, Japanese, and Korean characters. We use content characters as content guidance and style references to represent the target style. Here, we just plot a single image as the style reference for convenience.

Table 2. Ablation studies on different components. T, M, and A represent the **T**ransformer-based model, local self-attention **M**ask, and **A**uxiliary generation module. We highlight the best results in **bold** and the second best ones in underline.

Methods			Seen Fonts					Unseen Fonts				
T	M	A	L1 ↓	RMSE ↓	SSIM ↑	LPIPS ↓	FID ↓	L1 ↓	RMSE ↓	SSIM ↑	LPIPS ↓	FID ↓
✓			0.0552	0.1851	0.7879	0.0839	23.21	0.0575	0.1883	0.7764	0.0870	24.25
✓		✓	0.0527	0.1779	0.8011	**0.0780**	20.44	0.0549	0.1828	**0.7932**	**0.0824**	21.27
✓	✓		0.0548	0.1778	0.7910	0.0926	14.21	0.0567	0.1813	0.7815	0.0999	15.29
✓	✓	✓	**0.0522**	**0.1753**	**0.8015**	0.0861	**13.52**	**0.0543**	**0.1801**	0.7916	0.0925	**14.88**

decay is set to 0.05. The model is trained for 800 epochs with an initial learning rate of $1e-4$, which linearly decays with the number of epochs. We use an image resolution of 128×128 for FGTr and a batch size of 256 in the training stage. Font images generated by other methods are resized to 128×128 to calculate evaluation metrics and visual comparison. During online generation, we provide 10 reference images for one target style in a few-shot setting. A memory bank is built to store the feature sequences extracted by style and content encoders from real samples. AGM leverages these features to enhance the generation quality.

Comparison with state-of-the-art method. We conduct quantitative comparisons with existing state-of-the-art methods on seen and unseen fonts. FUNIT [24] is designed for few-shot image-to-image translation. MX-Font [33] and CG-GAN [22] leverage component-wise supervision for font generation by applying attention mechanism. DG-Font [43] and CF-Font [41] exploit pure CNNs and explore the application boundary of CNNs for FFG. FGTr outperforms other approaches, especially on unseen fonts, referred to Table 1. MX-Font, DG-Font, CG-GAN, and CF-Font, designed for FFG, generate high-quality font images and achieve style and content more consistent than FUNIT, designed for general image translation, as evidenced by better L1, RMSE, SSIM, and LPIPS scores. However, they lack generation diversity, as indicated by worse FID scores. This may be due to the limitation of using fixed source fonts, which prevents existing methods from fully transferring source style to the target while preserving

Fig. 7. The comparison of generated glyphs for CNN-bases FUNIT, DG-Font, and FGTr with or without LSAM. We use red boxes to highlight the generation differences. Convolution operation is beneficial for the local representation of glyphs, but its global attention and long-term dependencies on font images are weaker than attention mechanism, resulting in distorted space layout and poorer style consistency of generated fonts with the target. FGTr retains the advantages of convolution in Transformer based on attention modules with LSAM, and is capable of generating high-quality glyphs.

source content, resulting in a trade-off. FGTr leverages self-attention mechanism to replace convolution operations and stacks Transformer layers to obtain more detailed representations than CNN-based models, which captures local features at fine-grained locations to guide the stylization process. Moreover, the multi-head attention eliminates the need for component-wise supervision, as it can automatically attend to different local regions of the image and extract rich local and global representations. Fig. 5 compares the images generated by different models. For a character with different styles, such as "bu", FGTr generates more consistent stroke thickness and transitions according to style references, which is failed by DG-Font. In general, FGTr generates glyphs with a more reasonable spatial arrangement of components and radicals, and less stroke truncation and blurring. This may be attributed to the content encoder being trained with large-scale samples, to extract style-agnostic content representation.

4.3 Ablation Studies

Ablation study of different components. The results are shown in Table 2, where LSAM and AGM reinforce the generation quality and achieve numerical and visual improvements. LSAM reduces stroke truncation and radical blurring by enhancing the attention of neighboring patches, as displayed in Fig. 7 AGM enhances the visual perception and diversity, and clarifies the edges between characters and white background, reducing unnecessary gray pixels. We exhibit

