Rethinking Glyph Spatial Information in Font Generation

Supplementary Material

A. Derivations

We first introduce the derivations of the following equations presented in the main paper:

$$\begin{bmatrix} T_x \\ T_y \end{bmatrix} = \frac{H}{2} \cdot \left(\begin{bmatrix} 1 - \mathrm{F}_{\mathrm{scale}} \\ 1 + \mathrm{F}_{\mathrm{scale}} \end{bmatrix} - \begin{bmatrix} 0 \\ 2 \cdot \mathrm{B}_{\mathrm{offset}} \end{bmatrix} \right) \tag{1}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \frac{EM}{\mathrm{F}_{\mathrm{scale}} \cdot H} \cdot \begin{bmatrix} x - T_x \\ T_y - y \end{bmatrix} \tag{2}$$

$$\frac{\partial \mathcal{L}}{\partial I_{i,j}} = \underbrace{\sum_{p,q} \frac{\partial \mathcal{L}}{\partial Y_{p,q}} \cdot \mathcal{K}_\sigma(p-i, q-j)}_{\text{Broadcasting term}} + \underbrace{\frac{\partial \mathcal{L}}{\partial Y_{i,j}}}_{\text{Direct term}} \tag{3}$$

A.1. The Derivation of Eq. (1)

As shown in Fig. 2 in the main paper, rendering involves a transformation from coordinate system $\mathcal{G}$ to $\mathcal{R}$. We define the coordinates of the origin of $\mathcal{G}$ in $\mathcal{R}$ as (T_x, T_y). For a rendered target image with resolution H^2, we also define the scaling ratio $\mathrm{F}_{\mathrm{scale}}$ as the proportion of glyph size relative to image size, where $\mathrm{F}_{\mathrm{scale}} \leq 1$. The baseline offset ratio $\mathrm{B}_{\mathrm{offset}}$ compensates for the portion of glyphs extending below the baseline (i.e., the horizontal line at $y = 0$ in $\mathcal{G}$). To center the glyphs horizontally within the image, the equation is:

$$T_x = H - (T_x + H \cdot \mathrm{F}_{\mathrm{scale}}) \tag{4}$$

Thus, we obtain:

$$T_x = \frac{H}{2} \cdot (1 - \mathrm{F}_{\mathrm{scale}}) \tag{5}$$

Similarly, to achieve vertical alignment, we construct an equation for T_y, where T'_y is introduced as an intermediate variable representing the baseline offset:

$$\begin{cases} H - T'_y = T'_y - H \cdot \mathrm{F}_{\mathrm{scale}} \\ T_y = T'_y - H \cdot \mathrm{B}_{\mathrm{offset}} \end{cases} \tag{6}$$

Hence:

$$T_y = \frac{H}{2} \cdot (1 + \mathrm{F}_{\mathrm{scale}}) - H \cdot \mathrm{B}_{\mathrm{offset}} \tag{7}$$

Combining Eq. (5) and Eq. (7) yields Eq. (1).

A.2. The Derivation of Eq. (2)

After vectorization, the vector contours described by $\mathcal{V}$ are obtained, whose origin and direction align with those of $\mathcal{R}$, differing only in their units. Since we preserve the origin coordinates (T_x, T_y) of $\mathcal{G}$ in $\mathcal{R}$ and the scaling ratio $\mathrm{F}_{\mathrm{scale}}$ during rendering, we use the basis vector method to get the transformation from $\mathcal{V}$ to $\mathcal{R}$ for a control point coordinates $P(x, y)$. The basis vectors for $\mathcal{V}$ are:

$$\begin{cases} \mathbf{e}_{x,v} = (1, 0) \\ \mathbf{e}_{y,v} = (0, 1) \end{cases} \tag{8}$$

