FontAnimate: High Quality Few-shot Font Generation via Animating Font Transfer Process

Supplementary Material

1. Implement Details

In this section, we provide more implementation details about our proposed FontAnimate.

To prepare training and reference font videos, we use the FILM model to perform frame interpolation from the source font image (the first frame) to the target font image (the last frame). We modify the FILM model into a multiprocessing version and utilize two CPUs (Intel 6242R @ 3.10GHz) to generate the above font videos, which takes five days.

We adopt our FontAnimate on the public-available Latent Diffusion Model (LDM) [4] platform to generate 128×128 font images/videos with eight reference samples (8-shot), and utilize its VAE [5] as the image/video-latent projection network in our model.

1.1. Image-to-Image Font Generation Framework:

As discussed in the previous section, in this paper, we first construct our image-to-image font generation framework.

Following the common practice [2, 3] in the FFG task, we construct the style encoder E_s and content encoder E_c to extract the style and content conditions from the reference samples I_s and source images I_c, respectively. The architecture of style and content encoders is modified from MSD-Font by changing the output channels of the last block. We implement and modify a transformer-based neural network, namely PixArt-α [1], as our denoising network $\hat{z}^{img}_{\theta^{img}}(z^{img}_t, t, e^{img})$. PixArt-$\alpha$ is a transformer-based diffusion model for conditional generation, and we use this model with the following modifications: (1). The depth is 12; (2). The hidden size is 384; (3). The patch size is 1.

In the training stage, we obtain the ground truth latent features $z^{\text{img}}_0 \in R^{b\times 4\times 16\times 16}$ by using the VAE encoder $\mathcal{E}$ to project the target images $I_g \in R^{b\times 3\times 128\times 128}$ into the latent space, where b represents the batch axis. In the inference stage, the denoising network $\hat{z}^{img}_{\theta^{img}}(z^{img}_t, t, e^{img})$ will generate $z^{\text{img}}_0 \in R^{b\times 4\times 16\times 16}$ by gradually denoising z^{img}_t, and the generated font images x^{img}_0 can be obtained by using the VAE decoder $\mathcal{D}$ by project latent features z^{img}_0 back to image space. We utilize content encoder E_c to extract the content features $e^{\text{img}}_c \in R^{b\times 384\times 16\times 16}$ from the source images. We utilize style encoder E_s to extract the averaged style features $e^{\text{img}}_s \in R^{b\times 384\times 16\times 16}$ from the provided reference images (3 images for training and 8 images for inference). Finally, the content and style features are flattened and concatenated to build the condition $e^{\text{img}} \in R^{b\times 384\times 512}$.

1.2. Image-to-Video Font Generation Framework:

We further construct our font video denoising network $\hat{z}^{\text{vid}}_{\theta^{\text{vid}}}(z^{\text{vid}}_t, t, fi, e^{\text{vid}})$ by adding the proposed temporal part and frame-index information. For the noisy latent video representation $z^{\text{vid}}_t \in R^{b\times c\times f\times h\times w}$, where b and f represent the batch axis and the temporal axis. When the internal feature maps go through the image part, the temporal axis f is ignored by being reshaped into the batch axis b, allowing the network to process each frame independently. We then reshape the feature map back to the 5D tensor after the image part. On the other hand, our newly inserted temporal part ignores the spatial axis by reshaping h, w into batch axis b and then reshaping back after the calculation.

In the training stage, we use 8-frame font videos to optimize our model. We first sample 8 frames from each training video, and extract the same frames from the reference videos V_s. We obtain the ground truth latent feature $z^{\text{vid}}_0 \in R^{b\times 4\times 8\times 16\times 16}$ by using the VAE encoder $\mathcal{E}$ to project the target font videos $V_g \in R^{b\times 3\times 8\times 128\times 128}$ into the latent space in the frame-by-frame manner. In the inference stage, we generate font videos with 33 frames. The denoising network $\hat{z}^{vid}_{\theta^{vid}}(z^{vid}_t, t, fi, e^{vid})$ will generate $z^{\text{vid}}_0 \in R^{b\times 4\times 33\times 16\times 16}$ by gradually denoising z^{vid}_t, and the generated font video x^{vid}_0 can be obtained by using the VAE decoder $\mathcal{D}$ by project latent feature z^{vid}_0 back to RGB space in the frame-by-frame manner.

To generate font videos, we still use the source images to construct content conditions by utilizing content encoder E_c to extract the content features. Then the content features are broadcast to the shape $e^{\text{vid}}_c \in R^{b\times 384\times f\times 16\times 16}$, where $f = 8$ for training and $f = 33$ for inference. However, we use the reference font videos to provide style conditions. We employ style encoder E_s to extract the averaged style features $e^{\text{vid}}_s \in R^{b\times 384\times f\times 16\times 16}$ from the provided reference videos. Finally, the content and style features are flattened and concatenated to build the condition $e^{\text{vid}} \in R^{b\times 384\times f\times 512}$.

