LITEGLYPH: A COMPACT AUTOREGRESSIVE FRAMEWORK FOR FEW-SHOT CHINESE VECTOR FONT GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Few-shot Chinese vector font generation requires synthesizing editable outlines for unseen characters while preserving character structure and typeface style, given a standard-form structure reference and only a few cross-character style references. Existing raster methods require post-hoc vectorization, whereas direct vector approaches often rely on specialized multi-stage pipelines or substantially larger generative models. We present **LITEGLYPH**, a compact unified autoregressive framework that directly generates complete SVG command sequences. LITEGLYPH represents target-character structure as a visual prefix and encodes reference glyphs into dense style tokens that provide query-adaptive modulation throughout the decoder, allowing different SVG commands and coordinates to retrieve relevant local style evidence. We additionally introduce Raster-to-SVG initialization to establish a transferable vector prior and loop-aware rollback to improve decoding reliability. We train and evaluate on a new large-scale Chinese raster–vector corpus containing approximately 16M glyph pairs, over 2,600 curated fonts, and 6,763 characters. LITEGLYPH achieves the lowest FID among the evaluated external baselines across validation, unseen-font seen-character, and unseen-font unseen-character settings, while directly producing editable SVGs with a 350M-parameter decoder. Ablations further show that dense style modulation improves unseen-font generalization over global conditioning on most metrics, and that Raster-to-SVG initialization is particularly effective when downstream font diversity is limited.

1 INTRODUCTION

Vector fonts are the standard representation of digital typefaces in design and production. They describe glyphs as editable, resolution-independent outlines that can be directly revised, scaled, and compiled into font files. Given a standard-font image of a target character and several different characters from an unseen typeface, a model must preserve the target structure, infer style attributes shared across the references, and produce a syntactically valid and geometrically faithful outline. Most few-shot font generation methods instead operate in the raster domain (Park et al., 2021; Tang et al., 2022; Liu et al., 2022). Although these methods can synthesize visually plausible glyph images, recovering production-ready outlines requires an additional vectorization stage that may introduce redundant paths, inaccurate curves, and structural artifacts.

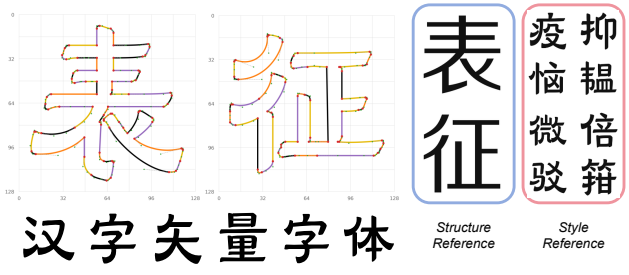


Figure 1: Few-shot Chinese vector font generation. A Heiti glyph provides target structure, while other glyphs from the target font provide style for an editable SVG output.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Direct vector generation avoids this post-processing step but poses a substantially harder sequence-modeling problem. An SVG sequence jointly specifies contour topology, drawing commands, point coordinates, and Bézier geometry. These elements are strongly dependent: an incorrect command or contour decision changes the interpretation of subsequent coordinates, while small geometric errors can visibly distort stroke boundaries. Earlier vector-font systems addressed these challenges using specialized representations, hierarchical generation, or contour refinement (Wang & Lian, 2021; Wang et al., 2023; Liu et al., 2023). More recent language-model approaches demonstrate strong direct vector synthesis, but employ billion-scale generators or multiple separately trained models (Huang et al., 2026; Yuan et al., 2026).

Few-shot Chinese vector font generation introduces an additional conditional-generation challenge. Chinese vector glyphs pose distinctive modeling challenges: they often contain many spatially dependent strokes, complex contour topology, and long SVG command sequences, such that small errors in contour organization or control-point placement can noticeably affect legibility. Moreover, the style references depict characters different from the target, requiring the model to distinguish reusable font-level attributes from character-specific geometry. Existing work has not established a unified framework that jointly handles complete SVG generation, factorized structure and cross-character style conditioning, and generalization across large and diverse Chinese font collections.

To address these challenges, we present **LITEGLYPH**, a unified autoregressive framework whose key design is a *factorized conditioning interface* for autoregressive explicit geometry: target-character structure is represented as a causal visual prefix that specifies *what* to generate, while cross-character font style is kept as an external dense memory that determines *how* the glyph is rendered and is queried throughout decoding. To preserve spatially localized style evidence, the reference glyphs are encoded into a compact set of dense style tokens rather than compressed into a single global representation. At every Transformer layer, each SVG token retrieves the style evidence relevant to its current command and geometry, enabling full-depth, query-adaptive modulation of the generation process. This design allows a compact model to directly generate complete and editable Chinese SVG glyphs without raster synthesis or post-hoc vectorization.

We train and evaluate LITEGLYPH on a large-scale Chinese vector-font corpus with fixed seen and unseen partitions over both fonts and characters, and additionally study Raster-to-SVG initialization as a data-efficient training strategy, whose benefit is most pronounced when downstream adaptation covers only a few font families and narrows as growing font diversity supplies increasingly sufficient vector supervision. Experiments address three questions. First, we evaluate vector-domain methods on a method-compatible subset with fonts held out from downstream adaptation (Table 1); this comparison is distinct from end-to-end unseen-font evaluation because its held-out fonts appear in Raster-to-SVG initialization. On the full-benchmark validation, UFSC, and UFUC splits, LITEGLYPH achieves the lowest FID among the evaluated external raster baselines (Table 2). Second, controlled ablations isolate the contribution of full-depth dense modulation over global conditioning: on UFSC, the dense model reaches an FID of 2.759 and an SSIM of 0.745, compared with 2.802 and 0.742 (Table 2). Third, UFSC and UFUC evaluations probe generalization to unseen fonts and characters, alongside decoding analyses of SVG validity, termination, repetitive loops, and geometric fidelity.

Our contributions are threefold:

- We construct a large-scale Chinese raster–vector corpus with approximately 16 million aligned glyph pairs from over 2,600 curated fonts and 6,763 characters, supported by a reproducible pipeline for outline canonicalization and quality validation.
- We introduce LITEGLYPH, a unified autoregressive framework that separates target-character structure from cross-character font style and directly generates complete, editable Chinese SVG glyphs.
- We formulate the conditioning of autoregressive explicit geometry as a factorized interface—a causal structure prefix plus an external dense style memory queried by every decoder layer—and show that this full-depth, query-adaptive access to style evidence outperforms global conditioning on most unseen-font metrics.

2 Related Work

2.1 Font Generation

Few-shot font generation synthesizes the unobserved characters of a typeface from a small set of reference glyphs. Most methods operate in raster space and transfer reference style to a standard-font content glyph through localized experts (Park et al., 2021), fine-grained content–style correspondence (Tang et al., 2022), or cross-modality pretraining over glyph images and stroke labels (Liu et al., 2022). More recent raster generators employ global and local style quantization, denoising diffusion, ideographic descriptions, or global-aware autoregression (Pan et al., 2023; Yang et al., 2024; Chen et al., 2024; Cai et al., 2026). Although these methods capture detailed style cues, their raster outputs require a separate vectorization stage; direct vector methods instead model glyph outlines. Aoki & Aizawa (2022) generate Chinese SVG glyphs from a content and style reference with a Transformer. DeepVecFont and DeepVecFont-v2 combine raster and sequence information (Wang & Lian, 2021; Wang et al., 2023), DualVector learns paired positive and negative paths from raster supervision (Liu et al., 2023), and VecFusion adopts cascaded raster and vector diffusion (Thamizharasan et al., 2024). More recently, VecGlypher formulates vector glyph synthesis as multimodal language modeling (Huang et al., 2026), while VecFontLLM decomposes complex Chinese glyph generation into anchor-scaffold prediction, refinement, and Bézier completion (Yuan et al., 2026). These approaches substantially advance font generation, yet direct vector systems commonly depend on specialized decompositions, refinement stages, or substantially larger language models; we instead ask whether structure preservation, cross-character style transfer, and complete SVG synthesis can coexist in a unified task-specific autoregressive framework.

2.2 General Vector Generation Methods

General vector graphics generation covers image vectorization and text-conditioned synthesis of icons, illustrations, and diagrams. DeepSVG introduces a hierarchical representation that separates high-level shapes from low-level drawing commands (Carlier et al., 2020), while SVGDreamer uses diffusion priors and score distillation to optimize editable vector primitives from text (Xing et al., 2024). Hybrid systems such as Chat2SVG combine language-model-generated templates with diffusion-guided path refinement (Wu et al., 2025). More recent multimodal language models generate SVG code directly: StarVector supports image vectorization, text-to-SVG generation, and diagram generation with diverse SVG primitives (Rodriguez et al., 2025), and LLM4SVG improves SVG understanding and generation through explicit semantic tokens and instruction tuning (Xing et al., 2025). Unlike these general-purpose systems, vector fonts impose a narrower but stricter requirement: the output must preserve the identity and closed-contour geometry of a target character while maintaining a coherent typeface style inferred from different reference characters.