The basis vectors of $\mathcal{G}$ in the representation over $\mathcal{V}$:

$$\begin{cases} \mathbf{e}_{x,g} = \left(\dfrac{\mathrm{F}_{\mathrm{scale}} \cdot H}{EM}, 0 \right) \\ \mathbf{e}_{y,g} = \left(0, -\dfrac{\mathrm{F}_{\mathrm{scale}} \cdot H}{EM} \right) \end{cases} \tag{9}$$

Where EM is the standard unit in the font design space that specifies the coordinate range of a glyph. With $P(x, y)$ expressed in $\mathcal{V}$ and $P(x', y')$ expressed in $\mathcal{G}$, we have:

$$\begin{cases} \overrightarrow{O_{\mathcal{V}} P} = x \cdot \mathbf{e}_{x,v} + y \cdot \mathbf{e}_{y,v} \\ \overrightarrow{O_{\mathcal{G}} P} = x' \cdot \mathbf{e}_{x,g} + y' \cdot \mathbf{e}_{y,g} \end{cases} \tag{10}$$

It follows that:

$$\begin{aligned} \overrightarrow{O_{\mathcal{G}} P} &= \overrightarrow{O_{\mathcal{V}} P} - \overrightarrow{O_{\mathcal{V}} O_{\mathcal{G}}} \\ &= (x, y) - (T_x, T_y) \end{aligned} \tag{11}$$

Hence:

$$\begin{cases} x' = \dfrac{EM}{\mathrm{F}_{\mathrm{scale}} \cdot H} \cdot (x - T_x) \\ y' = -\dfrac{EM}{\mathrm{F}_{\mathrm{scale}} \cdot H} \cdot (y - T_y) \end{cases} \tag{12}$$

Expressed in vector form, we obtain Eq. (2).

A.3. The Derivation of Eq. (3)

Given the GBM definition provided in Sec. 3.3.2 of the main paper, where $\mathcal{B}_\sigma(\cdot)$ denotes a Gaussian blur operation:

$$\mathrm{GBM}(I) = \mathcal{B}_\sigma(I) + \big(I - \mathcal{B}_\sigma(I).\mathrm{detach}()\big) \tag{13}$$

During forward computation, since $\mathcal{B}_\sigma(I).\mathrm{detach}()$ behaves identically to $\mathcal{B}_\sigma(I)$, the output simplifies to:

$$\mathrm{GBM}(I) \equiv I \quad \text{(Forward Computation)} \tag{14}$$

However, during backward propagation, the gradient flow becomes non-trivial. Let's compute the partial derivative of $\mathrm{GBM}(I)$ with respect to $I_{i,j}$, where $I_{i,j}$ denotes the pixel value at the (i, j) position of the image. Since

Figure 1. Examples from our introduced dataset, which contains 500 samples.

$\mathcal{B}_\sigma(I).\text{detach}()$ by construction has zero gradient during backward propagation:

$$\frac{\partial\, \mathcal{B}_\sigma(I).\text{detach}()}{\partial I_{i,j}} = \mathbf{0} \tag{15}$$

Thus:

$$\frac{\partial\, \text{GBM}(I)}{\partial I_{i,j}} = \frac{\partial\, \mathcal{B}_\sigma(I)}{\partial I_{i,j}} + \frac{\partial I}{\partial I_{i,j}} \tag{16}$$

The $\mathcal{B}_\sigma(\cdot)$ can be expressed as a convolution with kernel $\mathcal{K}_\sigma$:

$$\mathcal{B}_\sigma(I)_{p,q} = \sum_{i,j} I_{i,j} \cdot \mathcal{K}_\sigma(p-i, q-j) \tag{17}$$

Its gradient is:

$$\frac{\partial\, \mathcal{B}_\sigma(I)_{p,q}}{\partial I_{i,j}} = \mathcal{K}_\sigma(p-i, q-j) \tag{18}$$

