VecFusion: Vector Font Generation with Diffusion -Supplementary Material-

A: Implementation Details

Here we provide additional implementation details of our network architecture. Our model is implemented in PyTorch.

Raster diffusion denoiser. Our raster diffusion denoiser follows the UNet architecture in [4, 6]. The UNet model uses a stack of residual layers and downsampling convolutions, followed by a stack of residual layers with upsampling convolutions, with skip connections connecting the layers with the same spatial size. We provide an overview of the hyperparameters in Table 1. To condition the model on the character identifier, we use a look-up table to project it to an 896-dimensional embedding and then add it together with the time step embedding to modulate the feature maps of each residual block.

Few-shot font style encoder. In the application of few-shot font style transfer, we used a ConvNet to encode reference raster glyphs into a font style feature map. We used the encoder part of the UNet architecture in [4, 6]. The ConvNet encoder encodes the 64×64 input image into an $8 \times 8 \times 512$ high-dimensional feature map via 3 downsampling layers.

Vector diffusion denoiser. Our vector diffusion denoiser is an encoder-only transformer following BERT [3]. We set the number of transformer layers and the number of attention heads to 8 and 12 respectively. To condition the vector diffusion denoiser on the raster guidance x_0, we first encode the $64 \times 64 \times 4$ raster image to a $16 \times 16 \times 768$ high-dimensional feature map with a ConvNet encoder. The ConvNet encoder has two downsampling layers with self-attention layers at resolution 32×32 and 16×16. The ConvNet encoder is trained with the transformer jointly. After obtaining the $16 \times 16 \times 768$ high-dimensional feature map, we flatten it to a shape of 256×768, then we add it to each transformer layer via cross-attention following [6].

Computation cost. Raster-DM and Vector-DM are trained separately. Each of them is trained on 8 A100 GPUs for 5 days. Finally, at inference time, generating a glyph takes around 10 seconds on a A100.

Input shape	$64 \times 64 \times 4$
Diffusion steps	1000
Noise Schedule	cosine
Channels	224
Depth	2
Channel Multiplier	1,2,3,4
Attention resolutions	32,16,8
Head Channels	32
Batch Size	448
Learning Rate	3.24e-5

Table 1. Hyperparameters for raster diffusion denoiser

B: Additional Results

Comparison with a vectorizer approach. As an alternative comparison, we tried the following approach: instead of using the vector diffusion model, we use PolyVec [1] or LIVE [5] on the rasterized font image produced by our raster diffusion stage. We also tried upsampling the 64×64 output raster image to 256×256 using ESRGAN [7] before passing it to the vectorizer. We show qualitative comparison in Figure 1. In both cases, PolyVec and LIVE often failed to produce coherent curve topology, structure, and plausible control point distributions.

Additional comparisons with DeepVecFont-v2 [8]. Please see Figure 2 for more comparisons with DeepVecFont-v2 [8] on the task of few-shot font style transfer.

Additional comparisons with ChiroDiff [2]. Please see Figure 3 for more comparisons with ChiroDiff [2] on the task of missing unicode generation.

References

[1] Mikhail Bessmeltsev and Justin Solomon. Vectorization of line drawings via polyvector fields. *ACM Trans. Graph.*, 38