2.3 Autoregressive Generation of Explicit Geometry

MeshGPT, MeshAnything, and MeshAnything V2 generate explicit three-dimensional geometry as discrete token sequences with autoregressive Transformers (Siddiqui et al., 2024; Chen et al., 2025a;b), demonstrating that such models can learn the syntax and geometry of production representations rather than their rendered appearance alone. We adopt this principle for SVG glyphs, whose sequences encode path commands, contour closure, coordinate order, and Bézier geometry, under two heterogeneous conditions: a same-character structure reference and cross-character style exemplars. The resulting challenge is not merely raster-to-vector conversion, but controlled autoregressive generation of explicit geometry.

3 Method

As illustrated in Figure 2, LiteGlyph is a unified autoregressive framework for direct Chinese vector font generation. The target character structure is represented as a visual prefix, while the target typeface is represented by dense style tokens extracted from a reference set. These two conditions enter the generator through separate pathways: structure participates in causal self-attention, whereas style produces token-dependent normalization parameters throughout the decoder. We first establish vector generation capability through Raster-to-SVG initialization and then adapt the model to structure

and style conditioned generation. Loop-aware rollback is used during inference to improve decoding reliability; style classifier-free guidance (scale 1.5) is applied during sampling.

3.1 A Large-Scale Chinese Raster–Vector Font Corpus

We construct a paired corpus of approximately 16 million raster–vector glyphs covering 6,763 GB2312 characters in more than 2,600 typefaces, collected from multiple repositories and filtered from 3,082 hash-deduplicated TrueType and OpenType candidates; this scale and stylistic coverage substantially exceed the datasets used by earlier specialized Chinese vector-font systems (Wang & Lian, 2021; Wang et al., 2023; Liu et al., 2023). Curation enforces a requirement specific to structure-conditioned generation: the standard-form rendering of a character, which serves as the model's structure reference, must faithfully specify the structure of its target outline. Screening therefore removes corrupted, unsuitable, and near-duplicate typefaces and, crucially, the vast majority of cursive-script and traditional-form fonts, whose glyphs for the same codepoint deviate drastically, or entirely, from the canonical letterform. Keeping such fonts would pair a shared structure reference with unrelated target outlines and inject systematic supervision noise; removing them trades exotic styles for consistency while retaining broad stylistic diversity among structurally canonical typefaces. The remaining outlines are canonicalized and validated through automated consistency checks and manual audits; the full construction pipeline is described in Appendix A.

3.2 Structure-Conditioned Autoregressive SVG Generator

3.2.1 SVG Sequence Representation

We normalize each glyph to a fixed view box and represent its outline using absolute `M`, `L`, `C`, and `Z` commands defined by the SVG standard (World Wide Web Consortium, 2018). Coordinates are uniformly quantized and encoded as discrete tokens. Command symbols and coordinate values share the output vocabulary, while `BOS`, `EOS`, and `PAD` delimit and batch the sequences. The closing command `Z` carries no coordinate. We retain the original contour order so that the generated sequence preserves explicit path semantics and can be directly rendered or edited without a separate vectorization stage.

3.2.2 Visual Structure Prefix

The SVG generator follows a decoder-only Transformer architecture (Vaswani et al., 2017; Zhang et al., 2022). The structure condition is a Heiti rendering of the target character. A frozen DINOv2 encoder (Oquab et al., 2023) extracts a contextualized class token together with spatial patch tokens. A learned projection maps these features to the hidden dimension of the SVG generator, producing the structure prefix $\mathbf{V}_c$. The prefix is placed before `BOS` and the SVG tokens in the causal sequence. Consequently, every generated token can attend to both the global character representation and spatially localized stroke evidence.

Let $\mathbf{Y}_{f,c} = (y_1, \ldots, y_T)$ denote the target SVG sequence for character c in font f, and let $\mathbf{D}_f$ denote the dense style condition. The model factorizes the conditional distribution as

$$p_\theta(\mathbf{Y}_{f,c} \mid \mathbf{V}_c, \mathbf{D}_f) = \prod_{t=1}^{T} p_\theta\left(y_t \mid y_{<t}, \mathbf{V}_c, \mathbf{D}_f\right). \tag{1}$$

Training uses next-token cross-entropy over valid SVG positions. Structure prefix positions and padding positions are excluded from the loss. Generation starts from `BOS` and terminates when `EOS` is emitted or the maximum sequence length is reached.

Inference-time recovery. During autoregressive inference, we monitor completed SVG commands for short repetitive cycles. Once a loop is detected, decoding rolls back to the beginning of the final repeated copy, blocks the rejected continuation at the same prefix, rebuilds the KV cache for the preserved sequence, and resumes sampling. Recovery attempts are bounded, while trajectories without detected repetitions remain unchanged.

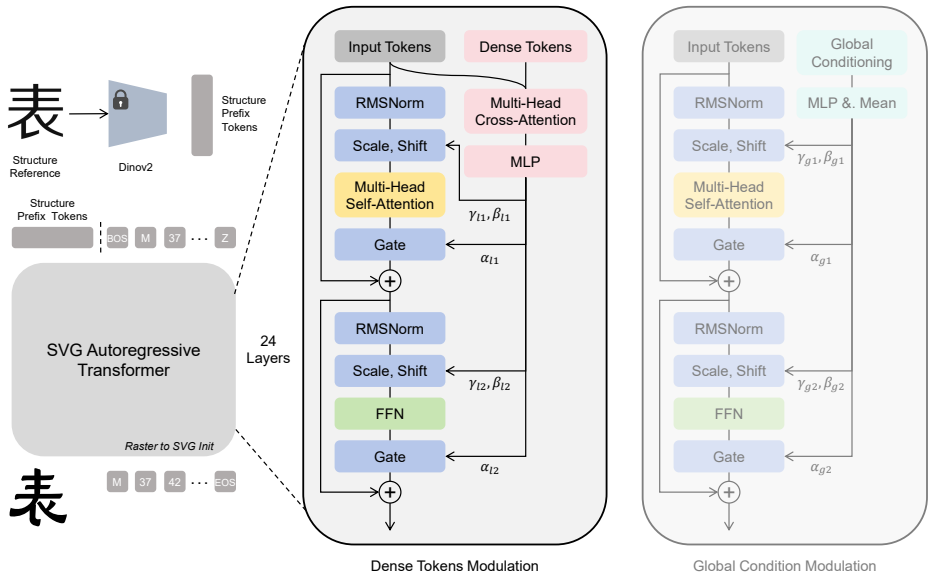


Figure 2: Overview of LITEGLYPH. The target-character structure reference is encoded into prefix tokens for the autoregressive SVG Transformer. The reference glyphs are compressed into dense style tokens, which provide query-adaptive modulation at every layer of the autoregressive SVG Transformer.

3.3 FULL-DEPTH DENSE STYLE MODULATION

3.3.1 DENSE REFERENCE-SET ENCODING

The style condition consists of multiple glyph images sampled from the target font, with the target character strictly excluded. A lightweight CNN with shared weights independently extracts spatial features from every reference. Because the encoder is shared across references, the representation focuses on recurring typeface attributes rather than the identity of an individual reference character.

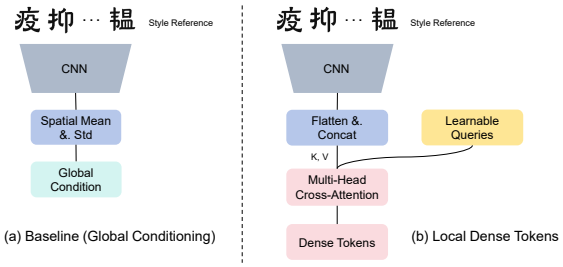


Figure 3: Reference-set style encoding. Global pools reference features into one condition; Dense retains local evidence using learned queries.

The spatial features from all references are flattened and collected into a shared style memory. A set of learnable queries attends to this memory through a Perceiver-style resampler (Alayrac et al., 2022), producing a fixed number of dense style tokens $\mathbf{D}_f$. These tokens retain localized evidence of stroke thickness, terminals, curvature, contrast, and other fine-grained characteristics. They are used as an external style memory and are not appended to the causal SVG sequence.

The global conditioning branch shown in Figures 2 and 3 represents the ablation baseline. It is not part of the final LITEGLYPH architecture, which uses the dense style pathway alone.