Therefore, we obtain:

$$\begin{aligned} \frac{\partial\, \text{GBM}(I)_{p,q}}{\partial I_{i,j}} &= \frac{\partial\, \mathcal{B}_\sigma(I)_{p,q}}{\partial I_{i,j}} + \frac{\partial I_{p,q}}{\partial I_{i,j}} \\ &= \mathcal{K}_\sigma(p-i, q-j) + \delta_{(p,q),(i,j)} \end{aligned} \tag{19}$$

Here, $\delta_{(p,q),(i,j)}$ denotes the Kronecker delta, which equals 1 when the indices (p, q) and (i, j) are identical, and 0 otherwise. We consider the backward process of the loss function $\mathcal{L}$. Let $Y = \text{GBM}(I)$, then we have:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial I_{i,j}} &= \sum_{p,q} \frac{\partial \mathcal{L}}{\partial Y_{p,q}} \cdot \frac{\partial Y_{p,q}}{\partial I_{i,j}} \\ &= \sum_{p,q} \frac{\partial \mathcal{L}}{\partial Y_{p,q}} \cdot \left(\mathcal{K}_\sigma(p-i, q-j) + \delta_{(p,q),(i,j)}\right) \\ &= \underbrace{\sum_{p,q} \frac{\partial \mathcal{L}}{\partial Y_{p,q}} \cdot \mathcal{K}_\sigma(p-i, q-j)}_{\text{Broadcasting term}} + \underbrace{\frac{\partial \mathcal{L}}{\partial Y_{i,j}}}_{\text{Direct term}} \end{aligned} \tag{20}$$

This derivation shows that the GBM maintains the original forward output while enabling gradient propagation across spatial locations through the Gaussian kernel $\mathcal{K}_\sigma$, effectively solving the gradient locality problem in bilinear sampling operations by allowing the gradient computed at (p, q) to propagate to (i, j).

Specifically, unlike the standard STN [4], which is typically placed at the front end of CNNs where gradient locality is partially compensated by large receptive fields in subsequent layers, our GBM is applied at the final output of the U-Net, where the loss is computed directly. This distinction allows the GBM to more effectively correct large spatial biases by spreading gradients to strengthen positional supervision for the Position Path.

Furthermore, the continuous nature of the GBM (as shown in Eq. 3) ensures a smoother optimization compared to discrete modeling. While discrete representations are effective for encoding latent intermediate features, a continuous distribution is particularly suited for final image synthesis, where precise pixel-wise alignment and sub-pixel accuracy are required.

B. Additional Implementation Details

Table 1. Hyperparameters of two-stage training.

Hyperparameters	Training Stage I	Training Stage II
Learning Rate	5e-5	5e-5
Batch Size	64	64
Iteration	800k	50 k
Optimizer	AdamW [8]	AdamW [8]
Weight Decay	1e-3	1e-3
LR Schedule	Cosine [7]	Cosine [7]
Warmup Steps	10k	1k

B.1. Descriptions of Our Dataset

Fig. 1 displays 500 sampled glyphs rendered through our SPR scheme, obtained from 222 fonts by selecting two or three samples from each font. All samples originate from vector font files, ensuring structural fidelity and flexible resolution. This figure highlights the diversity of stroke morphology and stylistic variations across the collected fonts.

B.2. Network Architectures

Our model is based on RDDM [6], which contains two U-Nets that predict noise and residual (in our experiment, it is $I_G^{l,\,init}$) respectively. We modified both U-Nets by adding Cross Attention Layers to enable them to receive style condition $\mathcal{F}_S$ from the style encoder $\mathcal{E}_{style}$. The $\mathcal{E}_{style}$ is a network composed of Residual Blocks and Downsample Blocks, where features derived from several references are averaged to obtain $\mathcal{F}_S$. We extract features from each layer of the U-Net used for predicting residual to form the position path. Specifically, the features from each layer of this U-Net undergo dimension reduction via 1×1 convolution and MLP. These features are then concatenated and fed through another MLP to obtain the spatial correction offset $\varphi_\triangle$.

