This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Neural Transformation Fields for Arbitrary-Styled Font Generation

Bin Fu[1], Junjun He[2], Jianjun Wang[1], and Yu Qiao[1,2*]

[1]ShenZhen Key Lab of Computer Vision and Pattern Recognition,
Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
[2]Shanghai Artificial Intelligence Laboratory
[1]{bin.fu, jj.wang2, yu.qiao}@siat.ac.cn, [2]{hejunjun, qiaoyu}@pjlab.org.cn

Abstract

Few-shot font generation (FFG), aiming at generating font images with a few samples, is an emerging topic in recent years due to the academic and commercial values. Typically, the FFG approaches follow the style-content disentanglement paradigm, which transfers the target font styles to characters by combining the content representations of source characters and the style codes of reference samples. Most existing methods attempt to increase font generation ability via exploring powerful style representations, which may be a sub-optimal solution for the FFG task due to the lack of modeling spatial transformation in transferring font styles. In this paper, we model font generation as a continuous transformation process from the source character image to the target font image via the creation and dissipation of font pixels, and embed the corresponding transformations into a neural transformation field. With the estimated transformation path, the neural transformation field generates a set of intermediate transformation results via the sampling process, and a font rendering formula is developed to accumulate them into the target font image. Extensive experiments show that our method achieves state-of-the-art performance on few-shot font generation task, which demonstrates the effectiveness of our proposed model. Our implementation is available at: https://github.com/fubinfb/NTF.

1. Introduction

Generating a new stylized font with a few reference samples, referred as the few-shot font generation (FFG) task, has received considerable attentions due to the academic, commercial, and artistic values, especially for some glyph-rich scripts such as Chinese and Korean. In recent years, the style-content disentanglement paradigm has become the most popular solution for FFG task, which decouples the stylized font images into the font-specific style codes and the character-specific content features. Therefore, the target stylized font will be generated from the carefully-designed decoder via the combination of the style codes from the reference samples and the content embeddings from the standard glyphs. Based on the style representations, existing approaches can be roughly divided into two categories. Early approaches mainly model font style information as global statistic features, and thus utilize the universal style representations to embed such information. Witnessing the fine-grained structure variations and local correlations (such as stoken and component) in font styles, recent approaches further develop component-wise or fine-grained localized style representations to boost FFG performance. However, as shown in Fig. 1, the differences between font styles mainly come from the shape deformation and transformation on the source glyph. Based on this observation, previous approaches may be the sub-optimal solution for FFG task due to the lack of modeling spatial transformation in font generation process.

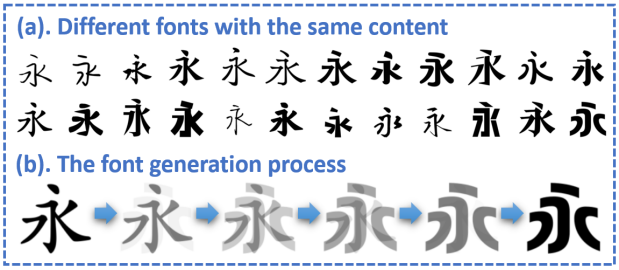

Figure 1. The motivation of this paper: (a). The differences of font styles mainly come from the shape deformation and transformation of the source font, such as the thickness of strokes and the writing pattern of glyphs. (b). We regard font generation as a continuous transformation process from the source font to target font via the creation and dissipation of font pixels.

Inspired by recent advances in Neural Radiance Field (NeRF) [23] in 3D view synthesis, we attempt to embed