3.3.2 QUERY-ADAPTIVE STYLE MODULATION

Dense style modulation is applied to every Transformer layer. At layer l, the RMS-normalized hidden states (Zhang & Sennrich, 2019) of the BOS and SVG positions serve as queries, while the dense style tokens provide keys and values:

$$\mathbf{R}_l = \mathrm{MHA}_l\left(\mathrm{RMSNorm}(\mathbf{H}_l^{\mathrm{svg}}), \mathbf{D}_f, \mathbf{D}_f\right). \tag{2}$$

The retrieved representation $\mathbf{R}_l$ is not directly added to the decoder hidden states. Instead, a lightweight modulation network predicts two sets of token-wise parameters:

$$\left[\boldsymbol{\beta}_l^{\text{attn}}, \boldsymbol{\gamma}_l^{\text{attn}}, \boldsymbol{\alpha}_l^{\text{attn}}, \boldsymbol{\beta}_l^{\text{ffn}}, \boldsymbol{\gamma}_l^{\text{ffn}}, \boldsymbol{\alpha}_l^{\text{ffn}}\right] = \mathcal{Z}_l\left(\text{SiLU}(\mathbf{R}_l)\right). \tag{3}$$

Here, $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ control the scale and shift of RMSNorm, while $\boldsymbol{\alpha}$ controls the corresponding residual update. For a sublayer transformation $\mathcal{F}_l$, the modulated update is

$$\widehat{\mathbf{H}}_l = (1 + \boldsymbol{\gamma}_l) \odot \text{RMSNorm}(\mathbf{H}_l) + \boldsymbol{\beta}_l, \qquad \mathbf{H}'_l = \mathbf{H}_l + (1 + \boldsymbol{\alpha}_l) \odot \mathcal{F}_l(\widehat{\mathbf{H}}_l). \tag{4}$$

This update is applied separately to the self-attention and feed-forward sublayers. Since its parameters vary across SVG positions, the model can retrieve different style evidence for different commands and coordinates. Style modulation is restricted to `BOS` and SVG positions, leaving the structure prefix unchanged.

Each decoder layer has an independent query projection, style key and value projections, and modulation network. The final projection of $\mathcal{Z}_l$ is initialized to zero. Therefore, all modulation offsets are zero at the beginning of adaptation, and the model initially reproduces the transferred Raster-to-SVG generator before progressively learning style control.

During autoregressive inference, the dense style keys and values are computed once and cached independently at each layer. Standard KV caching is also used for the structure prefix and generated SVG history.

Style guidance. To make style strength controllable at inference, we randomly replace the dense style memory with a learned null representation during training while retaining the same structure prefix and SVG history. At each decoding step, following the classifier-free guidance principle (Ho & Salimans, 2022), we combine the conditional and style-null logits. This operation strengthens the contribution of the reference style without removing the target-character constraint. The dropout probability and guidance scale are reported in Section 4.1.

3.4 Raster-to-SVG Initialization

Before structure and style adaptation, we train the same autoregressive generator to convert a raster glyph into the SVG outline of the same character and typeface. This stage uses only the training-font partition; the fonts reserved for full-benchmark UFSC and UFUC are excluded from both Raster-to-SVG initialization and downstream adaptation. The frozen DINOv2 encoder and visual projection produce the same prefix interface later used by the few-shot model. Both stages also share the SVG vocabulary, sequence representation, causal decoder, token embeddings, and output head.

This initialization exposes the generator to valid command transitions, contour organization, and raster-to-vector geometry before style adaptation; the visual projection and SVG generator transfer to the downstream model, while the reference-set encoder, Perceiver resampler, and dense modulation pathway are introduced there. Raster-to-SVG initialization therefore adds no module or inference stage to the final system. Section 4.4 examines when this initialization is beneficial as downstream font diversity increases.

4 Experiments

4.1 Implementation Details

Raster-to-SVG initialization. We pretrain the visual projection and autoregressive SVG decoder on paired glyph images and SVG sequences, using frozen DINOv2 features (Oquab et al., 2023) and the next-token objective in Section 3.2. The same representation and generator are transferred to the few-shot task. Pretraining uses only the 13.5M training pairs from 2,376 fonts; the 268 fonts reserved for UFSC and UFUC are excluded. Pretraining settings are given in Appendix B.

Structure and style adaptation. The downstream model conditions the transferred 350M-parameter SVG decoder on a Heiti rendering of the target character and $K = 8$ style references from the target font, excluding the target character, and is trained jointly for 400k updates; full architecture and optimization settings are given in Appendix B.

Table 1: **Comparison with vector-domain methods on fonts held out from downstream adaptation.** For our model, these fonts were included in Raster-to-SVG initialization. All SVG outputs are rasterized under the same configuration. Darker gray with bold text marks the best value; lighter gray with underlined text marks the second best.

Method	Adaptation Holdout				
	FID ↓	LPIPS ↓	L1 ↓	RMSE ↓	SSIM ↑
DeepVecFont	241.693	0.388	0.297	0.511	0.590
DeepVecFont v2	<u>72.876</u>	<u>0.295</u>	<u>0.204</u>	<u>0.418</u>	**0.640**
DualVector	235.070	0.350	0.211	0.431	<u>0.635</u>
LITEGLYPH	**3.686**	**0.173**	**0.117**	**0.278**	0.606

Inference. Unless stated otherwise, decoding uses temperature 0.8, top-k 50, nucleus sampling with $p = 0.95$ (Holtzman et al., 2020), and style guidance scale 1.5 with KV caching; all comparisons share target glyphs, reference sets, and random seeds, and the decoding procedure is detailed in Appendix B.1. Loop-aware rollback is evaluated separately in Section 4.4 and is not enabled in the main tables.

4.2 Datasets and Evaluation

Dataset splits. We use the large-scale Chinese raster–vector corpus introduced in Section 3.1, with 2,376 seen and 268 unseen fonts and 6,093 seen versus 670 unseen characters. The seen-font, seen-character subset is divided into 13.5M training and 0.7M validation samples, and the unseen-font evaluation sets contain 1.6M seen-character and 0.18M unseen-character samples; exact counts are reported in Appendix A.2.

Evaluation metrics. All generated SVG glyphs are rasterized using the same renderer and resolution. We report FID (Heusel et al., 2017), LPIPS (Zhang et al., 2018), L1, RMSE, and SSIM (Wang et al., 2004) for visual fidelity. For vector-specific analysis, we additionally measure SVG validity and contour Chamfer distance (Fan et al., 2017). EOS termination and command-loop frequency are reported in the decoding ablation.

4.3 Comparisons with Prior Methods

Vector-domain methods. We compare LITEGLYPH with DeepVecFont (Wang & Lian, 2021), DeepVecFont-v2 (Wang et al., 2023), and DualVector (Liu et al., 2023). To accommodate the limited character sets and sequence lengths supported by these methods, we derive a smaller dataset from our full corpus and train and evaluate all four models on it. For our model, the 65 fonts held out from this smaller dataset's downstream training were already present in Raster-to-SVG initialization. Table 1 therefore measures transfer to fonts unseen during adaptation, not end-to-end unseen-font generalization; it is distinct from the full-corpus UFSC split. Appendix C.2 details the dataset construction and evaluation protocol, and Figure 4 shows predictions on matched targets.

Raster font generation baselines. We compare with FontDiffuser (Yang et al., 2024), IF-Font (Chen et al., 2024), VQ-Font (Pan et al., 2023), and GAR-Font (Cai et al., 2026). These methods synthesize target glyphs directly in the raster domain. All four were retrained on our corpus rather than evaluated solely from released end-model checkpoints; their common training pool contains 13.5M glyph pairs from 2,376 fonts, with method-specific training stages described in Appendix C.2. For a consistent comparison, we rasterize our generated SVGs and evaluate all predictions using the same metric implementation at each method's native output resolution with resolution-matched ground truth. Figure 13 (Appendix E) shows representative outputs under matched inputs.

Multimodal LLM baselines. We further evaluate Claude Opus 5 (Anthropic, 2026), Qwen3.8-Flash (Alibaba Cloud, 2026), and DeepSeek V4 Flash Vision Exp (DeepSeek-AI, 2026) as general purpose multimodal baselines. Each model receives one target-character structure image and the same eight style reference images used by our method, together with a fixed prompt requesting a complete SVG glyph. All three models directly generate SVG code in a zero-shot setting without

Table 2: **Comparison with font-generation baselines.** Results on validation, UFSC, and UFUC use 10k target subsets per split; multimodal LLM metrics exclude invalid outputs and therefore use model-specific retained subsets (Appendix C.2). Predictions are evaluated at each method's native resolution against resolution-matched ground truth. Within each split, darker gray with bold text marks the best value and lighter gray with underlined text marks the second best; – denotes not evaluated.