The SDE is designed as a simple yet effective regression network. It primarily involves a downsampler DS and an upsampler US. The DS is a parameter-free bilinear downsampling method that reduces the image resolution from $H^2 = 128^2$ to $L^2 = 64^2$. The US, which is mainly composed of a Bilinear Upsampling Process, Residual Blocks, and Cross Attention Layers, uses the Bilinear Upsampling Process to obtain a blurred image of resolution H^2 as the residual, while the Cross Attention Layers take style details from $\mathcal{F}_S$ to enhance fine image details.

Table 2. Additional qualitative comparison with SOTA methods on the UFUC dataset. The table contains enlarged visual results for detailed inspection and spans two pages, with the next page continuing the same table.

LF-Font [9]	MX-Font [10]	NTF [1]	MSD-Font [2]	**Ours**	Ground Truth
					

(Continued)

Table 3. Qualitative results of our method in cross-language generation. We use Simplified Chinese as the style reference for generating glyphs in other writing systems. Zoom-in for better inspection.

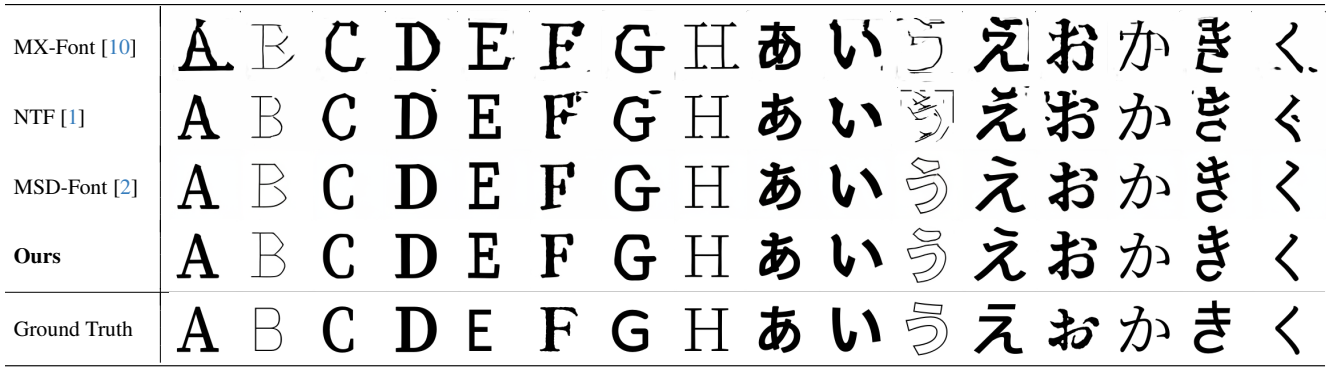


Table 4. Additional ablation study of our proposed module, evaluated on the UFSC datasets.

	Configurations	RMSE↓	PSNR↑	SSIM↑	LPIPS↓
4-shot for UFSC	Base	0.1300	19.12	0.8820	0.0573
	Base + SDE	0.1000	25.10	0.9034	0.0512
	+ SPD w/o GBM	0.1017	24.71	0.9001	0.0522
	+ SPD w/ GBM	**0.0947**	**25.51**	**0.9093**	**0.0498**

B.3. Training Details

Our training process consists of two stages. In the first stage, we train the font style transfer at the low resolution $L^2 = 64^2$. In the second stage, we freeze the style encoder $\mathcal{E}_{style}$ and separately train the SDE at the high resolution $H^2 = 128^2$. The hyperparameters are shown in Tab. 1.

We do not use pre-trained models because glyph images require single-channel grayscale inputs rather than the three-channel RGB format assumed by most existing models, and the distribution of glyphs differs fundamentally from that of natural images on which general-purpose models are trained. Therefore, all of our models are trained from scratch.

C. Further Analysis and Discussions

In this section, we provide detailed discussions to address potential concerns regarding our methodology and experimental design, aiming to offer a deeper understanding of the principles and motivations underlying this work.