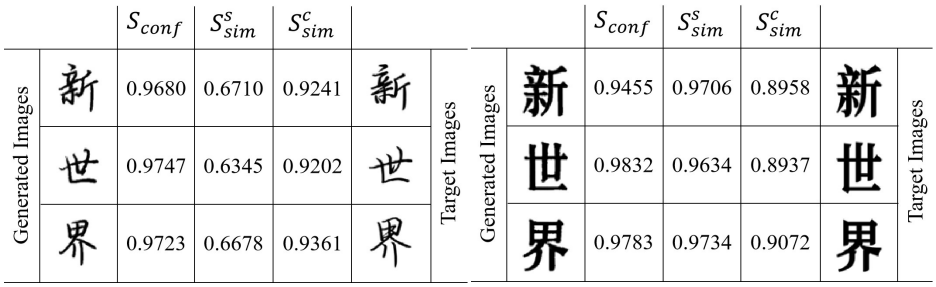

Fig. 8. Model confidence, style and content similarity scores $S_{conf}, S^s_{sim}, S^c_{sim}$ for generated glyphs with two styles and three contents.

the confidence and similarity scores measured by AGM for generated images with different styles and contents in Fig. 8. The flowing and cursive fonts have more difficult writing styles to transfer. The regular and printed fonts lead to more challenges in maintaining content consistency.

Generation to unseen languages. We compare to generate Japanese, Korean, and Traditional Chinese scripts. Traditional Chinese characters are closer to training data than Japanese and Korean, and have more complex strokes and structures. Japanese characters have simple structures but require more details at stroke level. Korean characters are square and focus on the positional relationship between components. As shown in Fig. 6, FUNIT has difficulty maintaining content consistency for unseen characters. DG-Font retains better original content but has flaws in generalizing styles, especially for Japanese. CG-GAN has the problem of missing strokes and wrong stroke thickness. Our FGTr outperforms in generating unseen language systems, fully preserves the strokes and structure of the content, and better transforms to the target style.

5 Conclusion

In this paper, we propose the Font Generation Transformer (FGTr). We introduce the LSAM to enhance attention modules' focus on adjacent patches, thereby improving generation quality. AGM is applied to evaluate confidence, style and content similarity for generated glyphs during online generation. An image with the highest score is selected as the result. Experimental results show that FGTr achieves exciting results in the task of FFG, and the Appendix provides more generated examples.

Acknowledgements

This work was supported by the Natural Science Fund of Hubei Province (Grant 2022CFB823), HUST Independent Innovation Research Fund (Grant 2021XXJS096), and grants from the National Key Lab of MSIIPT (Grant 6142113220309) as well as the MoE Key Lab of Image Processing & Intelligent Control, among others.

References

1. An, J., Huang, S., Song, Y., Dou, D., Liu, W., Luo, J.: Artflow: Unbiased image style transfer via reversible neural flows. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 862–871 (2021)
2. Baek, K., Choi, Y., Uh, Y., Yoo, J., Shim, H.: Rethinking the truly unsupervised image-to-image translation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14154–14163 (2021)
3. Below, V.C.A.L.: Cjkv information processing chinese japanese korean and vietnamese computing
4. Chang, J., Gu, Y., Zhang, Y., Wang, Y.F., Innovation, C.: Chinese handwriting imitation with hierarchical generative adversarial network. In: BMVC. p. 290 (2018)
5. Chen, D., Yuan, L., Liao, J., Yu, N., Hua, G.: Stylebank: An explicit representation for neural image style transfer. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1897–1906 (2017)
6. Deng, Y., Tang, F., Dong, W., Ma, C., Pan, X., Wang, L., Xu, C.: Stytr2: Image style transfer with transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11326–11336 (2022)
7. Deng, Y., Tang, F., Dong, W., Sun, W., Huang, F., Xu, C.: Arbitrary style transfer via multi-adaptation network. In: Proceedings of the 28th ACM international conference on multimedia. pp. 2719–2727 (2020)
8. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
9. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. arXiv preprint arXiv:1610.07629 (2016)
10. Gao, Y., Wu, J.: Gan-based unpaired chinese character image translation via skeleton transformation and stroke rendering. In: proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 646–653 (2020)
11. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2414–2423 (2016)
12. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022)
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. Advances in neural information processing systems **30** (2017)
14. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of the IEEE international conference on computer vision. pp. 1501–1510 (2017)
15. Huang, X., Liu, M.Y., Belongie, S., Kautz, J.: Multimodal unsupervised image-to-image translation. In: Proceedings of the European conference on computer vision (ECCV). pp. 172–189 (2018)
16. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1125–1134 (2017)
17. Jiang, Y., Lian, Z., Tang, Y., Xiao, J.: Dcfont: an end-to-end deep chinese font generation system. In: SIGGRAPH Asia 2017 Technical Briefs, pp. 1–4 (2017)