Method	Validation					UFSC					UFUC				
	FID↓	LPIPS↓	L1↓	RMSE↓	SSIM↑	FID↓	LPIPS↓	L1↓	RMSE↓	SSIM↑	FID↓	LPIPS↓	L1↓	RMSE↓	SSIM↑
FontDiffuser	11.951	0.233	0.149	0.320	0.530	13.310	0.237	0.149	0.320	0.529	13.245	0.239	0.151	0.322	0.525
IF-Font	8.086	0.229	0.143	0.330	0.573	8.634	0.228	0.141	0.327	0.576	8.993	0.239	0.149	0.339	0.562
VQ-Font	9.674	0.192	**0.112**	**0.272**	0.639	10.238	0.199	**0.116**	**0.280**	0.626	10.434	0.202	**0.119**	**0.284**	0.620
GAR-Font	20.167	**0.151**	0.124	**0.272**	0.618	20.067	**0.183**	0.142	0.301	0.562	22.086	0.213	0.159	0.328	0.520
DeepSeek V4 Flash Vision			–					–			91.978	0.358	0.297	0.522	0.528
Qwen3.8-Flash			–					–			50.901	0.361	0.307	0.531	0.516
Claude Opus 5			–					–			36.373	0.357	0.303	0.527	0.512
Ours (Global)	1.841	0.175	0.123	0.334	0.764	2.802	0.201	0.141	0.360	0.742	**2.743**	0.204	0.144	0.364	0.739
Ours (Dense, $K = 1$)	2.009	0.188	0.133	0.349	0.751	2.942	0.209	0.149	0.371	0.733	2.959	0.212	0.151	0.374	0.731
Ours (Dense, $K = 8$)	**1.838**	0.174	0.122	0.333	**0.765**	**2.759**	0.198	0.139	0.357	**0.745**	2.759	**0.201**	0.142	0.361	**0.741**

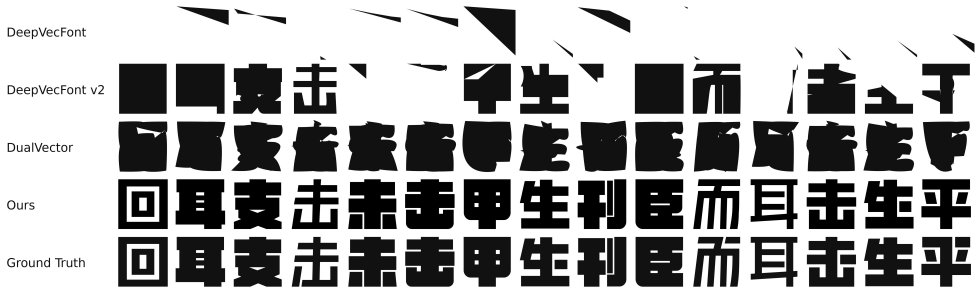


Figure 4: **Qualitative comparison with vector-domain methods on fonts held out from downstream adaptation.** Each row uses the same target character, Heiti structure reference, and style references. All predictions are rendered under the same settings; the ground-truth glyph is shown for comparison.

task-specific training. Their outputs are parsed, normalized, and rendered under the same evaluation configuration as our SVG predictions. Owing to inference cost, these models are evaluated on the unseen-font, unseen-character setting.

4.4 Ablation Studies

Full-depth dense style modulation. We isolate the effect of the style-conditioning design by comparing our final Dense model with the Global baseline. Both variants use the same structure prefix, autoregressive decoder, reference sampling, training data, and optimization schedule. Global compresses the reference set into a single font-level condition and uses it throughout the decoder. In contrast, Dense preserves localized reference evidence as Perceiver-resampled style tokens and performs query-adaptive modulation at every Transformer layer. Table 2 reports the quantitative results on validation, UFSC, and UFUC, while Figure 5 compares local stroke thickness, curvature, and terminals under identical inputs.

Data-efficient Raster-to-SVG initialization. We examine whether Raster-to-SVG initialization reduces the downstream font diversity required for structure and style adaptation. Nested training subsets of increasing font diversity are constructed with fixed seeds; in each pair, one model is initialized from the Raster-to-SVG checkpoint and one is trained from random initialization, with matched fonts, characters, architecture, update budget, and optimization schedule, and all variants are evaluated on identical validation, UFSC, and UFUC samples.

Figure 6 shows the largest initialization gain with 250 training fonts; the gap narrows as font diversity increases. At full scale, the reported metrics are mixed rather than favoring one model across the board. Together, the controlled subset experiments support initialization primarily as a data-efficient

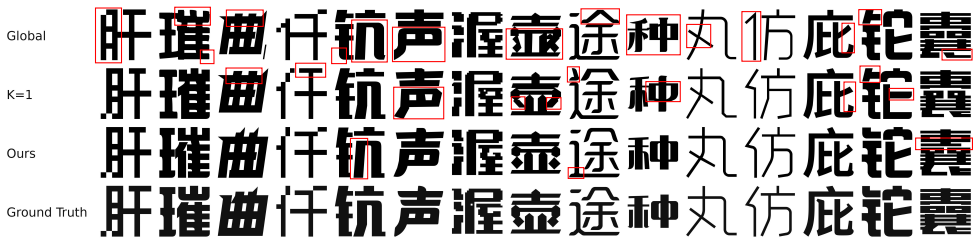


Figure 5: **Effect of full-depth dense style modulation.** Global and Dense use identical structure images and style-reference sets. Aligned close-ups highlight differences in stroke weight, curvature, and terminals.

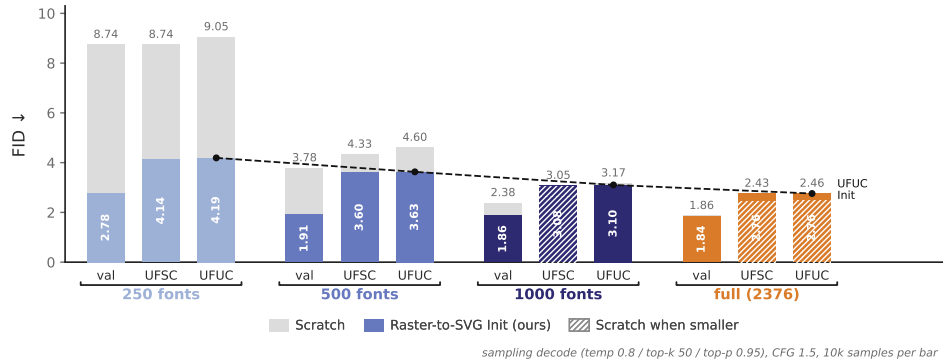


Figure 6: **Effect of Raster-to-SVG initialization across downstream font scales.** We vary the number of downstream training fonts while keeping the Raster-to-SVG checkpoint, architecture, character set, and optimization budget fixed within each matched pair. Results are reported on validation, UFSC, and UFUC.

vector prior, not a consistent source of final-quality gains when target-task data are abundant. This scale dependence is qualitatively consistent with observations in visual transfer learning (He et al., 2019). The exact subset sizes and additional UFUC metrics are reported in Appendix D.2.

Loop-aware rollback. Using the same checkpoint, samples, and decoding configuration, rollback suppresses repetitive trajectories without degrading normally decoded outputs; quantitative results are reported in Appendix D.3 (Table 9).

Number of style references. We assess reference availability with the same Dense checkpoint using one or eight style exemplars; Table 2 reports both settings on matched target splits.

5 Conclusion and Limitations

We presented LiteGlyph, a unified autoregressive framework for few-shot Chinese vector font generation and a large-scale paired raster–vector corpus. Our results show that complete SVG glyphs can be generated directly by separating target-character structure from cross-character font style: structure provides a stable geometric scaffold, while dense reference features provide localized style evidence throughout decoding. Raster-to-SVG initialization further acts as a data-efficient vector prior when downstream font diversity is limited. Together, these results support factorized structure–style conditioning as a practical approach to few-shot generation of editable vector glyphs. Our evaluation is limited to Chinese fonts, and transfer to other writing systems remains untested. Autoregressive decoding may still produce repetitive command cycles, which rollback mitigates rather than eliminates. Finer coordinate quantization and broader vector representations remain directions for future work.

AI USE STATEMENT

We used generative AI tools to discuss methodological alternatives and experimental designs, assist with the implementation and debugging of data processing, training, and evaluation code, and analyze preliminary results. We also used these tools to identify and summarize related work, draft and edit manuscript text, and refine figure layouts. The authors made the final research decisions, reviewed AI-assisted code and writing, checked references against their original sources, and verified reported results against experimental outputs. We take responsibility for the final content of this paper, including AI-assisted text, code, and figures.

ETHICS STATEMENT

Our study uses font files collected for research on Chinese glyph generation. Because fonts may be subject to different licenses, any release of font files or derived assets will respect their respective redistribution terms. Generated glyphs may resemble existing typefaces; practical use should therefore consider the rights associated with reference fonts. Our experiments focus on Chinese fonts, and we do not claim that the measured performance extends to other writing systems.