C.1. Conceptual Clarification and Intuition

To provide a clearer understanding of the core concepts in this paper, we offer the following intuitive clarifications:

- *Glyph Spatial Information* refers to the information defined for glyphs in the font design coordinate system of printed fonts and can be understood as the predefined placement and scaling of a glyph within the font design canvas.
- *Distorted Rendering* corresponds to the practices exemplified by Tab. 1 of the main paper, such as bounding-box centering or non-uniform scaling.
- *Spatial Bias* refers to the outcome of *Distorted Rendering*, such as position drift.
- *Learned Mapping* refers to the input–output patterns learned by the model during training, which are disrupted by *Spatial Bias*.

In essence, these core concepts stem from our core perspective: we treat glyphs as spatially constrained objects rather than coordinate-free textures as in previous typical FFG methods.

C.2. Comparison with Recent SOTAs

At the time of our experiments, several recent works such as FontAnimate [3] and HFH-Font [5] had not released their official code. Furthermore, FontDiffuser [12] is not a few-shot method, and HFH-Font [5] reports results primarily under a large-reference setting (e.g., $N_{ref} = 775$), which deviates from the conventional few-shot setting. To ensure a fair and reproducible comparison, we therefore do not include these methods in our main evaluation and instead focus on established baselines with accessible implementations.

C.3. Effectiveness of our SPR Scheme

As discussed in Sec. 4.3 of the main paper, the SPR scheme improves performance by removing spatial bias at the data level. For example, under distorted rendering, the differences between predictions and GTs often arise from small positional shifts rather than missing strokes. Without well-defined spatial information, the model may be penalized for such semantically meaningless shifts, which hinders the learning of structural consistency.

C.4. Necessity of the Position Path

Typical FFG requires a source (source glyph image) to provide content. Under different style conditions, the same

source glyph corresponds to target glyphs at different positions. A style-reference-conditioned diffusion model may implicitly learn such positional mappings, but our ablation study in Tab. 5 of the main paper shows our Position Path yields further gains.

C.5. Selection of Resolution

Our adoption of the 128^2 resolution is aligned with our goal of obtaining usable TTF files, which requires precise semantic coordinate information rather than mere pixel density. While recent impressive works such as HFH-Font [5] explore higher resolutions to capture intricate raster details, our approach provides a complementary perspective. As validated in Sec. 4.5.1 of the main paper, when accurate spatial mappings are maintained via our SPR scheme, the resolution need not be excessively high, as the errors introduced during rendering and vectorization become marginal. This suggests that the current primary bottleneck in font generation lies in the performance of the generative model itself, rather than the image resolution.

C.6. Potrace and Vectorization

Recent trends in FFG aim at vectorizing from pixels rather than directly generating vector control points, as the latter is known to be more fragile, as discussed in Sec. 2.1 of the main paper. Therefore, our current method primarily focuses on the raster modality. Once spatial bias is removed by our SPR scheme, glyph raster-to-vector conversion becomes a robust post-processing step, for which the deterministic Potrace (2003) [11] is sufficient to demonstrate our performance. Consequently, advanced learned vectorization and explicit vector quality metrics are beyond the scope of this work, and we leave them for future work.

D. Additional Experimental Results

D.1. Additional Results

We further provide additional experimental results, including Tab. 2 and Tab. 4.

D.2. Exploring Other Writing Systems

As illustrated in Tab. 3, although our model is trained exclusively on Simplified Chinese, it demonstrates strong generalization to unseen writing systems. We hypothesize that this stems from the explicit decoupling of spatial transform and style transfer modeled entirely in pixel space, allowing the model to capture cross-lingual glyph distributions rather than relying on language-specific local structural priors. Qualitative tests on English and Japanese glyphs show well-preserved structures and consistent styles, confirming the model's potential for multi-script font generation.