1.3. Optimization Details

We utilize a two-stage training strategy to optimize our FontAnimate. In the first stage, we train our style encoder E_s, content encoder E_c, and denoising network $\hat{z}^{img}_{\theta^{img}}$ with AdamW optimizer while keeping VAE encoder $\mathcal{E}$ and decoder $\mathcal{D}$ fixed. We use a single RTX 3090 GPU to optimize our model with the learning rate 5×10^{-5} and batch size 64 for this stage. In the second stage, construct the font video denoising network $\hat{z}^{vid}_{\theta^{vid}}$, and optimize our framework (in-

Figure 1. Qualitative comparisons of our FontAnimate with other six state-of-the-art methods in the Unseen Fonts and Unseen Contents (UFUC) setting.

Table 1. Ablation study for the training strategy of our FontAnimate on UFUC dataset in the second training stage. The bold number indicates the best.

Imag.	Temp. + FI	RMSE↓	PSNR↑	SSIM↑	LPIPS↓
-	✓	0.2516	12.25	0.7168	0.1372
✓	✓	**0.2483**	**12.43**	**0.7209**	**0.1340**

Table 2. Ablation study on the number of interpolation frames.

N_f	RMSE↓	PSNR↑	SSIM↑	LPIPS↓
I2I	0.2575	12.08	0.7105	0.1448
2	0.2607	11.97	0.7058	0.1455
2^2+1	0.2592	12.03	0.7065	0.1464
2^3+1	0.2560	12.09	0.7198	0.1442
2^4+1	0.2538	12.24	0.7144	0.1393
2^5+1	0.2483	12.43	0.7209	0.1340

cluding E_s, E_c, $\hat{z}^{vid}_{\theta^{vid}}$) with font videos. We use 4 RTX 3090 GPUs to optimize our model with the learning rate 2.5×10^{-5} and batch size 32 for this stage. In the training stage, the reference samples are randomly selected from the training set at each iteration step.

2. Additional Ablation Study

2.1. Ablation Study on Training Strategy

Our FontAnimate is built on the I2I-based font generation network by incorporating the temporal (Temp.) part and the frame-index (FI) information. Since the image (Imag.) part has already been optimized in the first stage, we perform an ablation study to evaluate whether further optimization is necessary in the second stage. As shown in Tab. 1, compared to freezing the image part in the second stage, continuing to train this part leads to significant improvements, demonstrating that optimizing the image part is still necessary for effectively learning font transformation patterns.

2.2. Ablation Study on the Interpolation of Different Number Frames

We interpolate the font images with different N_f for training, and the results (in UFUC testing set) are present in Tab. 2. The performance of $N_f = 2, 5$ is lower than I2I model, since small N_f cannot well describe the font transfer process. When N_f become large, the performance increases, verifying that the performance indeed comes from the interpolation of frames. The inference time increases when the N_f increase. We think that performance will continue to improve when $N_f > 33$.

2.3. Ablation Study of the Number of Frames in Inference.

In this paper, we use the font video of 33 frames to train our model. We test different number of frames N_{inf} in inference. The results (in Tab. 3) show that performance will decrease when N_{inf} not aligned with training samples.

Table 3. Ablation study on the number of frames in inference.

N_{inf}	RMSE↓	PSNR↑	SSIM↑	LPIPS↓
2	0.3006	10.69	0.6470	0.2061
5	0.2780	11.41	0.6815	0.1686
9	0.2574	12.10	0.7102	0.1422
17	0.2519	12.30	0.7175	0.1356
33	0.2483	12.43	0.7209	0.1340

2.4. Ablation Study on the effects of VAE

In LDM framework, VAE indeed damages the image quality. We investigate this problem by using VAE encoder to project image into latent space and then reconstructing images via VAE decoder in UFUC testing set. The RMSE, PSNR, SSIM, LPIPS are: 0.017, 36.60, 0.9962, 0.002, verifying the loss of VAE can be ignored in our current model.

2.5. Visualization of Training Samples

We provide the training samples of our model in the Fig. 2.

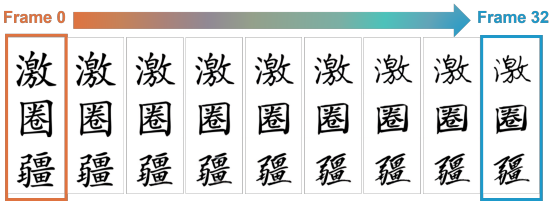

Figure 2. Visualization of training samples.

2.6. Discussion on the Inference Time and Computational Efficiency

Since we construct our font generation platform based on diffusion-based video generation models and use the noise inversion (NI) mechanism to perform condition alignment, the inference time and computational resources are increased compared to previous GAN-based FFG methods. However, we believe it is worth spending more time to generate font images, due to the following reasons: (1). Although I2V model and NI mechanism bring more inference time, our model achieves all best performance in UFUC setting, and have superior visual quality than other model. Due to this improvement. (2). NI is a test-time noise inversion mechanism, which can be viewed as a test-time scaling (TTS) strategy. We provide a pioneering attempt to apply TTS in FFG task. (3). In the generated videos, some intermediate frames can be regarded as new fonts, providing more creative potential for users. (4). Finally, considering performance improvement, new perspective, and new technique, we think our work will offer a new direction for FFG.