REPRODUCIBILITY STATEMENT

The method and experimental sections describe the model, training stages, decoding procedure, dataset splits, and evaluation metrics. The appendix provides further details on corpus construction, SVG preprocessing, baseline protocols, and additional experiments. We use fixed data splits and evaluation references to enable comparisons under the same protocol. To support direct verification of our results, we provide an anonymized repository at https://anonymous.4open.science/r/LiteGlyph-D4F2 containing the corpus-construction and evaluation code, the font manifest with per-font source and license information, the exact seen/unseen font and character partition lists used in every experiment, and a script that programmatically checks the disjointness of all training stages—including Raster-to-SVG initialization—from the 268 unseen evaluation fonts and 670 unseen characters used in UFSC and UFUC. Glyph-level derived assets and trained checkpoints will additionally be distributed at camera-ready for the subset of fonts whose licenses permit redistribution; for the remaining fonts we release identifiers and source pointers so that the corpus can be reconstructed from the original sources. For the small-dataset protocol of Table 1, we release the exact held-out font lists, documenting the fonts that were included in our model's Raster-to-SVG initialization.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L. Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikołaj Bińkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: A visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-1723. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/960a172bc7fbf0177ccccbb411a7d800-Abstract-Conference.html. 5, 15

Alibaba Cloud. Qwen3.8-Flash model documentation. Alibaba Cloud Model Studio, 2026. URL https://help.aliyun.com/en/model-studio/qwen3-8-flash. Accessed: 2026-09-21. 7

Anthropic. Introducing Claude Opus 5. Anthropic, 2026. URL https://www.anthropic.com/news/claude-opus-5. Accessed: 2026-09-21. 7

Haruka Aoki and Kiyoharu Aizawa. SVG vector font generation for chinese characters with transformer. In *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 646–650, 2022. doi: 10.1109/ICIP46576.2022.9897633. URL https://arxiv.org/abs/2206.10329. 3

Haonan Cai, Yuxuan Luo, and Zhouhui Lian. Beyond patches: Global-aware autoregressive model for multimodal few-shot font generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 517–528, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Cai_Beyond_Patches_Global-aware_Autoregressive_Model_for_Multimodal_Few-Shot_Font_Generation_CVPR_2026_paper.html. 3, 7, 18

Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. DeepSVG: A hierarchical generative network for vector graphics animation. In *Advances in Neural Information Processing Systems*, volume 33, pp. 16351–16361, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/bcf9d6bd14a2095866ce8c950b702341-Abstract.html. 3

Xinping Chen, Xiao Ke, and Wenzhong Guo. IF-Font: Ideographic description sequence-following font generation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 14177–14199, 2024. doi: 10.52202/079017-0454. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/19ded4cfc36a7feb7fce975393d378fd-Abstract-Conference.html. 3, 7, 18

Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, Guosheng Lin, and Chi Zhang. MeshAnything: Artist-created mesh generation with autoregressive transformers. In *International Conference on Learning Representations (ICLR)*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/7fc69f53a6ca2664ef23a9597903ca6f-Abstract-Conference.html. 3

Yiwen Chen, Yikai Wang, Yihao Luo, Zhengyi Wang, Zilong Chen, Jun Zhu, Chi Zhang, and Guosheng Lin. MeshAnything V2: Artist-created mesh generation with adjacent mesh tokenization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13922–13931, October 2025b. URL https://openaccess.thecvf.com/content/ICCV2025/html/Chen_MeshAnything_V2_Artist-Created_Mesh_Generation_with_Adjacent_Mesh_Tokenization_ICCV_2025_paper.html. 3

DeepSeek-AI. DeepSeek-V4-Flash-Vision-Exp: Multimodal api release. DeepSeek API Documentation, 2026. URL https://api-docs.deepseek.com/news/news260821/. Accessed: 2026-09-21. 7

Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 605–613, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Fan_A_Point_Set_CVPR_2017_paper.html. 7, 17

fontTools Project. fontTools documentation. Read the Docs, 2026. URL https://fonttools.readthedocs.io/en/latest/. Accessed: 2026-09-23. 14

Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking ImageNet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4918–4927, 2019. URL https://openaccess.thecvf.com/content_ICCV_2019/html/He_Rethinking_ImageNet_Pre-Training_ICCV_2019_paper.html. 9

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. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html. 7, 17

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598. 6

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH. 7

Xiaoke Huang, Bhavul Gauri, Kam Woh Ng, Tony Ng, Mengmeng Xu, Zhiheng Liu, Weiming Ren, Zhaochong An, Zijian Zhou, Haonan Qiu, Yuyin Zhou, Sen He, Ziheng Wang, Tao Xiang, and Xiao Han. VecGlypher: Unified vector glyph generation with language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24373–24383, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Huang_VecGlypher_Unified_Vector_Glyph_Generation_with_Language_Models_CVPR_2026_paper.html. 2, 3

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 (CVPR)*, pp. 7905–7914, June 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Liu_XMP-Font_Self-Supervised_Cross-Modality_Pre-Training_for_Few-Shot_Font_Generation_CVPR_2022_paper.html. 1, 3

Ying-Tian Liu, Zhifei Zhang, Yuan-Chen Guo, Matthew Fisher, Zhaowen Wang, and Song-Hai Zhang. DualVector: Unsupervised vector font synthesis with dual-part representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14193–14202, June 2023. doi: 10.1109/CVPR52729.2023.01364. URL https://openaccess.thecvf.com/content/CVPR2023/html/Liu_DualVector_Unsupervised_Vector_Font_Synthesis_With_Dual-Part_Representation_CVPR_2023_paper.html. 2, 3, 4, 7, 17

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision, 2023. URL https://arxiv.org/abs/2304.07193. 4, 6, 15

Wei Pan, Anna Zhu, Xinyu Zhou, Brian Kenji Iwana, and Shilin Li. Few shot font generation via transferring similarity guided global style and quantization local style. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19506–19516, October 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Pan_Few_Shot_Font_Generation_Via_Transferring_Similarity_Guided_Global_Style_ICCV_2023_paper.html. 3, 7, 18

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 (ICCV)*, pp. 13900–13909, October 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Park_Multiple_Heads_Are_Better_Than_One_Few-Shot_Font_Generation_With_ICCV_2021_paper.html. 1, 3

Juan A. Rodriguez, Abhay Puri, Shubham Agarwal, Issam H. Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. StarVector: Generating scalable vector graphics code from images and text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16175–16186, June 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Rodriguez_StarVector_Generating_Scalable_Vector_Graphics_Code_from_Images_and_Text_CVPR_2025_paper.html. 3

Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. MeshGPT: Generating triangle meshes with decoder-only transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19615–19625, June 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Siddiqui_MeshGPT_Generating_Triangle_Meshes_with_Decoder-Only_Transformers_CVPR_2024_paper.html. 3

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 (CVPR)*, pp. 7895–7904, June 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Tang_Few-Shot_Font_Generation_by_Learning_Fine-Grained_Local_Styles_CVPR_2022_paper.html. 1, 3

Vikas Thamizharasan, Difan Liu, Shantanu Agarwal, Matthew Fisher, Michaël Gharbi, Oliver Wang, Alec Jacobson, and Evangelos Kalogerakis. VecFusion: Vector font generation with diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7943–7952, June 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Thamizharasan_VecFusion_Vector_Font_Generation_with_Diffusion_CVPR_2024_paper.html. 3

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html. 4

Yizhi Wang and Zhouhui Lian. DeepVecFont: Synthesizing high-quality vector fonts via dual-modality learning. *ACM Transactions on Graphics*, 40(6):265:1–265:15, December 2021. doi: 10.1145/3478513.3480488. URL https://doi.org/10.1145/3478513.3480488. 2, 3, 4, 7, 17

Yuqing Wang, Yizhi Wang, Longhui Yu, Yuesheng Zhu, and Zhouhui Lian. DeepVecFont-v2: Exploiting transformers to synthesize vector fonts with higher quality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18320–18328, June 2023. doi: 10.1109/CVPR52729.2023.01757. URL https://openaccess.thecvf.com/content/CVPR2023/html/Wang_DeepVecFont-v2_Exploiting_Transformers_To_Synthesize_Vector_Fonts_With_Higher_Quality_CVPR_2023_paper.html. 2, 3, 4, 7, 17

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. doi: 10.1109/TIP.2003.819861. URL https://doi.org/10.1109/TIP.2003.819861. 7, 17

World Wide Web Consortium. Scalable vector graphics (SVG) 2. W3C Candidate Recommendation, 2018. URL https://www.w3.org/TR/2018/CR-SVG2-20181004/. 4 October 2018. 4, 16

Ronghuan Wu, Wanchao Su, and Jing Liao. Chat2SVG: Vector graphics generation with large language models and image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23690–23700, June 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wu_Chat2SVG_Vector_Graphics_Generation_with_Large_Language_Models_and_Image_CVPR_2025_paper.html. 3

Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. SVGDreamer: Text guided svg generation with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4546–4555, June 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Xing_SVGDreamer_Text_Guided_SVG_Generation_with_Diffusion_Model_CVPR_2024_paper.html. 3

Ximing Xing, Juncheng Hu, Guotao Liang, Jing Zhang, Dong Xu, and Qian Yu. Empowering LLMs to understand and generate complex vector graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19487–19497, June 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Xing_Empowering_LLMs_to_Understand_and_Generate_Complex_Vector_Graphics_CVPR_2025_paper.html. 3

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*, volume 38, pp. 6603–6611, 2024. doi: 10.1609/aaai.v38i7.28482. URL https://doi.org/10.1609/aaai.v38i7.28482. 3, 7, 18

Hao Yuan, Yuxuan Luo, Xing Chen, and Zhouhui Lian. VecFontLLM: Anchor-guided direct synthesis of chinese vector fonts, 2026. URL https://arxiv.org/abs/2607.17251. 2, 3

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html. 5

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*, pp. 586–595, 2018. doi: 10.1109/CVPR.2018.00068. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_The_Unreasonable_Effectiveness_CVPR_2018_paper.html. 7, 17

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open pre-trained transformer language models, 2022. URL https://arxiv.org/abs/2205.01068. 4, 15

A Corpus Construction and Quality Control

This section summarizes how the paired raster–SVG corpus was curated and partitioned. Further dataset statistics appear in Table 3 and Figure 7.

