Neural Transformation Fields for Arbitrary-Styled Font Generation
Supplementary Materials

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

1. Font Rendering Formulation Derivation

In this section, we provide the derivation of our font render formulations and the corresponding approximation equations. We model the font transformation process in neural transformation field (NTF) via the creation intensity φ and dissipation rate τ. The transformed intensity at location ω can be expressed as:

$$\frac{dI(\omega)}{d\omega} = \varphi(\omega)\tau(\omega) - \tau(\omega)I(\omega). \tag{1}$$

The first term models the creation process while the second term models the dissipation process of font pixels. To solve this equation, we bring the second term to the left hand side and multiple the integrating factor $\exp\left(\int_0^\omega \tau(t)dt\right)$ to the both sides:

$$\left(\frac{dI(\omega)}{d\omega} + \tau(\omega)I(\omega)\right)\exp\left(\int_0^\omega \tau(t)dt\right) = \varphi(\omega)\tau(\omega)\exp\left(\int_0^\omega \tau(t)dt\right), \tag{2}$$

which can be expressed as

$$\frac{d}{d\omega}\left(I(\omega)\exp\left(\int_0^\omega \tau(t)dt\right)\right) = \varphi(\omega)\tau(\omega)\exp\left(\int_0^\omega \tau(t)dt\right). \tag{3}$$

Integrating this equation from the original point $\omega = 0$ to the estimated location $\omega = \theta$, we have

$$I(\theta)\exp\left(\int_0^\theta \tau(t)dt\right) - I_0 = \int_0^\theta \varphi(\omega)\tau(\omega)\exp\left(\int_0^\omega \tau(t)dt\right)d\omega. \tag{4}$$

Thus the $I(\theta)$ can be expressed as

$$I(\theta) = I_0\exp\left(-\int_0^\theta \tau(t)dt\right) + \int_0^\theta \varphi(\omega)\tau(\omega)\exp\left(-\int_\omega^\theta \tau(t)dt\right)d\omega. \tag{5}$$

Since the font pixels are generated and transformed from the original point $\omega = 0$ to the estimated location $\omega = \theta$ in our model, the first term can be ignored, which leads to

$$I(\theta) = \int_0^\theta \varphi(\omega)\tau(\omega)\exp\left(-\int_\omega^\theta \tau(t)dt\right)d\omega. \tag{6}$$

*Corresponding author: Yu Qiao

With the definition $T(\omega) = \exp\left(-\int_{\omega}^{\theta} \tau(x)dx\right)$, we can arrive

$$I(\theta) = \int_0^{\theta} \varphi(\omega)\tau(\omega)T(\omega)d\omega \tag{7}$$

Based on Eq. 7, 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 Eq. 7 in the segment i can be approximated by

$$\begin{aligned}
I_i &= \int_{\theta_i}^{\theta_{i+1}} \tau_i \varphi_i \exp\left(-\int_{\omega}^{\theta} \tau(x)dx\right) d\omega \\
&= \tau_i \varphi_i \int_{\theta_i}^{\theta_{i+1}} \exp\left(-\int_{\omega}^{\theta_{i+1}} \tau(x)dx\right) \exp\left(-\int_{\theta_{i+1}}^{\theta} \tau(x)dx\right) d\omega \\
&= \tau_i \varphi_i \exp\left(-\int_{\theta_{i+1}}^{\theta} \tau(x)dx\right) \int_{\theta_i}^{\theta_{i+1}} \exp\left(-\int_{\omega}^{\theta_{i+1}} \tau(x)dx\right) d\omega \\
&\approx \tau_i \varphi_i \exp\left(-\int_{\theta_{i+1}}^{\theta} \tau(x)dx\right) \left.\frac{\exp\left(-\tau_i(\theta_{i+1} - \omega)\right)}{\tau_i}\right|_{\theta_i}^{\theta_{i+1}} \\
&= T_i \left(1 - \exp\left(-\tau_i \xi\right)\right) \varphi_i,
\end{aligned} \tag{8}$$

with

$$T_i = \exp\left(-\sum_{j=i+1}^{N} \tau_i \xi\right). \tag{9}$$

Therefore, the integral in Eq. 7 can be approximated by

$$I = \sum_{i=1}^{N} T_i \left(1 - \exp\left(-\tau_i \xi\right)\right) \varphi_i, \tag{10}$$