18. Jiang, Y., Lian, Z., Tang, Y., Xiao, J.: Scfont: Structure-guided chinese font generation via deep stacked networks. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 4015–4022 (2019)
19. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14. pp. 694–711. Springer (2016)
20. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4401–4410 (2019)
21. Kim, T., Cha, M., Kim, H., Lee, J.K., Kim, J.: Learning to discover cross-domain relations with generative adversarial networks. In: International conference on machine learning. pp. 1857–1865. PMLR (2017)
22. Kong, Y., Luo, C., Ma, W., Zhu, Q., Zhu, S., Yuan, N., Jin, L.: Look closer to supervise better: one-shot font generation via component-based discriminator. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 13482–13491 (2022)
23. Li, C., Wand, M.: Precomputed real-time texture synthesis with markovian generative adversarial networks. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14. pp. 702–716. Springer (2016)
24. Liu, M.Y., Huang, X., Mallya, A., Karras, T., Aila, T., Lehtinen, J., Kautz, J.: Few-shot unsupervised image-to-image translation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10551–10560 (2019)
25. Liu, S., Lin, T., He, D., Li, F., Wang, M., Li, X., Sun, Z., Li, Q., Ding, E.: Adaattn: Revisit attention mechanism in arbitrary neural style transfer. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6649–6658 (2021)
26. Liu, W., Liu, F., Ding, F., He, Q., Yi, Z.: Xmp-font: self-supervised cross-modality pre-training for few-shot font generation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 7905–7914 (2022)
27. Liu, Y., Lian, Z.: Fonttransformer: Few-shot high-resolution chinese glyph image synthesis via stacked transformers. Pattern Recognition **141**, 109593 (2023)
28. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021)
29. Lyu, P., Bai, X., Yao, C., Zhu, Z., Huang, T., Liu, W.: Auto-encoder guided gan for chinese calligraphy synthesis. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). vol. 1, pp. 1095–1100. IEEE (2017)
30. Pan, W., Zhu, A., Zhou, X., Iwana, B.K., Li, S.: Few shot font generation via transferring similarity guided global style and quantization local style. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19506–19516 (2023)
31. Park, D.Y., Lee, K.H.: Arbitrary style transfer with style-attentional networks. In: proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5880–5888 (2019)
32. Park, S., Chun, S., Cha, J., Lee, B., Shim, H.: Few-shot font generation with localized style representations and factorization. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 2393–2402 (2021)

33. Park, S., Chun, S., Cha, J., Lee, B., Shim, H.: Multiple heads are better than one: Few-shot font generation with multiple localized experts. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 13900–13909 (2021)
34. Paul, S., Chen, P.Y.: Vision transformers are robust learners. In: Proceedings of the AAAI conference on Artificial Intelligence. vol. 36, pp. 2071–2081 (2022)
35. Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., Dosovitskiy, A.: Do vision transformers see like convolutional neural networks? Advances in Neural Information Processing Systems **34**, 12116–12128 (2021)
36. Sun, D., Ren, T., Li, C., Su, H., Zhu, J.: Learning to write stylized chinese characters by reading a handful of examples. arXiv preprint arXiv:1712.06424 (2017)
37. Sun, D., Zhang, Q., Yang, J.: Pyramid embedded generative adversarial network for automated font generation. In: 2018 24th International Conference on Pattern Recognition (ICPR). pp. 976–981. IEEE (2018)
38. Tang, L., Cai, Y., Liu, J., Hong, Z., Gong, M., Fan, M., Han, J., Liu, J., Ding, E., Wang, J.: Few-shot font generation by learning fine-grained local styles. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 7895–7904 (2022)
39. Tian, Y.: zi2zi: Master chinese calligraphy with conditional adversarial networks. Internet] https://github. com/kaonashi-tyc/zi2zi **3**, 2 (2017)
40. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
41. Wang, C., Zhou, M., Ge, T., Jiang, Y., Bao, H., Xu, W.: Cf-font: Content fusion for few-shot font generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1858–1867 (2023)
42. Wu, S.J., Yang, C.Y., Hsu, J.Y.j.: Calligan: Style and structure-aware chinese calligraphy character generator. arXiv preprint arXiv:2005.12500 (2020)
43. Xie, Y., Chen, X., Sun, L., Lu, Y.: Dg-font: Deformable generative networks for unsupervised font generation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5130–5140 (2021)
44. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
45. Zhang, Y., Zhang, Y., Cai, W.: Separating style and content for generalized style transfer. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8447–8455 (2018)