A.1 Font Curation and Glyph Processing

We assembled TrueType and OpenType fonts from public Chinese-font collections. We removed unreadable files, exact duplicates, and fonts with insufficient Simplified Chinese coverage using font metadata and character maps checked with FontTools (fontTools Project, 2026). Visual screening further excluded unsuitable typefaces. Of 3,082 candidates entering glyph construction, 2,644 yielded usable fonts in the final corpus.

Composite glyphs are decomposed into component contours, converted from quadratic to cubic Bézier form as absolute `M`, `L`, `C`, and `Z` commands, and normalized to a 64×64 view box. Coordinates are quantized into 128 bins per axis. Paired images are rendered from the canonical SVG at 896×896 and downsampled to 224×224 with Lanczos filtering. We exclude a font if any target glyph exceeds 2,500 raw SVG tokens, and reject invalid geometry or failed conversions; tokenization–decoding round trips, raster consistency tests, and manifest checks further guard SVG syntax and raster–vector alignment. The final corpus contains 15,960,970 aligned raster–SVG pairs. A post-packaging audit found matching manifest and index records and no asset errors in 5,000 sampled reads.

A.2 Partitions and Leakage Prevention

We use deterministic hash-based font and character splits with a fixed seed: 2,376 seen and 268 unseen fonts, and 6,093 seen and 670 unseen characters. Training and validation draw from seen fonts and characters; unseen fonts with seen characters form UFSC, and the same unseen-font pool with unseen characters forms UFUC. The validation pairs are selected deterministically within the seen-font, seen-character pool. Table 3 reports the resulting sample counts. Font and character identities do not cross their respective seen/unseen boundaries.

A Heiti rendering of the target character provides the structure condition. Style references are sampled from the target font with the target character excluded. Sampling varies across training epochs but is fixed for evaluation, ensuring that each method receives the same references for a given target.

Table 3: **Corpus partitions.** Sample counts refer to aligned raster–SVG glyph pairs. UFSC and UFUC contain the same unseen-font pool.

Split	Fonts	Characters	Glyph pairs
Train	2,376 seen	6,093 seen	13,476,734
Validation	2,375 seen	6,093 seen	708,064
UFSC	268 unseen	6,093 seen	1,600,181
UFUC		670 unseen	175,991
Total	2,644	6,763	15,960,970

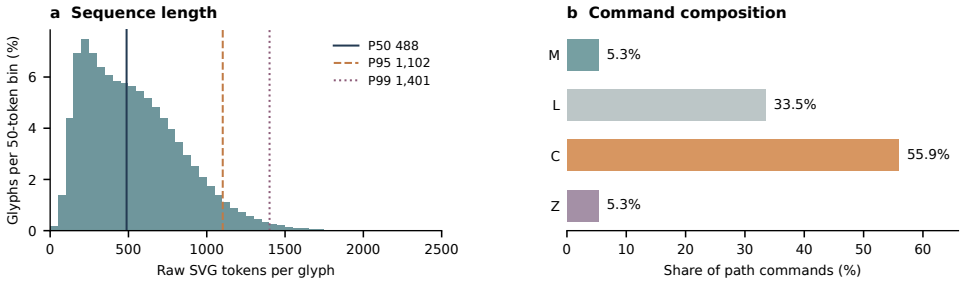


Figure 7: **SVG sequence statistics.** (a) Raw token lengths across 15,960,970 glyphs. (b) Command frequencies in a deterministic sample of 79,817 glyphs.

A.3 Sequence Statistics

Across the corpus, the median raw SVG length is 488 tokens, with P95 at 1,102 and P99 at 1,401; the maximum is 2,495. Figure 7 shows the full length distribution and the relative frequencies of path commands estimated from a deterministic sample of 79,817 glyphs.

B Model and Training Details

Table 4 gathers the settings needed to reproduce the Dense model in Section 3. The frozen DINOv2 structure encoder supplies the visual prefix, and a shared CNN followed by a Perceiver-style resampler supplies dense style tokens (Oquab et al., 2023; Alayrac et al., 2022). The generator is an OPT-based causal Transformer (Zhang et al., 2022). The style branch modulates all decoder layers; the Global branch in the architecture figures is an ablation baseline rather than part of the final model.

Raster-to-SVG initialization. The initial stage pairs each glyph image with its own SVG sequence and trains the visual projection and decoder with next-token cross-entropy. It uses only the 13,476,734 training pairs from 2,376 fonts; none of the 268 full-benchmark UFSC/UFUC fonts is used for Raster-to-SVG training. The downstream stage transfers the projection, decoder, token embeddings, and output head, then trains them jointly with the new style branch while keeping DINOv2 frozen. The initial stage uses a global batch of 32 (four samples on each of eight GPUs) for one epoch, with 3,000 warmup updates from 10^{-6} to 5×10^{-5} followed by cosine decay to 10^{-6}. We initialize the downstream stage from its final checkpoint at update 421,147; it is not selected using downstream evaluation results.

Style condition. For each downstream training target, K distinct reference characters are sampled from the same font, excluding the target character. Reference order is shuffled during training. At evaluation, target and reference identifiers are fixed across methods. Null-style dropout replaces the complete style condition with a learned null representation, retaining the structure prefix and SVG history. The shared CNN has four downsampling stages with output widths 64, 128, 256, and 256, producing a 14×14 feature grid from a 224×224 reference. A two-layer, eight-head

Table 4: **Architecture and training configuration.** Values are for the final Dense model unless a comparison explicitly changes them.

Component	Value	Component	Value
Total / trainable	680.0M / 375.6M (architecture-derived)	SVG decoder	350M, 24 layers
Structure encoder	Frozen DINOv2 + learned projection	Style encoder	Shared-weight CNN
Style resampler	64 learned queries; 2 layers, 8 heads	Style modulation	All 24 decoder layers
Style references	$K = 8$; target excluded	SVG budget	2,500 raw tokens
Coordinate bins	128 per axis	Structure image	224×224
Optimizer	AdamW, $\beta = (0.9, 0.999)$	Weight decay / clipping	0.1 / 1.0
Peak learning rate	5×10^{-5}	Schedule	3k warmup; cosine to 10^{-6}
Adaptation updates	400k	Global batch size	32 (4 per GPU)
Hardware	8 NVIDIA H800 GPUs	Precision	FP16; gradient checkpointing
Null-style dropout	0.1	Style guidance scale	1.5
Sampling	$T = 0.8$, top-$k = 50$, top-$p = 0.95$	Caching	Decoder and per-layer style KV
Rollback	Bounded loop-aware recovery	Seed	Train: 0; references: 20260809

resampler compresses the reference features into 64 learned-query tokens. Reference selection uses seed 20260809.

B.1 SVG Tokenization and Decoding

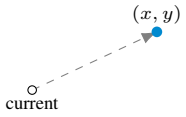
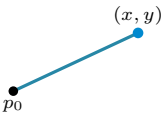
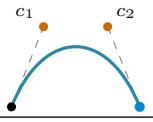
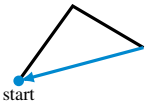
Each outline is serialized as one absolute path using `M`, `L`, `C`, and `Z` according to SVG 2 (World Wide Web Consortium, 2018). Command symbols, 128-bin coordinates, and sequence markers share a discrete vocabulary. `Z` carries no coordinates. The 2,500-token data limit counts commands and coordinates before `BOS`/`EOS`/`PAD`; prefix and padding positions are excluded from the training loss. Generation begins with `BOS`, predicts the SVG one token at a time, and stops at `EOS` or the model's output limit.

B.1.1 Command Vocabulary

Table 5 makes the command arguments and contour closure explicit. The geometry examples are explanatory and do not add tokens to the sequence.