3. Additional Qualitative Results

In this section, we present additional qualitative results of our proposed FontAnimate framework. Specifically, Fig. 1 illustrates a qualitative comparison between FontAnimate and six state-of-the-art methods in UFUC setting. Our approach effectively preserves overall structural completeness when generating font images, even for complex characters. Moreover, we present further visualization results of the generated font videos (Fig.3) and font images (Fig.4) produced by our proposed FontAnimate framework.

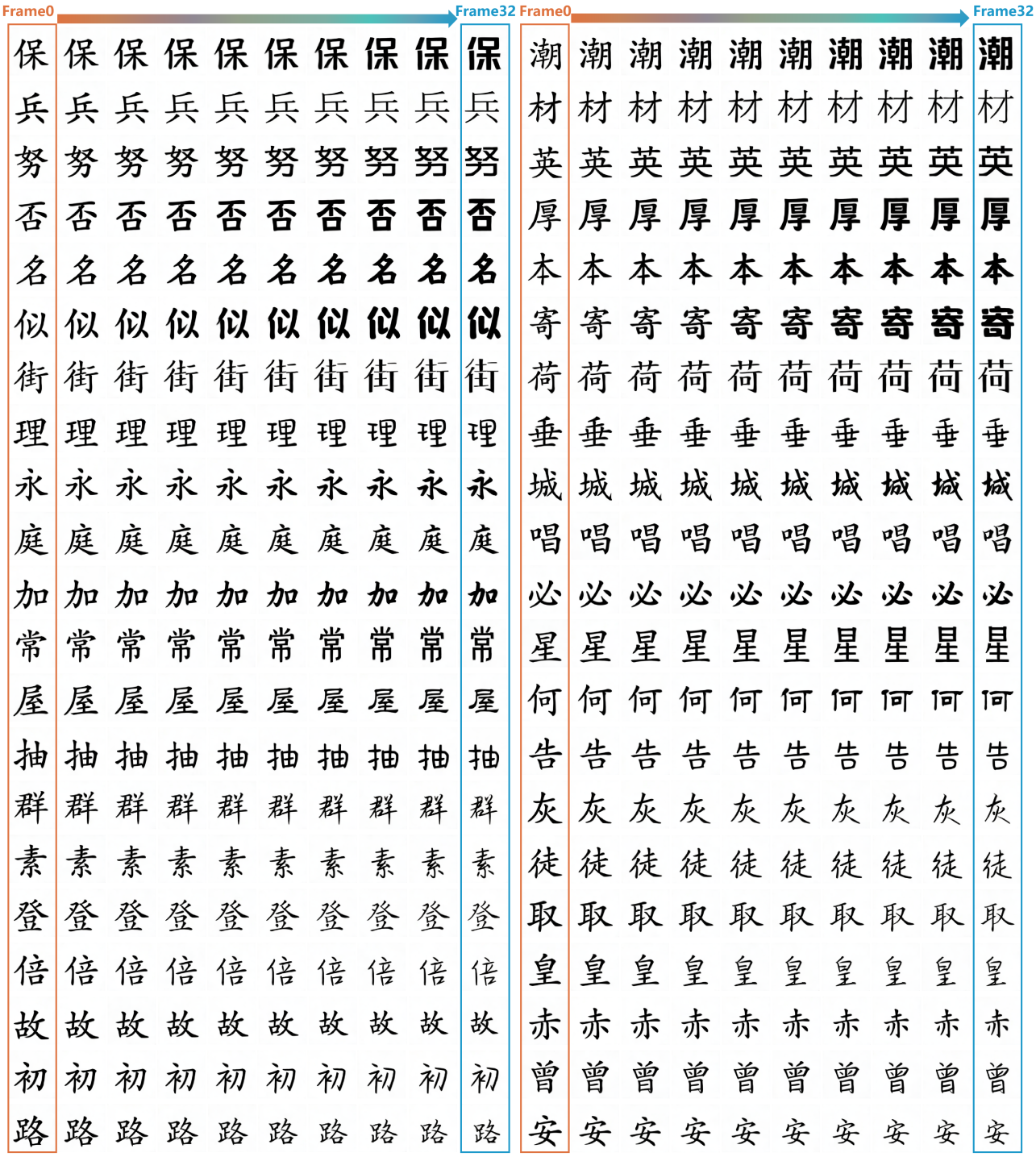

Figure 3. Generated font videos from our FontAnimate in the Unseen Fonts and Unseen Contents (UFUC) setting.

Figure 4. Additional qualitative results of our FontAnimate on UFUC dataset.

References

[1] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023. 1

[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. 1

[3] 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. 1

[4] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1

[5] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 1