2. Implement Details

2.1. Network Architectures

As shown in Fig. 3 (b) and (c) of main body, the architectures of the neural transformation field (NTF) are built up by Conv Blocks, Residual Blocks, Up-Sampling Blocks, Down-Sampling Block, and AdaIN Blocks. The detailed structures of such blocks are illustrated in Fig. 1.

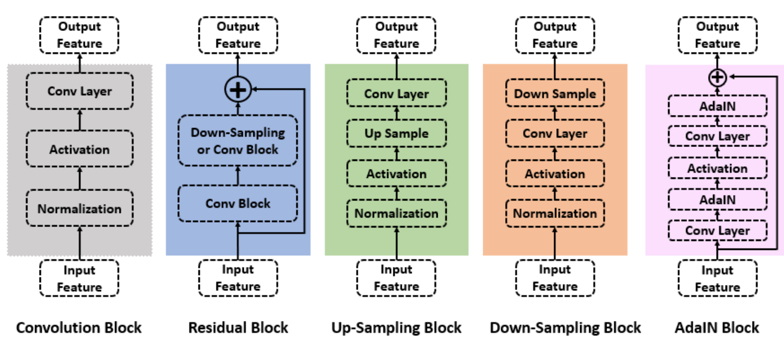

Figure 1. The detailed structures of Conv Blocks, Residual Blocks, Up-Sampling Blocks, Down-Sampling Block, and AdaIN Blocks in our paper. We implement the instance normalization (IN) and ReLU as the normalization operation and activation function, respectively.

Based on the above blocks, we construct our entire font generation model, and the details architectures of the style estimator E_θ, structure encoder E_c, and neural transformation field are present in Tab. 1.

Style Estimator E_θ for Localized Style Representation							
Layer Type	Normalization	Activation	Paddding	Kernel Size	Stride	Downsample	Output Feature
Convolution Block	-	-	1	3	1	-	32
Down-Sampling Block	IN	ReLU	1	3	1	AvgPool	64
Convolution Block	IN	ReLU	1	3	1	-	64
Down-Sampling Block	IN	ReLU	1	3	1	AvgPool	128
Convolution Block	IN	ReLU	1	3	1	-	128
Residual Block $\times 3$	IN	ReLU	1	3	1	-	128
Residual Block	IN	ReLU	1	3	1	AvgPool	256
Residual Block $\times 2$	IN	ReLU	1	3	1	-	256
Residual Block	IN	ReLU	1	3	1	-	128
Residual Block	IN	ReLU	1	3	1	-	64
Residual Block	IN	ReLU	1	3	1	-	32
Residual Block	IN	ReLU	1	3	1	-	3

Structure Encoder E_c							
Layer Type	Normalization	Activation	Paddding	Kernel Size	Stride	Downsample	Output Feature
Convolution Block	-	-	1	3	1	-	32
Down-Sampling Block	IN	ReLU	1	3	1	AvgPool	64
Convolution Block	IN	ReLU	1	3	1	-	64
Down-Sampling Block	IN	ReLU	1	3	1	AvgPool	128
Convolution Block	IN	ReLU	1	3	1	-	128
Residual Block $\times 3$	IN	ReLU	1	3	1	-	128
Residual Block	IN	ReLU	1	3	1	AvgPool	256
Residual Block $\times 2$	IN	ReLU	1	3	1	-	256