Loop-aware rollback. After each completed command, the decoder checks for repeated command-level cycles. When triggered, it restores the prefix preceding the final repeated copy, rejects the sampled continuation at that point, reconstructs the decoder KV cache, and resumes sampling up to a bounded number of attempts. The detector matches three consecutive copies of an identical command block of length 1–12; both commands and quantized coordinates must match exactly, with no geometric tolerance. On detection, it discards the third copy, preserves the first two, and rejects the offending continuation before resampling at temperature 1.0 without top-k or top-p restriction. Recovery allows at most three rollbacks per glyph; an additional detected loop stops generation with a recovery-exhausted status. We report residual loop rate, EOS termination, SVG validity, and raster quality in Section 4.4.

Table 5: **SVG command vocabulary.** Coordinates are absolute and uniformly quantized. Dashed lines and hollow points illustrate geometric semantics and are not emitted into the SVG sequence.

Command	Arguments	Semantics	Geometric example
M	$x,\ y$	Moves the current point to (x, y) without drawing and starts a new contour. Every contour begins with one M.	
L	$x,\ y$	Draws a straight segment from the current point p_0 to the endpoint (x, y).	
C	x_1, y_1, x_2, y_2 x, y	Draws a cubic Bézier segment from p_0 to (x, y) using (x_1, y_1) and (x_2, y_2) as control points.	
Z	∅	Closes the current contour by drawing a segment from its last point back to the point established by the most recent M. The command carries no coordinate token.	
BOS/EOS	∅	Marks sequence boundaries. Decoding starts from BOS and terminates when EOS is emitted or the maximum length is reached.	BOS → M ··· Z → EOS
PAD	∅	Pads variable-length sequences for batching and is excluded from the training loss.	Z PAD PAD

C Evaluation and Baseline Protocols

The comparisons in Section 4.3 use fixed target identifiers and, where the input interface permits, fixed structure and style references. A result on a method-compatible subset must be identified separately from a result on the full unseen-font split.

C.1 Metric Computation

For vector outputs, we rasterize generated and reference SVGs with CairoSVG at 512×512 on a white background before computing FID, LPIPS, L1, RMSE, and SSIM (Heusel et al., 2017; Zhang et al., 2018; Wang et al., 2004). LPIPS uses VGG features, while FID uses 2,048-dimensional InceptionV3 features; unless stated otherwise, FID is computed from 10,000 matched glyphs per split. We additionally report contour Chamfer distance using 200 arclength-sampled and independently normalized points per outline (Fan et al., 2017), SVG validity, EOS rate, and residual loop rate. Validity requires a nonempty, closed M/L/C/Z path with legal coordinates; a loop is a command pattern of up to eight commands repeated three times consecutively.

C.2 Baseline Evaluation Protocol

Vector methods. DeepVecFont, DeepVecFont-v2, DualVector (Wang & Lian, 2021; Wang et al., 2023; Liu et al., 2023), and LiteGlyph are trained and evaluated on the same smaller dataset derived from our full Chinese vector-font corpus (Table 1). We rank characters by the number of SVG path commands in their canonical Heiti structure glyphs, a proxy for structural complexity, and retain the 300 simplest. Their mean command count is 22.5, compared with 42.3 before selection. Among fonts covering at least 90 of these characters, we rank each font by the mean command count of its own glyphs on the selected characters and keep the 325 simplest. A deterministic font-level split (seed 0) holds out 65 fonts as the adaptation holdout and assigns the remaining 260 to training. The resulting dataset contains 71,796 training glyphs, 1,024 validation episodes from training fonts, and 17,772 held-out-font glyphs for evaluation. Each episode pairs the target with eight other characters from the

same font as style references. All four models use approximately 1.55 million training episodes and are scored on matched targets under the same rendering and metric protocol. These 65 fonts are held out from the smaller dataset's downstream training but belong to the full corpus's training-font pool; for our model, their glyphs were seen during Raster-to-SVG initialization. Consequently, Table 1 measures adaptation-time transfer and does not establish end-to-end unseen-font generalization.

Raster methods. FontDiffuser, IF-Font, VQ-Font, and GAR-Font (Yang et al., 2024; Chen et al., 2024; Pan et al., 2023; Cai et al., 2026) are adapted and retrained on our corpus, rather than evaluated solely with released end-model checkpoints. The common training pool comprises 13,476,734 glyph pairs from 2,376 fonts; method-specific preprocessing, optimizer schedules, and pretrained components are retained. GAR-Font also uses its native few-shot adaptation stage on reference glyphs from held-out fonts, a different use of references from our inference-only conditioning. We evaluate all four methods at their native output resolutions against resolution-matched ground truth on fixed target lists, following the protocol of prior font-generation benchmarks in which every method is evaluated at its official resolution and hyperparameters (Cai et al., 2026). Ground truth is downsampled to match each resolution, while our SVG outputs are rasterized at 512×512, reflecting the resolution independence of the vector representation; absolute FID values across resolution regimes are therefore not directly comparable, and the comparison is system-level. This asymmetry is itself a consequence of the representation: raster baselines are architecturally bound to fixed output resolutions (the original GAR-Font is limited to 64×64, restricting high-DPI applications), whereas direct SVG generation is resolution-independent by construction. The comparison controls target selection and metric computation, but not model capacity, reference interface, or total training exposure.

Multimodal LLMs. We evaluate Claude Opus 5, Qwen3.8-Flash, and DeepSeek V4 Flash Vision on a fixed 10,000-target UFUC subset. Each model receives one Heiti image of the target character and eight style glyphs from the target font, excluding the target character. The shared prompt (Figure 8) requests one complete SVG with a 256×256 view box, a single black path with even-odd filling, and explicit absolute `M/L/C/Z` commands. We allow at most 16 closed subpaths and 256 commands; truncated SVGs and responses with extraneous text fail validation.

The initial output budget is 4,096 tokens. Qwen and DeepSeek use temperature 1.0 and top-$p = 0.95$ with reasoning disabled; the Claude gateway does not expose explicit sampling parameters. For Qwen and DeepSeek, invalid responses receive at most two further attempts with an 8,192-token budget; Claude uses the separate repair and regeneration procedure reported in Table 6. Valid first responses are not regenerated, and image-quality metrics never trigger a retry. Table 6 reports first-response validity and final retained outputs under each model's recovery protocol; image metrics use those retained outputs.

Table 6: **SVG validity on 10,000 UFUC targets.** First-response rates use the original strict 16-subpath constraint. Qwen and DeepSeek receive up to two format-triggered retries. Claude additionally uses 565 local repairs and 32 regeneration attempts under a relaxed 20-subpath limit; its final rate is therefore not directly comparable.

Model	First response	Final retained	Valid outputs
Claude Opus 5	94.1%	100%	10,000
Qwen3.8-Flash	84.4%	99.26%	9,926
DeepSeek V4 Flash Vision	80.6%	98.97%	9,897

C.3 Vector Quality and Reliability

Table 7 complements raster similarity with validity, contour geometry, and termination behavior. These results use the Dense checkpoint after approximately 400k updates and the same 10,000-sample subsets as the main raster comparison.

TASK
You are an SVG glyph generator.

Keep private reasoning under 200 tokens: do not describe every stroke, component, or coordinate; make only a few terse planning decisions, reserve most of the output budget for the final SVG, and proceed to the final answer immediately.

OUTPUT FORMAT
The final answer must start with <svg and end with </svg>; expose no analysis or explanation. The first image is the black structural reference for the target character; the following images are style references from the same font, each a different character. Reproduce the target character **<CHARACTER>** in the demonstrated style.

Return exactly one complete SVG on one line and nothing else. Do not call tools and do not use Markdown fences. Use exactly one <path> element with fill="#000" and fill-rule="evenodd"; represent disconnected components as multiple closed subpaths inside its single d attribute. Use viewBox="0 0 256 256".

STRUCTURE AND STYLE CONDITIONING
The first structural image is authoritative. Trace its observed component layout and negative-space topology; do not replace it with a canonical glyph recalled from the Unicode name or character decomposition. Preserve the number, placement, relative bounds, gaps, and counters of visible forms. First infer the target glyph silhouette from the structural image. Use the style images only to infer stroke weight, curvature, terminals, contrast, and balance. The output must remain recognizably the target character, not a generic decoration or placeholder pattern.

STRICT PATH GRAMMAR
Symbolic notation only; never output the placeholders. Each subpath is `M x y`, followed by zero or more explicit `L x y` or `C x1 y1 x2 y2 x y` commands, and then `Z`. Every command letter must be written explicitly and must be uppercase.

The only permitted letters in d are M, L, C and Z: never use lowercase commands, implicit repeated coordinates, H, V, Q, S, T or A. Use at most 256 total command letters across the entire d attribute, preferably 40--160, and at most 16 closed subpaths. Keep coordinates within 0..256. Use varied, image-derived coordinates; do not quantize them to a fixed step. Use C curves where the observed outline is curved.

NEGATIVE SPACE
Because fill-rule is evenodd, put every white counter or interior opening inside its enclosing black contour as a closed subpath. Never draw a filled black subpath where the structural image shows a white counter. Preserve counter count, shape, and placement before adding style.