Compared to other SOTA methods, LF-Font [9] requires component labels, yet the training data only contains Chinese, making inference unable. MX-Font [10] only requires component labels during training, but its dependency limits performance. NTF [1] is constrained by GAN's generative capability, leading to stroke errors. MSD-Font [2] and Ours are trained using spatially unbiased complex Chinese glyphs obtained via SPR scheme, yielding favorable results on simpler glyphs.

D.3. Vectorization Quality and Analysis

Fig. 2 illustrates a representative case of the intermediate results from the experiment described in Fig. 4 of the main paper. We observe that the 128^2 resolution is sufficient to validate the effectiveness of our proposed approach.

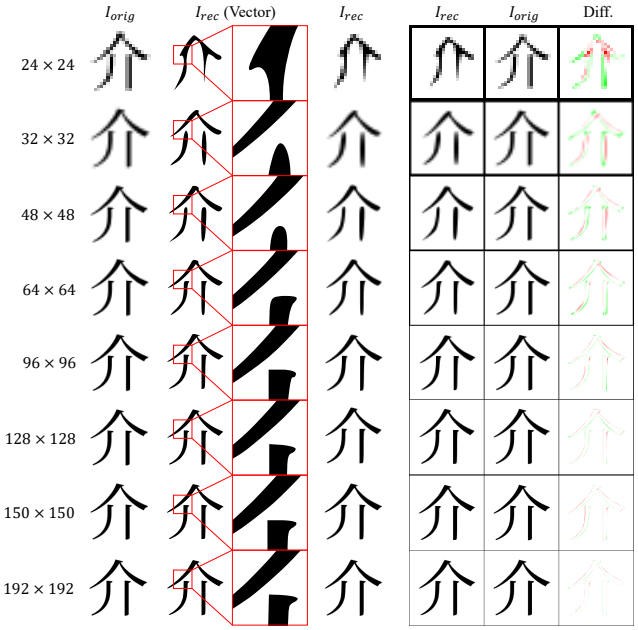


Figure 2. Vectorization results across different resolutions. Zoom-in for better inspection.

D.4. Comparison of Rendering and Analysis

The discrepancies in Fig. 3 arise from bounding-box definitions in certain fonts that deviate from actual glyph positions, where centering based on such data introduces positional offsets that shift glyphs away from the image center. Additionally, non-uniform scaling causes inconsistent glyph sizes even within the same font style. In raster-based models, pixel-wise differences sometimes stem from translations. Without the explicit coordinate definition of SPR, these offsets lack a valid semantic basis for optimization, ultimately hindering model refinement.

D.5. Analysis of the Normalized Metric

D.5.1. Sensitivity Analysis and Comparison

As shown in Fig. 4, we select a random glyph with varying stroke thicknesses from the same font family. When the

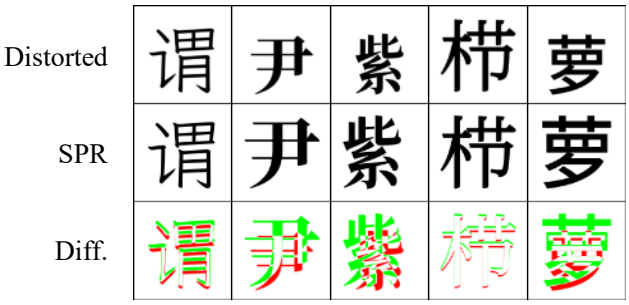


Figure 3. Visual comparison between distorted rendering and our SPR scheme.