*Corresponding author: Yu Qiao

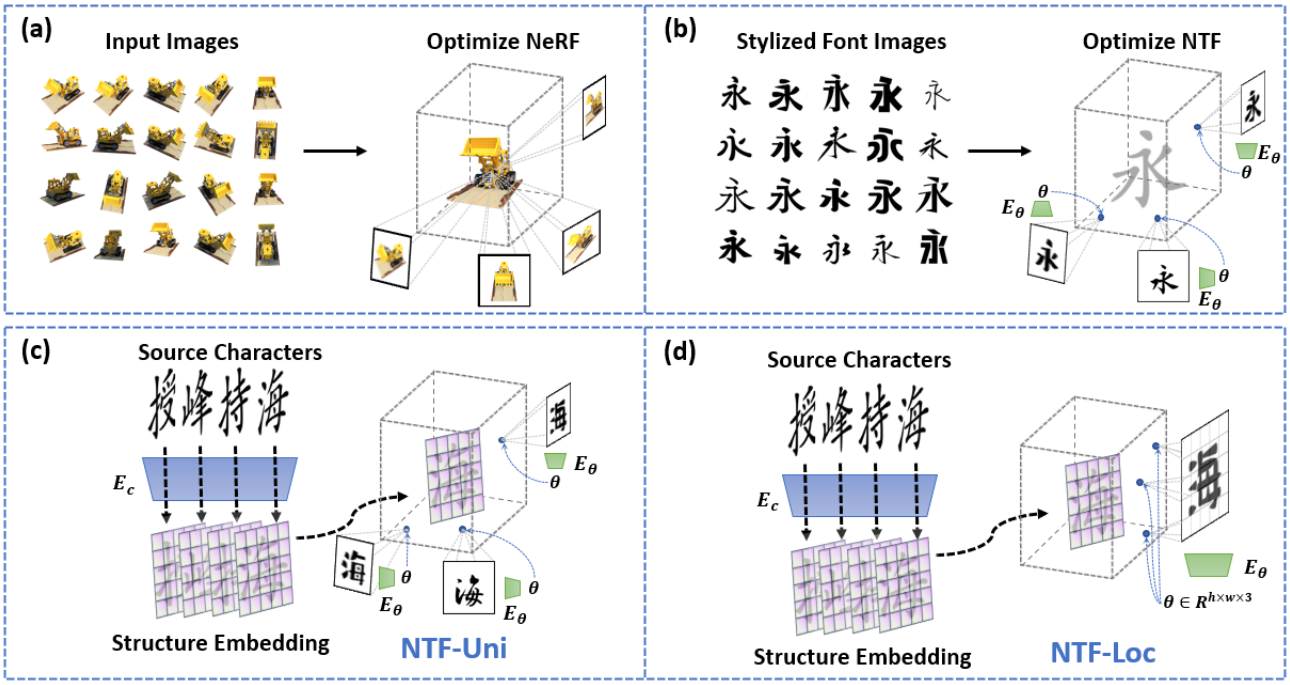

Figure 2. (a). The methodology of NeRF. (b). We embed the desired transformations of font generation into the neural transformation field. The style estimator E_θ predicts the locations θ of each font style, and the font generation process can be viewed as a transformation process of font pixels from the original point to this location. (c). As each NeRF only corresponds to a specific scene, it is impracticable for the FFG task. Thus we generalize our NTF to model the transformations for all characters by introducing the structure embedding (extracted by a structure encoder E_c) of characters. (d). Finally, considering the localized characteristic of font style, we further generalize our NTF into the localized style representation.

the desired spatial transformations in a neural transformation field, and thus the font generation process can be reformulated as the accumulation of a set of intermediate transformation results along a specific path. The methodology of NeRF is presented in Fig. 2 (a). The NeRF constructs a neural radiance field to represent a specific 3D scene as a 5D function, whose inputs are the location and view direction while outputs are the emitted color together with the volume density. An MLP network is utilized to approximate this function, where the scene information is embedded into the parameters via the optimization process. To generate a novel view, the color of each pixel is rendered along the color ray passing through the scene via volume rendering technique [22].

Motivated by the above method, instead of directly predicting the pixel-level deformation offsets, we model the font generation as a continuous transformation process via the creation intensity φ and dissipation rate τ of font pixels, and embed such transformations into a neural transformation filed. To make the description clearly, we use the universal-representation-based font generation to introduce our method. As shown in Fig. 2 (c), the neural transformation field (NTF) is constructed to model the font transformation process based on the structure embeddings of source characters. Each location in NTF represents a specific structure-related transformation and the path from the original point to this location corresponding to the transformation process from the source font to the target font. Each font style has a specific location relating to the desired transformations for generating font images in NTF, and we utilize an estimator to estimate this location. With the estimated location and the corresponding transformation path, NTF generates a set of intermediate transformations via the sampling process, and a font rendering formula is developed to accumulate them into the target font image. Since the font styles contain many fine-grained structures and local correlations, the localized style representation shows significant advantages over the universal style representation. Therefore, as shown in Fig. 2 (d), we generalize our NTF into the localized style representations and conduct extensive experiments to evaluate our model. Experimental results show that our model achieves new state-of-the-art performance in few-shot font generation tasks, both in the seen fonts with unseen contents testing and unseen fonts with unseen contents testing, which demonstrate the superior generation performance of our proposed method.

In summary, our contribution is threefold in this paper:

1). We regard font generation as a continuous transformation process via the creation and dissipation of font pixels along the transformation path, and embed such transfor-

mations into the neural transformation field (NTF).

2). A differentiable font rendering procedure is developed to accumulate the intermediate transformations into the target font image.

3). Experimental results show that our method outperforms the state-of-the-art methods in the few-shot font generation task, which demonstrate the effectiveness of our proposed method.

2. Related Works

In this section, we briefly review the recent progresses in font generation and the development of neural radiance field (NeRF).

2.1. Font Generation

Font generation aims to generate stylized font images from reference samples. Since this task can be viewed as the mapping from the source domain to the target domain, several methods [6, 33, 34] utilize the image-to-image (I2I) translation models [7, 15, 15, 19, 41] to transfer font from source style to target style. Zi2zi [33] and Rewrite [34] are implemented on pix2pix [15] frameworks and optimized in a supervised manner with style labels. DC-font [16] utilizes a feature reconstruction network to embed style information for font synthesis. Moreover, although designed for the I2I task, FUNIT [19] has been modified to perform font generation due to its generalization ability for unseen styles.