Neural Transformation Field for Localized Style Representation (NTF-Loc)							
Layer Type	Normalization	Activation	Paddding	Kernel Size	Stride	Upsample	Output Feature
Convolution Layer	-	-	0	1	1	-	256
Residual Block $\times 3$	IN	ReLU	1	3	1	-	256
Up-Sampling Block	IN	ReLU	1	3	1	Nearest	128
Up-Sampling Block	IN	ReLU	1	3	1	Nearest	64
Up-Sampling Block	IN	ReLU	1	3	1	Nearest	32
Prediction Head for Creation Intensity φ							
Convolution Layer	-	-	1	3	1	-	32
Convolution Layer	-	-	1	3	1	-	1
Output Layer	-	Tanh	-	-	-	-	1
Prediction Head for Dissipation Rate τ							
Convolution Layer	-	-	1	3	1	-	32
Convolution Layer	-	-	1	3	1	-	1
Output Layer	-	Sigmoid	-	-	-	-	1

Table 1. Architecture of the style estimator E_θ, structure encoder E_c, and neural transformation field for Localized Style Representation. IN denotes instance normalization.

2.2. Font Rendering Process

In this section, we provide the pseudo-code of font rendering process for Localized Style Representation (NTF-Loc) in Algorithm 1.

Algorithm 1: Font Rendering Process for Localized Style Representation

Data: the estimated location θ, the structure embedding F_c, the number of sampling points N

Result: the target font image I_t

$\xi \leftarrow \frac{1}{N}\theta$;

for $i \leftarrow N$ **to** 1 **do**

  $\theta_i \leftarrow i\xi$;

  $(\varphi_i, \tau_i) \leftarrow NTF(F_c \copyright \theta_i)$;

  if $i = N$ **then**

    $T_i \leftarrow 1$;

    $\tilde{T}_i \leftarrow \exp(-\tau_i \xi)$;

    $I_t \leftarrow T_i \left(1 - \tilde{T}_i\right) \varphi_i$;

  else

    $T_i \leftarrow T_{i+1}\tilde{T}_{i+1}$;

    $\tilde{T}_i \leftarrow \exp(-\tau_i \xi)$;

    $I_t \leftarrow I_t + T_i \left(1 - \tilde{T}_i\right) \varphi_i$;

  end

end

In practice, since the current φ_i and τ_i are not dependent on the previous steps in the NTF function, we parallelly calculate φ_i and τ_i with respect to the different locations θ_i. For details, please refer to our released code.

2.3. Optimization Details

We utilize Adam optimizer [2] to optimize our proposed method. The learning rates of style estimator E_θ, structure encoder E_c, and neural transformation field (NTF) are $\mathrm{lr}_{\mathrm{Base}} = 4 \times 10^{-4}$, while the learning rate of discriminator is $\mathrm{lr}_{\mathrm{D}} = 2 \times 10^{-3}$. The weights of the whole network are initialized by Kaiming initialization [1]. We set hyperparameters of the loss function as $\lambda_{adv} = 1.0$ and $\lambda_{rec} = 0.1$. Our model is optimized for $800k$ iterations on a single NVIDIA RTX 3090 GPU.

3. Additional Experimental Results

In this section, we provide more quantitative and qualitative results to demonstrate the effectiveness of our method.

3.1. Quantitative Comparison

In this section, we compare our NTF with other SOTA methods in terms of inference running time (FPS), number of parameters N_p, and computation complexity (MACs) for generating 128×128 font images. The quantitative experimental results are present in Tab. 2, from which we can draw the following conclusions: (1). In our NTF, as the sampling points increase, the number of parameters (N_p) remain constant, while inference running time and computation complexity will increase. (2). Compared with MX-font, our NTF achieves higher performance with less N_p, less MACs, and high FPS.

Table 2. Quantitative comparison in inference running time (FPS), number of parameters (N_p), and computation complexity (MACs).

Methods	FPS	N_p	MACs
FUNIT [3]	175	29.77M	21.01G
DG-font [6]	128	16.25M	23.99G
LF-font [4]	107	7.92M	24.79G
MX-font [5]	45	22.76M	51.12G
STF-Loc (N=5)	112	9.07M	24.78G
STF-Loc (N=15)	91	9.07M	43.64G

3.2. One-shot Font Generation

In this paper, our proposed Neural Transformation Field (NTF) is a general font generation method, which can perform both one-shot and few-shot font generation tasks. In Tab. 3 of main body, we evaluate our NTF in few-shot setting. In this section, we utilize our method to perform one-shot font generation task and compare the experimental results with CG-GAN.