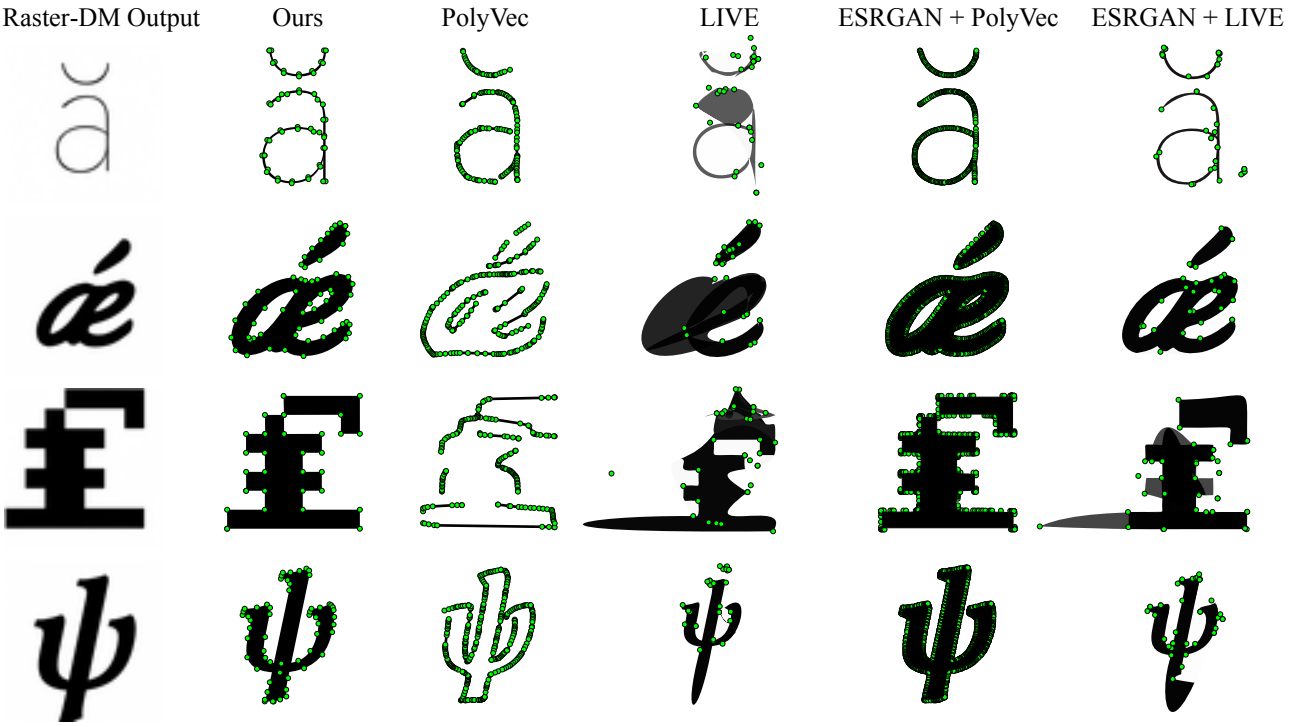

Figure 1. We compare our results (Ours) with PolyVec [1] and LIVE [5] applied to the raster image produced by our raster diffusion stage (left-most column). We also compare with PolyVec and LIVE applied to a higher-resolution version of the raster image upsampled via ESRGAN [7]. For each glyph, we show the predicted control points as well. Using our vector diffusion stage instead of an off-the-shelf vectorizer produces higher-quality glyphs and much more plausible control point distributions. Compared to our vector diffusion model, ESRGAN + PolyVec requires about ten times more control points for effective glyph reconstruction but sacrifices user editability and SVG compactness. We recommend the viewer to zoom in for better clarity.

(1):9:1–9:12, 2019. 1, 2

[2] Ayan Das, Yongxin Yang, Timothy Hospedales, Tao Xiang, and Yi-Zhe Song. Chirodiff: Modelling chirographic data with diffusion models. In *International Conference on Learning Representations*, 2023. 1, 3

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[4] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 1

[5] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2022. 1, 2

[6] 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

[7] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1905–1914, 2021. 1, 2

[8] Yuqing Wang, Yizhi Wang, Longhui Yu, Yuesheng Zhu, and Zhouhui Lian. Deepvecfont-v2: Exploiting transformers to synthesize vector fonts with higher quality. *arXiv preprint arXiv:2303.14585*, 2023. 1, 3