In recent years, generating font images with few samples has received more attention. Most few-shot font generation (FFG) methods focus on the style-content disentanglement paradigm, which transfers the target font styles by combining the content representations of source characters with the style codes of reference samples. Existing approaches can be roughly divided into two categories, the universal representations [2, 11, 40] and localized representations [4, 18, 20, 26, 27, 32, 36]. Early approaches mainly model font style information as global statistic features, and thus utilize the universal style representations to embed such information. For example, EMD [40] and AGISNet [11] combine the style vector of reference samples and content vector of source characters to generate the target font images. Witnessing the fine-grained structure deformable and local correlations in font styles, recent approaches further develop localized style representations to boost FFG performance. DM-Font [4] utilizes the compositional script to decompose each glyph into several predefined components, and generates the target font with component-wise features. LF-Font [26] simplifies component-wise styles to the localized style representation by a product of component factor and style factor via a factorization strategy. Since DM-Font and LF-Font both require component labels at test time, MX-Font [27] generalizes the above methods by employing multiple encoders to learn different local concepts via weak supervision from the component classifier. CG-GAN [18] employs a component-wise discriminator to supervise font generator in a fine-grained manner. [32] utilizes a cross-attention mechanism to aggregate style features into the fine-grained style representation.

2.2. Neural Radiation Field

Recently, neural radiance field (NeRF) [23] has become a prevalent method for novel view synthesis with rapid progress, which represents a scene with a neural network via mapping a position and view direction to the emitted color and volume density. Many following works attempt to extend NeRF to the dynamic scene [21, 30, 37], fast training and rendering [8, 12, 31] and scene editing [3, 24, 38]. Besides the scene reconstruction and view synthesis, many follow-up works generalize NeRF to various tasks. Some papers [1, 5, 9, 10] employ NeRF to generate editable facial images. For example, RigNeRF [1] construct a deformation field guided by a 3D morphable face model to enable full control of head pose and facial expressions. StyleSDF [25] combines a Signed Distance Fields (SDF) volume renderer and a 2D style-transfer network to synthesis high-quality facial images. Another promising application of NeRF method is to model human movement and reconstruct dynamic human bodies from video. For example, Animatable NeRF [29] employs NeRF to embed 3D static human bodies and utilizes a deformation field to model body movement by transforming observation-space points to the canonical space. HumanNeRF [35] represents a moving person with a canonical appearance pose warped to an observed pose, modeled by a canonical appearance field and a motion field.

Inspired by the appealing quality of NeRF [23] and its good flexibility with large follow-up extensions, in this paper, we attempt to construct a neural transformation field to embed the desired transformations, and thus reformulate the font generation process as the accumulation of a set of intermediate transformation results along a specific estimated path.

3. Methodology

In this section, we provide a detailed description of our proposed method, including the Neural Transformation Field (NTF) for font generation (in Sec. 3.1), the Font Rendering with NTF (in Sec. 3.2), the overall framework (in Sec. 3.3), and optimization process (in Sec. 3.4).

3.1. Representing Font Generation as the Transformation Process of Font Pixels in NTF

In this paper, we regard font generation as the creation and dissipation processes of font pixels with respect to the source font, and thus reformulate the font style transfer as a continuous transformation process in neural transformation