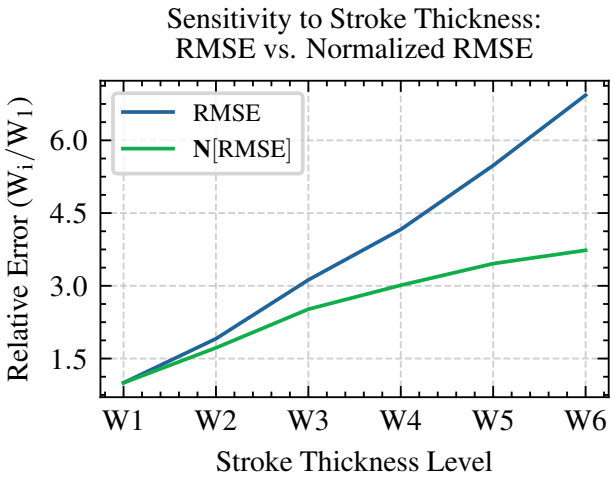
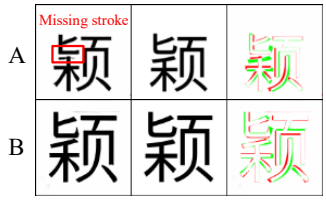


Figure 4. Sensitivity analysis of RMSE versus Normalized RMSE (**N**[RMSE]) under varying stroke thicknesses of the GT glyph.

thickness of the GT glyph changes, it can be observed that while the standard RMSE increases sharply, our normalized RMSE exhibits a much flatter and more stable trend.

D.5.2. Correlation with Human Perception

As shown in Fig. 5, Glyph A has obvious stroke missing, yet its unnormalized metrics are "diluted" by larger padding, leading to results that contradict human judgment. In contrast, our normalized metric yields consistent results by focusing on the valid glyph region.



A: Distorted Rendering
B: SPR Scheme

	A	B
RMSE	0.1660	0.1705
N[RMSE]	1.2090	0.8489

Figure 5. A representative case illustrating the contradiction between unnormalized metrics and human visual perception.

Table 5. Comparison with SOTA methods on the UFUC dataset showing cases with imperfect strokes or stylistic details.

LF-Font [9]	MX-Font [10]	NTF [1]	MSD-Font [2]	**Ours**	Ground Truth
					

D.6. Limitation

As shown in Tab. 5, some glyphs still present imperfect strokes or style details. This is mainly due to the high cost of Chinese font production, which makes certain styles such as calligraphy and handwriting relatively scarce in available fonts. As a result, our dataset remains challenging, especially for rare stylistic categories.

References

[1] Bin Fu, Junjun He, Jianjun Wang, and Yu Qiao. Neural transformation fields for arbitrary-styled font generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22438–22447, 2023. 4, 5, 6, 7, 8

[2] Bin Fu, Fanghua Yu, Anran Liu, Zixuan Wang, Jie Wen, Junjun He, and Yu Qiao. Generate like experts: Multi-stage font generation by incorporating font transfer process into diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6892–6901, 2024. 4, 5, 6, 7, 8

[3] Bin Fu, Zixuan Wang, Kainan Yan, Shitian Zhao, Qi Qin, Jie Wen, Junjun He, and Peng Gao. Fontanimate: High quality few-shot font generation via animating font transfer process. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16015–16025, 2025. 6

[4] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015. 3

[5] Hua Li and Zhouhui Lian. Hfh-font: Few-shot chinese font synthesis with higher quality, faster speed, and higher resolution. *ACM Transactions on Graphics (TOG)*, 43(6):1–16, 2024. 6, 7

[6] Jiawei Liu, Qiang Wang, Huijie Fan, Yinong Wang, Yandong Tang, and Liangqiong Qu. Residual denoising diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2773–2783, 2024. 3

[7] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 3

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 3

[9] 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*, pages 2393–2402, 2021. 4, 5, 7, 8

[10] 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, 5, 6, 7, 8

[11] Peter Selinger. Potrace: a polygon-based tracing algorithm, 2003. 7

[12] Zhenhua Yang, Dezhi Peng, Yuxin Kong, Yuyi Zhang, Cong Yao, and Lianwen Jin. Fontdiffuser: One-shot font generation via denoising diffusion with multi-scale content aggregation and style contrastive learning. In *Proceedings of the AAAI conference on artificial intelligence*, pages 6603–6611, 2024. 6