Table 3. Performance comparison on one-shot font generation task under UFUC setting.

Methods	SSIM↑	ms-SSIM↑	LPIPS↓	FID↓
Unseen Fonts and **Unseen** Contents				
CG-GAN [3]	0.6812	0.3755	0.1825	40.67
NTF-Loc (Ours)	0.6429	0.3889	0.1225	23.15

As shown in Tab. 3, our method is better than CG-GAN in ms-SSIM, LPIPS, FID, showing that NTF can generate promising results in one-shot font generation task.

3.3. The Creation and Dissipation in Font Transformation Process

In this section, we plot the source image, intermediate rendered image, and target image into a single figure to better present the creation and dissipation of font pixels in our method. As shown in Fig. 2, the first line displays the source image, target image, and the intermediate rendered images. We re-draw them to better display the creation and dissipation process in the second line. Specifically, we re-color the source image and target (ground truth) image in red and blue, respectively. Then we change the transparency of the intermediate rendered images and move them on top of the source and target images. Based on these operations, the transformation process in our proposed model can be better observed.

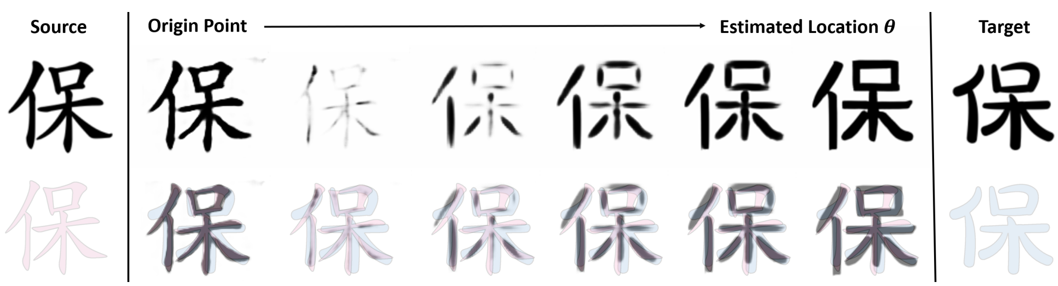

Figure 2. The visualization results for the creation and dissipation in font transformation process.

3.4. Visualization Results

In this section, we provide more visualization results to further verify the effectiveness of our proposed model. The font images are generated by our NTF-Loc model, and 8 font images are utilized as the reference images. The visualization results for Unseen Fonts and Unseen Contents (UFUC-test) are present in Fig. 3. The visualization results for Unseen Fonts and Seen Contents (UFSC-test) are shown in Fig. 4.

Figure 3. Additional visualization results for Unseen Fonts and Unseen Contents generated by our model. We utilize 8 font images as the reference images (8-shot) to generate the target font.

Figure 4. Additional visualization results for Unseen Fonts and Seen Contents generated by our model. We utilize 8 font images as the reference images (8-shot) to generate the target font.

4. Limitation

There are several limitations of our NTF: (1). As shown in Tab. 2, the computation complexity will increase and the inference running time will decrease, when we utilize more sampling numbers to perform font rendering. This is a common weakness in NeRF-based generation models. Fortunately, in our method, we only need few sampling points to generate the promising font images, thus our NTF satisfies the real-time requirement in font generation task. (2). Since only few reference samples are provided and does not cover all strokes, few-shot font generation task is still a difficult task in computer vision community. Some local details of generated images are imperfect. Moreover, as shown in Fig. 5, for the fancy font, although the overall structure and writing pattern can be modeled, the specific ornamentations and local details are missing.

Figure 5. Visualization results on fancy font.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 4

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[3] 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. 4, 5

[4] 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. 4

[5] 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. 4

[6] Yangchen Xie, Xinyuan Chen, Li Sun, and Yue Lu. Dg-font: Deformable generative networks for unsupervised font generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5130–5140, 2021. 4