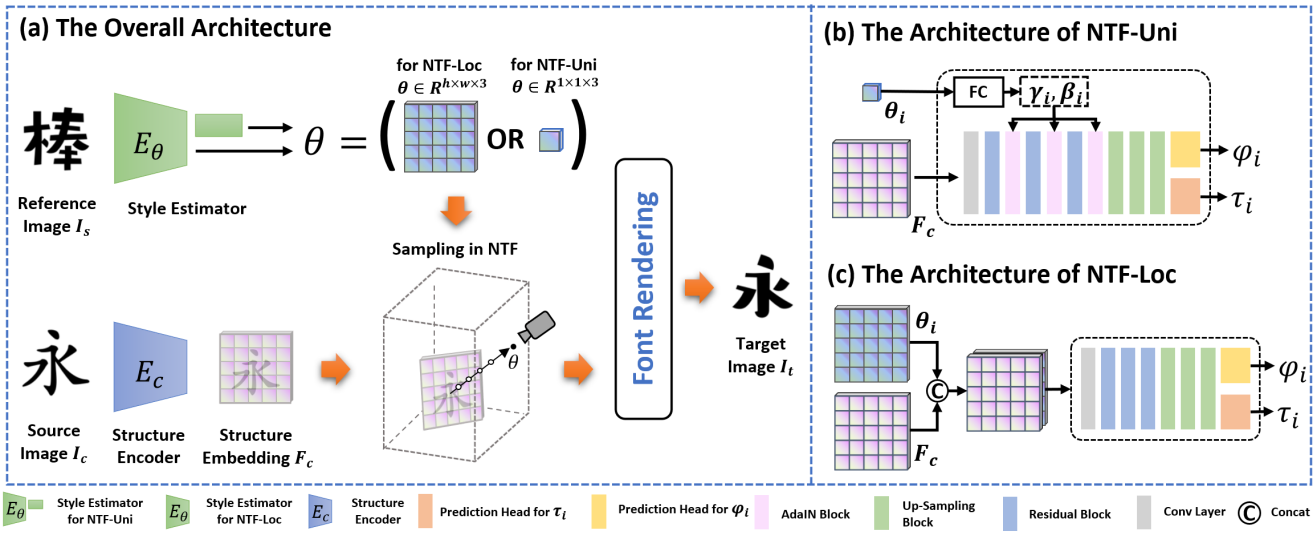

Figure 3. The overall framework of our proposed model, which consists of a style estimator E_θ, a structure encoder E_c, a neural transformation field (NTF), and a discriminator D. Following the previous methods [4,19,26,27], we employ a multi-head projection discriminator in our framework, which is not plotted in this figure.

field (NTF). As shown in Fig. 2 (c), we utilize a style estimator E_θ to express each font style as a 3D location θ in NTF, and define the source font as the original point. Based on this definition, the path from the original point to the location θ stands for a set of desired transformations from the source font to the target font, and the final font image can be expressed as the accumulation of such transformations (creation and dissipation of font pixels) along this path. In this paper, we formulate this accumulation process as a font rendering process, which will be introduced in Sec. 3.2.

As each radiance field only corresponds to a specific scene in NeRF, it is an impracticable solution for the font generation task, thus we generalize our NTF to model the spatial transformation for all characters by introducing the structure embedding F_c of characters in Fig. 2 (c). The inputs of NTF are the 3D location θ together with the structure embedding F_c of the source character, while the outputs are the creation intensity φ and transformation rate τ at this location. In practice, we utilize a convolution neural network (CNN) as the style estimator E_θ to project font styles of reference samples into the locations θ, and implement a CNN-based encoder to embed the character-specific structure information of source characters as the conditions for NTF. We approximate this NTF function as a CNN network F_Θ:

$$(\varphi, \tau) = F_\Theta(\theta | F_c) \tag{1}$$

where φ and τ denote the creation intensity and dissipation rate of the font pixels at location θ conditioned on the structure embedding F_c, respectively. The creation intensity φ and dissipation rate τ are the transformation parameters for font generation, which will be accumulated into final font images in the following section. The NTF will learn the desired transformations of font generation task via the optimization process (in Sec. 3.3) from the training samples, and embed such knowledge into its weight Θ.

As discussed in previous sections, since the font styles contain many fine-grained structures and local correlations, the localized style representation shows significant advantages over the universal style representation in FFG task. Therefore, as shown in Fig. 2 (d), we further generalize our NTF into the localized style representation, termed as NTF-Loc, to track the transformation process and generate the final font images.

3.2. Font Rendering with Neural Transformation Field

To generate the font image, we need to collect the intermediate transformation results and formulate the font generation as a font rendering process in neural transformation field.

The transformations of different font styles can be divided into two categories, the font pixel generation process and dissipation process, respectively. Based on this observation, we introduce two quantities, the creation intensity φ and dissipation rate τ namely, to model above transformations. The creation intensity φ represents the non-font pixel changing to font pixel or the font pixel enhancing its intensity. In contrast, the dissipation rate τ represents a font pixel weakening its intensity with the rate τ at the location ω. The transformed intensity at location ω can be expressed

as the following differential equation:

$$\frac{dI(\omega)}{d\omega} = \varphi(\omega)\tau(\omega) - \tau(\omega)I(\omega). \tag{2}$$

The first term models the creation process while the second term models the dissipation process of font pixels. The solution to this differential equation is:

$$I(\theta) = \int_0^{\theta} \varphi(\omega)\tau(\omega)T(\omega)d\omega. \tag{3}$$

This equation is the font rendering formula in NTF, and $T(\omega)$ represents the accumulated transformations along the path from the location ω to θ, which can be expressed as

$$T(\omega) = \exp\left(-\int_{\omega}^{\theta} \tau(x)dx\right). \tag{4}$$

Based on Eq. 3, generating a stylized font image at the location θ requires estimating this integral from the original point to θ in our neural transformation field. In practice, we estimate this continuous integral numerically. The interval from the original point to location θ is partitioned into N evenly-spaced segments with the length $\xi = \frac{1}{N}\theta$, and we draw one sample in each segment i at the location $\theta_i = i\xi$. Therefore, the integral in Eq. 3 can be approximated by [22]

$$I = \sum_{i=1}^{N} T_i \left(1 - \exp\left(-\tau_i \xi\right)\right) \varphi_i, \tag{5}$$

and T_i can be expressed as

$$T_i = \exp\left(-\sum_{j=i+1}^{N} \tau_i \xi\right), \tag{6}$$

where ξ is the length of each segment i.