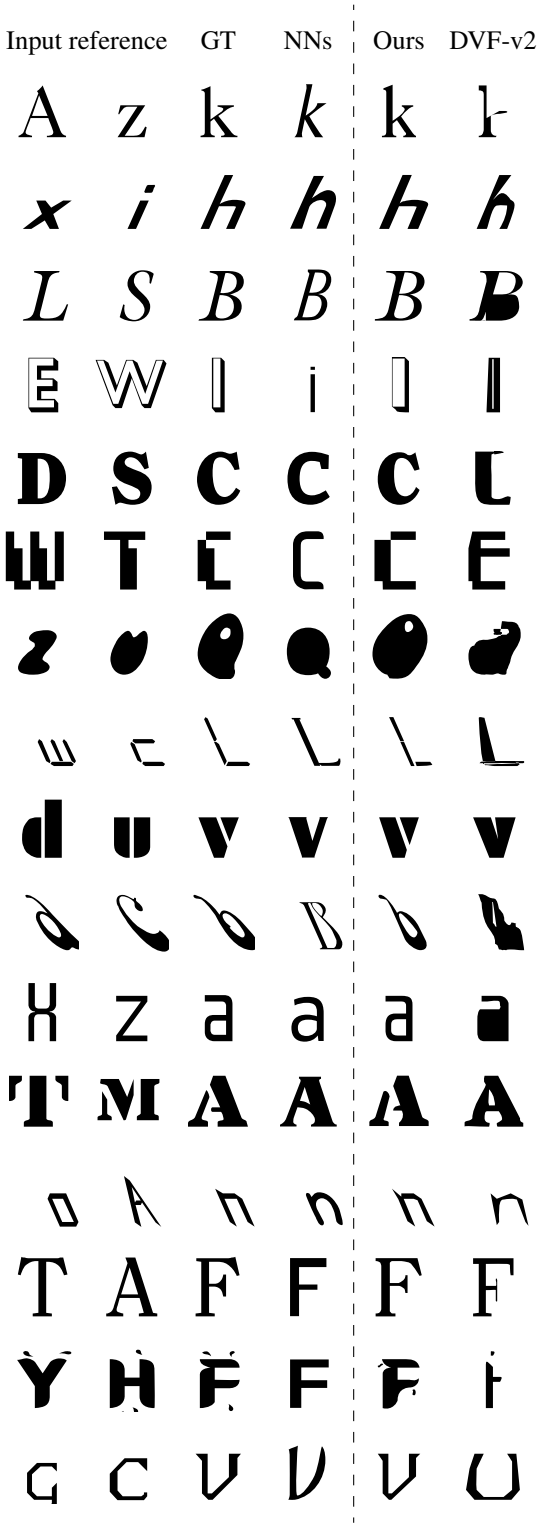

Figure 2. Few-shot style transfer results. From left to right, we show the reference glyphs (2 out of 4) belonging to a novel font style, the artist-made ("ground-truth/ GT") glyphs, the nearest-neighbours ("NNs") to GT in the training data, our generated ones, and DeepVecFont-v2 (DVF-v2) [8]

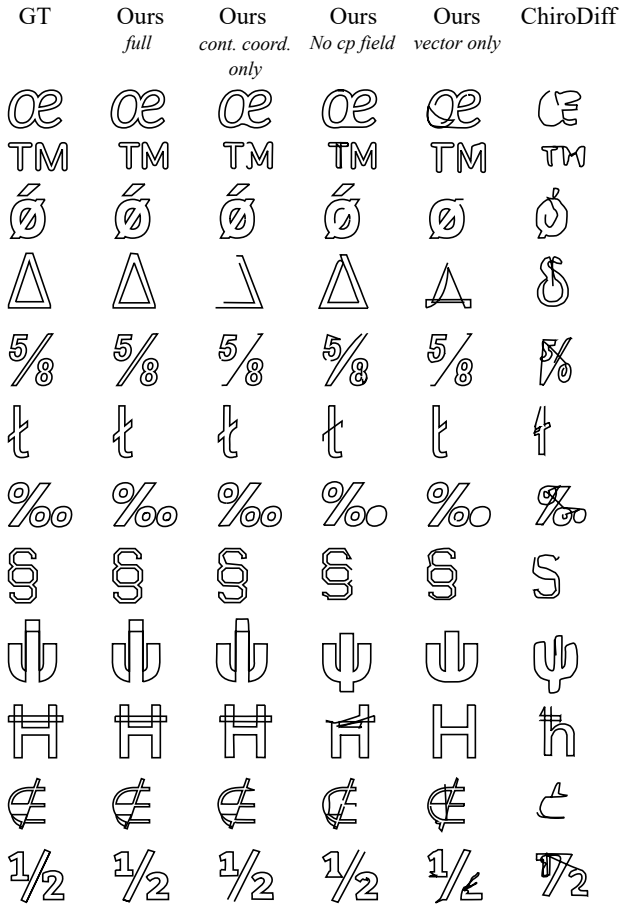

Figure 3. Glyph generation results for test cases from the Google font dataset. We compare our method to ChiroDiff [2] and degraded variants of our method. Our full method is able to generate glyphs that are much closer to artist-made ("ground-truth"/"GT") ones compared to alternatives.