3.3. Overall Framework

The overall architecture of our proposed model is shown in Fig. 3, which consists of a style estimator E_θ, a structure encoder E_c, a neural transformation field (NTF), and a discriminator D. Given a style image $I_s \in R^{H\times W\times 1}$ as the reference image and a source image $I_c \in R^{H\times W\times 1}$, the estimator E_θ estimates the 3D location θ from the reference image while the structure encoder network E_c embeds the structure information $F_c \in R^{h\times w\times c}$ from the source character. Then we generate N sampling points $\theta_1, \cdots, \theta_N$ from the path 0 to θ and combine each sampled location θ_i with the structure embedding F_c as the inputs for NTF. Since the current results are not dependent on the outputs of previous steps in the NTF function (Eq. 1), we parallelly calculate NTF function with respect to the different locations. Finally, we collect the predicted creation intensity φ_i and dissipation rate τ_i at all sampled locations θ_i, and utilize Eq. 5 to generate font images. As we discussed before, in this paper, we provide two NTF structures for the universal style representation $\theta \in R^{1\times 1\times 3}$ and localized style representation $\theta \in R^{h\times w\times 3}$, termed as NTF-Uni (Fig. 3 (b)) and NTF-Loc (Fig. 3 (c)), respectively.

NTF for Universal Style Representation: For the universal style representation, the global location $\theta_i \in R^{1\times 1\times 3}$ together with the structure embedding $F_c \in R^{h\times w\times c}$ will be utilized to calculate the predicted creation intensity φ_i and dissipation rate τ_i. We adopt AdaIN mechanism [14,17] to fuse the location $\theta_i \in R^{1\times 1\times 3}$ and the structure embedding $F_c \in R^{h\times w\times c}$ at several intermediate layers of NTF, which can be expressed as

$$F_{out,m} = \gamma_m \left(\frac{F_{in,m} - \mu(F_{in,m})}{\sigma(F_{in,m})}\right) + \beta_m \tag{7}$$

where the location $\theta_i \in R^{1\times 1\times 3}$ will be first projected to the scale γ_m and bias β_m for AdaIN operation via full-connected layers, and then they are utilized to separately normalize each input feature map $f_{in,m}$ via Eq. 7. The details architecture of NTF-Uni is shown in Fig. 3 (b), which employs 3 residual blocks, 3 up-sampling blocks, and 3 AdaIN fusion blocks to generate the feature $F \in R^{H\times W\times 32}$. Finally, two convolution heads are utilized to predict creation intensity $\varphi_i \in R^{H\times W\times 1}$ and dissipation rate $\tau_i \in R^{H\times W\times 1}$.

NTF for Localized Style Representation: For the localized style representation, the estimated location $\theta_i \in R^{h\times w\times 3}$ has the same size in space dimension with the structure embedding $F_c \in R^{h\times w\times c}$, thus we directly concatenate θ_i with F_c as the inputs of NTF to predict creation intensity φ_i and dissipation rate τ_i. As shown in Fig. 3 (c), the architecture of NTF-Loc is modified from NTF-Uni by removing the AdaIN fusion blocks. Finally, two prediction heads are utilized to generate the creation intensity $\varphi_i \in R^{H\times W\times 1}$ and dissipation rate $\tau_i \in R^{H\times W\times 1}$.

3.4. Optimization Process

As shown in Eq. 1 and Eq. 5, the font generation process of our proposed model is naturally differentiable, thus we formulate the optimization process in an end-to-end manner. We utilize the following loss functions to optimize our model:

Reconstruction loss: We adapt an $L1$ loss as the reconstruction loss between the generated font image $\tilde{y}$ and the ground truth image y to learn the desired transformation for our model. We also utilize $L1$ loss to optimize network to reconstruct the source image I_c at the original point in NTF. Therefore, the reconstruction loss can be expressed as

$$L_{rec} = E\left[||y - \tilde{y}||_1 + ||I_c - \tilde{y}_0||_1\right]. \tag{8}$$

where $\tilde{y}_0$ denotes the generated font image at the original point.

Adversarial loss: To synthesize the realistic image, we adapt the widely-used adversarial loss to make our model learn to generate indistinguishable images from real samples. Following the previous methods [4, 19, 26, 27], we employ a multi-head projection discriminator in our framework, and calculate the adversarial loss according to:

$$\begin{aligned} L_{adv}^{D} &= -E_{(y,s,c)\sim p_{data}} \max(0, -1 + D_{s,c}(y)) \\ &\quad -E_{(\tilde{y},s,c)\sim p_{gen}} \max(0, -1 - D_{s,c}(\tilde{y})) \\ L_{adv}^{G} &= -E_{(\tilde{y},s,c)\sim p_{gen}} D_{s,c}(\tilde{y}) \end{aligned} \tag{9}$$

where $\tilde{y}$ is the generated font image, c and s are the corresponding content and style labels, respectively.

Overall objective loss: We optimize our model under all losses mentioned above:

$$\min_{G} \max_{D} \lambda_{adv}(L_{adv}^{D} + L_{adv}^{G}) + \lambda_{rec} L_{rec}, \tag{10}$$

where λ_{adv} and λ_{rec} are the hyperparameters and we empirically set $\lambda_{adv} = 1.0$ and $\lambda_{rec} = 0.1$ in our experiments.

4. Experiments

In this section, we conduct extensive experiments to demonstrate the effectiveness of our proposed model. We first introduce the dataset and evaluation metrics in our experiments. Then we perform ablation study and evaluate the performance of font generation in various settings. Finally, we compare our model with current state-of-the-art models, which verifies the promising performance of our method.

4.1. Implement Details

We implement our model on PyTorch platform [28] with the public-available toolkit [26, 27] and conduct extensive experiments to demonstrate the effectiveness of our proposed model. In our experiments, the size of font images is $H = W = 128$. At the inference stage, we utilize 8 stylized font images as the reference images (8-shot) to generate the target font. More details are provided in Appendix.

4.2. Dataset and Evaluation Metrics

Datasets: To evaluate our proposed method on font generation task, we collect a Chinese font dataset including 403 font style. We randomly select 353 fonts as the training set, and the remaining 50 fonts as the testing set. We employ 214 characters as the unseen contents and randomly sample other 800 characters to optimize our model. To evaluate our model, we construct the Unseen Fonts Unseen Contents (UFUC) testing set, termed as UFUC-test, by combining the 214 unseen characters with the 50 testing fonts. Moreover, we further randomly select 200 characters from the 800 training characters, and then combine them with the 50 testing fonts to evaluate the generation performance on Unseen Fonts Seen Contents (UFSC), termed as UFSC-test.

Table 1. Ablation study on the style representations with the measurements of SSIM, ms-SSIM, LPIPS, FID. We use 15 sampling points ($N = 15$) to evaluate our models. The bold number indicates the best.

Methods	SSIM↑	ms-SSIM↑	LPIPS↓	FID↓
Unseen Fonts and **Seen** Contents				
NTF-Uni	0.6102	0.3489	0.1264	30.46
NTF-Loc	**0.6299**	**0.4246**	**0.0999**	**13.72**
Unseen Fonts and **Unseen** Contents				
NTF-Uni	0.6331	0.3468	0.1283	33.12
NTF-Loc	**0.6533**	**0.4187**	**0.1019**	**15.67**

Evaluation Metrics: In this paper, to quantitative evaluate our proposed method, four commonly used measurements are employed: (1) SSIM and (2) its multi-scale version ms-SSIM are adopt to evaluate the image quality based on the similarity of luminance, contrast, and structure; (3) LPIPS [39] is employed to quantify the perceptual similarity; and (4). FID [13] is calculated to measure the domain distribution between real samples and generated images.

4.3. Ablation Study

4.3.1 Ablation Study on Style Representations

As we discussed in Sec. 3.3, we provide two NTF structures for the font generation task, the universal style representation (termed as NTF-Uni) and the localized style representation (termed as NTF-Loc), respectively. In this section, we use 15 sampling points $N = 15$ to estimate the integral in Eq. 5 and conduct extensive experiments to evaluate the generation performance of two structures. As shown in Tab. 1, the localized style representation performs significantly better than universal style representation in all measurements, which is consistent with the recent development of FFG model, thus we employ the localized style representation (NTF-Loc) as the default model to perform extensive experiments in the following sections.

4.3.2 The Number of Sampling Points for Font Rendering

In our proposed model, we draw N sampling points evenly to numerically estimate the continuous integral in Eq. 5. To verify the correctness and effectiveness of our model, we conduct extensive experiments on various sampling points N. The experimental results are present in Tab. 2, and we can draw the following conclusions: (1). Since the base model is the traditional font generation process, the

Table 2. Ablation study on the number of sampling points for rendering font images.

Num.	SSIM↑	ms-SSIM↑	LPIPS↓	FID↓
Unseen Fonts and **Seen** Contents				
Base	0.6189	0.3674	0.1114	18.74
$N = 5$	0.6229	0.3997	0.1035	15.35
$N = 10$	0.6280	0.4159	0.1017	14.14
$N = 15$	**0.6299**	**0.4246**	**0.0999**	**13.72**
$N = 20$	0.6260	0.4160	0.1012	16.38
Unseen Fonts and **Unseen** Contents				
Base	0.6413	0.3649	0.1156	19.05
$N = 5$	0.6466	0.3938	0.1065	17.86
$N = 10$	0.6503	0.4107	0.1046	16.55
$N = 15$	**0.6533**	**0.4187**	**0.1019**	**15.67**
$N = 20$	0.6473	0.4035	0.1049	18.10

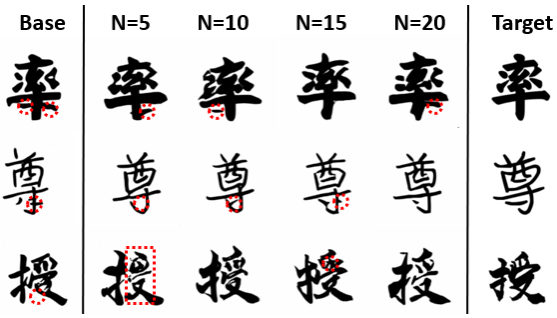

Figure 4. The visualization results of the generated font images with respect to the different sampling numbers N. We highlight the anomalous font pixels and structures by the colored boxes.

improvements from the base model to the $N = 5$ case verify the effectiveness of our NTF. (2). As we expected, increasing sampling points in the proper range is beneficial for the generation performance, which demonstrates that our NTF has indeed learned the desired transformation process for font generation task. (3). However, with the sampling points further increasing, the generation performance begins to decrease (at $N = 20$ in our experiments), which may come from the noise accumulation in the font rendering process (the integral in Eq. 3).

We present the visualization results of the generated font images with respect to the different sampling numbers N in Fig. 4. From this figure, we find that a small number of sampling points is enough to generate stylized font images. With the sampling number increasing, our model can gradually improve the quality of generated font images by (a) recovering the broken structures, (b) removing the anomalous pixels, or (c) strengthening the fine-grained structures. Moreover, as we have discussed in the previous paragraph, with more sampling points, some unexpected font pixels appear in the final image, which leads to performance degradation. To further verify our model, we visualize the transformation process and present the intermediate rendered results along the integral path in Fig. 5. As shown in Fig. 5, the generated font image gradually transforms from the source style to the target style, where the unrelated font pixels in source images are dissipated while other pixels of target font are gradually created. Moreover, for the creation process, the skeleton of the font is first generated, then the fine-grained details are added and the structures of the character become smooth and complete.

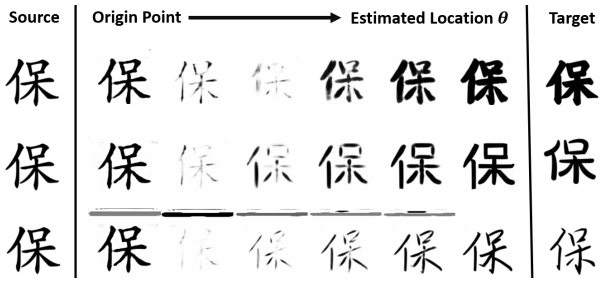

Figure 5. The visualization results of the intermediate rendering process from original point to the estimated location along the integral path.

4.4. Comparison with the State-of-the-art Methods

To further demonstrate the superior performance of our proposed model, we compare our method with recent state-of-the-art works.

4.4.1 Comparison Methods

We compare our model with five state-of-the-art methods, including FUNIT [19], DG-Font [36], LF-Font [26], FS-Font [32], and MX-Font [27]. Following the previous practices [26, 27], we modify and implement FUNIT to perform few-shot font generation on our dataset. The DG-Font [36] models the shape deformable of font generation via the deformation skip connection, which can be served as a comparison baseline for shape transformation models. LF-Font [26] constructs the localized style representation by a product of component factor and style factor via a factorization strategy. MX-Font [27] employs multiple encoders to learn different local concepts via weak supervision of the component classifier. FSFont [32] utilizes the cross attention to aggregate fine-grained local styles. For a fair comparison, we train the above methods from scratch on our dataset and utilize the same reference images in inference stage.

4.4.2 Quantitative Comparison

The quantitative experimental results are present in Tab. 3, from which we can draw the following conclusions: (1). Compared with the universal style representation method,

Figure 6. Visual comparisons of our NTF-Loc with other state-of-the-art methods on our dataset.

Table 3. Performance comparison on few-shot font generation task.

Methods	SSIM↑	ms-SSIM↑	LPIPS↓	FID↓
Unseen Fonts and **Seen** Contents				
FUNIT [19]	0.5990	0.2565	0.1438	15.12
DG-font [36]	0.6124	0.3784	0.1262	64.25
LF-font [26]	0.6208	0.3495	0.1238	36.37
FSFont [32]	0.6358	0.4123	0.1100	62.73
MX-font [27]	0.6053	0.3474	0.1054	25.80
NTF-Loc (Ours)	**0.6299**	**0.4246**	**0.0999**	**13.72**
Unseen Fonts and **Unseen** Contents				
FUNIT [19]	0.6174	0.2651	0.1505	19.29
DG-font [36]	0.6433	0.3924	0.1293	70.90
LF-font [26]	0.6466	0.3052	0.1277	41.21
FSFont [32]	0.6463	0.4051	0.1188	66.49
MX-font [27]	0.6368	0.3676	0.1075	29.34
NTF-Uni (Ours)	0.6331	0.3468	0.1283	33.12
NTF-Loc (Ours)	**0.6533**	**0.4187**	**0.1019**	**15.67**

our NTF-Uni performs better than FUNIT in terms of SSIM and LPIPS. (2). With the help of localized style representation, our NTF-Loc has 0.0202, 0.0264, and 17.45 improvements than NTF-Uni in terms of SSIM, LPIPS, and FID, respectively. (3). Our NTF-Loc achieves the best generation performance in all evaluation metrics, which demonstrates the powerful generation ability of our model.

4.4.3 Qualitative Comparison

To qualitatively evaluate our model, we visualize the generated font images under the UFUC setting in Fig. 6. As shown in this figure, FUNIT and LF-Font often produce font images with incomplete structures. DG-Font, FSFont and MX-Font usually lose some local details of characters, and contain some artifacts near the character region. Our NTF-Loc generates high-quality font images than other state-of-the-art methods, which have better structure completeness and style consistency. Finally, compared with target font images, we find that generating unseen stylized font is still a challenging task, especially for the styles with large deformation. More qualitative results are provided in Appendix.

5. Conclusion

In this paper, we develop a novel approach for the few-shot font generation (FFG) task, which regards the font generation process as a continuous transformation via the creation and dissipation of font pixels. We construct a neural transformation field (NTF) to embed such transformations, where each location in NTF represents a specific structure-related transformation. The path from the origin point to this location corresponds to the transformation process from the source font to the target font. A style estimator is implemented to estimate the location of font styles from the reference image, while a structure encoder is employed to extract structure information from the source image. With the estimated location, a set of sampling points are generated to calculate the intermediate transformations. Then we accumulate these intermediate results to generate the final font images via the proposed font rendering formula. Extensive experiments demonstrate the effectiveness of our proposed model.

6. Acknowledgements

This work was supported in part by the National Key R&D Program of China (NO. 2022ZD0160100), the National Natural Science Foundation of China under Grant (62272450), the Joint Lab of CAS-HK, in part by the Shanghai Committee of Science and Technology (Grant No. 21DZ1100100).

References

[1] ShahRukh Athar, Zexiang Xu, Kalyan Sunkavalli, Eli Shechtman, and Zhixin Shu. Rignerf: Fully controllable neural 3d portraits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20364–20373, 2022. 3

[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. 3

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 3

[4] Junbum Cha, Sanghyuk Chun, Gayoung Lee, Bado Lee, Seonghyeon Kim, and Hwalsuk Lee. Few-shot compositional font generation with dual memory. In *European Conference on Computer Vision*, pages 735–751. Springer, 2020. 3, 4, 6

[5] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022. 3

[6] Jie Chang, Yujun Gu, Ya Zhang, Yan-Feng Wang, and CM Innovation. Chinese handwriting imitation with hierarchical generative adversarial network. In *BMVC*, page 290, 2018. 3

[7] 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

[8] Forrester Cole, Kyle Genova, Avneesh Sud, Daniel Vlasic, and Zhoutong Zhang. Differentiable surface rendering via non-differentiable sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6088–6097, 2021. 3

[9] Yu Deng, Jiaolong Yang, Jianfeng Xiang, and Xin Tong. Gram: Generative radiance manifolds for 3d-aware image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10673–10683, 2022. 3

[10] Guy Gafni, Justus Thies, Michael Zollhofer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8649–8658, 2021. 3

[11] 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. 3

[12] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14346–14355, 2021. 3

[13] 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

[14] 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. 5

[15] 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

[16] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. Dcfont: an end-to-end deep chinese font generation system. In *SIGGRAPH Asia 2017 Technical Briefs*, pages 1–4. 2017. 3

[17] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 5

[18] Yuxin Kong, Canjie Luo, Weihong Ma, Qiyuan Zhu, Shenggao Zhu, Nicholas Yuan, and Lianwen Jin. 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*, pages 13482–13491, 2022. 3

[19] 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, 4, 6, 7, 8

[20] Wei Liu, Fangyue Liu, Fei Ding, Qian He, and Zili Yi. 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*, pages 7905–7914, 2022. 3

[21] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 3

[22] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 2, 5

[23] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, pages 405–421, 2020. 1, 3

[24] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021. 3

[25] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. Stylesdf: High-resolution 3d-consistent image and geometry generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13503–13513, 2022. 3

[26] Song Park, Sanghyuk Chun, Junbum Cha, Bado Lee, and Hyunjung Shim. Few-shot font generation with localized style representations and factorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2393–2402, 2021. 3, 4, 6, 7, 8

[27] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13900–13909, 2021. 3, 4, 6, 7, 8

[28] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 6

[29] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14314–14323, 2021. 3

[30] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 3

[31] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021. 3

[32] Licheng Tang, Yiyang Cai, Jiaming Liu, Zhibin Hong, Mingming Gong, Minhu Fan, Junyu Han, Jingtuo Liu, Errui Ding, and Jingdong Wang. Few-shot font generation by learning fine-grained local styles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7895–7904, 2022. 3, 7, 8

[33] Yuchen Tian. zi2zi: Master chinese calligraphy with conditional adversarial networks. 3

[34] Yuchen Tian. Rewrite: Neural style transfer for chinese fonts. 2016. 3

[35] Chung-Yi Weng, Brian Curless, Pratul P Srinivasan, Jonathan T Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16210–16220, 2022. 3

[36] 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. 3, 7, 8

[37] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021. 3

[38] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18353–18364, 2022. 3

[39] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6

[40] 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. 3

[41] 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