ANTI-LOOP RULES
Never draw a checkerboard, grid, barcode, repeated squares, repeated rectangles, parallel placeholder bars, or any regularly tiled pattern. Never repeat an identical command sequence more than twice. Each subpath must correspond to a distinct visible component of the target glyph.

If approaching a limit, simplify the recognizable glyph outline, close the current subpath with Z, and immediately close the path and SVG. No concrete path coordinates are provided in this instruction and none should be copied.

FINAL VERIFICATION
Before returning, silently verify: the silhouette matches the target image; the style matches the references; there is one SVG and one path using fill-rule="evenodd"; only uppercase M/L/C/Z occur; there are at most 256 commands and 16 subpaths; there is no regular repeated pattern; every subpath ends in Z; and the final characters are </svg>.

Figure 8: **Multimodal LLM prompt.** The same user prompt is used for all models; only <CHARACTER> changes between targets. The system message is empty. Line breaks are adjusted for typesetting.

Table 7: **Vector quality and decoding reliability.** Sampling uses $T = 0.8$, top-$k = 50$, top-$p = 0.95$, and style guidance 1.5, without rollback. Chamfer is averaged over valid outlines; Sec./glyph is eight-GPU generation wall time divided by the 10,000 outputs.

Split	Samples	Valid ↑	Chamfer ↓	EOS ↑	Loop ↓	Sec./glyph ↓
Validation	10,000	99.68%	0.611	99.85%	3.05%	0.620
UFSC	10,000	99.74%	0.761	99.92%	2.67%	0.574
UFUC	10,000	99.67%	0.772	99.86%	2.49%	0.584

D Extended Ablations and Analysis

D.1 Dense Style Modulation

The main comparison in Table 2 evaluates Global against the final Dense model under matched data, structure prefixes, reference sets, decoder, optimization budget, and sampling settings. Global compresses the reference set to one font-level condition; Dense retrieves spatial style evidence at each decoder layer. The model definitions imply 661.9M total and 357.5M trainable parameters for Global, versus 680.0M total and 375.6M trainable for Dense; the latter adds approximately 18.1M trainable parameters. This comparison therefore does not isolate style representation from capacity.

D.2 Raster-to-SVG Initialization Across Font Scales

Section 4.4 compares Raster-to-SVG initialization with random initialization across nested training-font subsets. We randomly order the 2,376 training fonts once and take nested subsets of 250, 500, and 1,000 fonts, so $F_{250} \subset F_{500} \subset F_{1000}$; the full setting uses all fonts. Each subset retains all available training glyphs of its selected fonts without subsampling characters.

Each model is trained for one epoch with a global batch of 32 (eight GPUs, four samples per GPU), so the update budget scales with subset size. We use 3,000 warmup updates followed by cosine learning-rate decay from 5×10^{-5} to 10^{-6}, FP16 precision. Within a subset, initialization arms use the same data and optimization budget.

Table 8 reports validation, UFSC, and UFUC results for the nested subsets.

Figure 9 shows the corresponding training-loss trajectories for the three font subsets.

Figures 10–12 show qualitative comparisons for the same initialization study across the three evaluation splits.

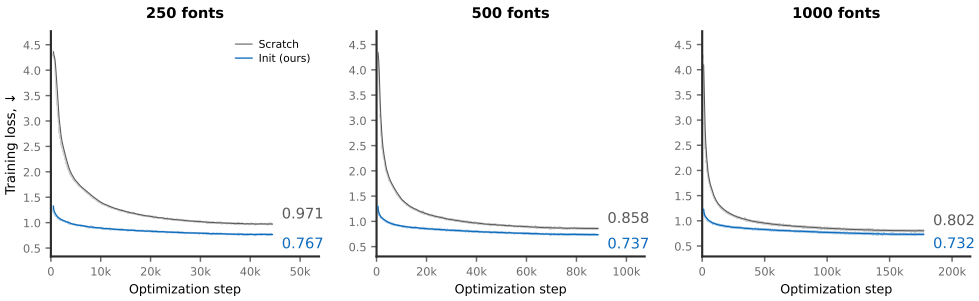


Figure 9: **Optimization trajectories with and without Raster-to-SVG initialization.** Training loss over one epoch on the 250-, 500-, and 1,000-font subsets. The horizontal ranges differ because each subset contains a different number of training samples; validation and test metrics are reported in Table 8.

Table 8: **Raster-to-SVG initialization: full results.** Metrics on validation, UFSC, and UFUC for each nested font subset, with and without Raster-to-SVG initialization. Sampling decoding (temp 0.8, top-k 50, top-p 0.95), CFG 1.5, 10k samples per cell. Green shading on FID cells marks the relative reduction in FID achieved by Raster-to-SVG initialization over random initialization; darker green indicates a larger reduction.

Fonts	Init	Validation					UFSC					UFUC				
		FID	LPIPS	L1	RMSE	SSIM	FID	LPIPS	L1	RMSE	SSIM	FID	LPIPS	L1	RMSE	SSIM
250	R2SVG	2.78	0.219	0.157	0.382	0.725	4.14	0.245	0.175	0.408	0.703	4.19	0.247	0.178	0.410	0.700
	random	8.74	0.251	0.192	0.428	0.688	8.74	0.264	0.203	0.440	0.677	9.05	0.266	0.204	0.442	0.675
500	R2SVG	1.91	0.184	0.131	0.347	0.754	3.60	0.229	0.163	0.391	0.717	3.63	0.231	0.164	0.393	0.715
	random	3.78	0.209	0.155	0.380	0.729	4.33	0.239	0.176	0.408	0.704	4.60	0.242	0.179	0.412	0.700
1000	R2SVG	1.87	0.180	0.127	0.341	0.759	3.08	0.215	0.151	0.376	0.730	3.10	0.217	0.153	0.378	0.728
	random	2.38	0.192	0.138	0.357	0.747	3.05	0.219	0.159	0.385	0.723	3.17	0.222	0.161	0.389	0.719
2,376 (full)	R2SVG	1.84	0.174	0.122	0.333	0.765	2.76	0.198	0.139	0.357	0.745	2.76	0.201	0.142	0.361	0.741
	random	1.86	0.181	0.128	0.342	0.758	2.43	0.199	0.141	0.361	0.742	2.46	0.203	0.145	0.366	0.738



Figure 10: **Raster-to-SVG initialization on validation examples.** For fixed structure and style references, we compare models trained from scratch and with Raster-to-SVG initialization on 250, 500, and 1,000 fonts against the target glyphs.



Figure 11: **Raster-to-SVG initialization on unseen fonts and seen characters (UFSC).** The structure and style references are shared within each comparison; rows show the three font-scale settings and ground truth.

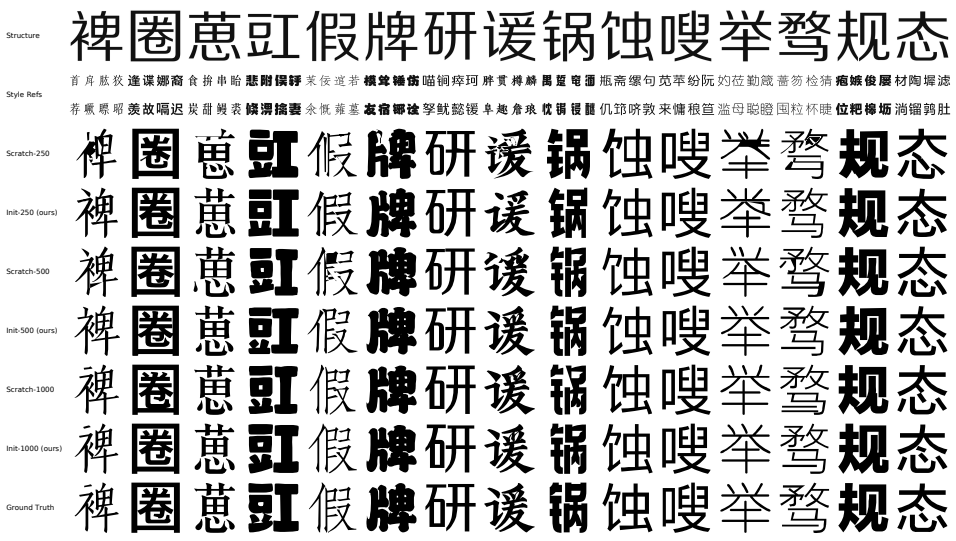


Figure 12: **Raster-to-SVG initialization on unseen fonts and unseen characters (UFUC).** The comparison uses matched references and targets across the three font-scale settings.

D.3 Decoding Reliability

Table 9 compares standard sampling with loop-aware rollback using the same checkpoint and decoding configuration. The available measurements cover loop rate, EOS rate, validity, and raster quality.

Table 9: **Effect of loop aware rollback.** Both variants use the same model checkpoint and initial sampling settings. We report residual loop rate, EOS termination, SVG validity, and raster quality.

Decoding Method	Decoding Reliability			Generation Quality				
	Loop Rate↓	EOS Rate↑	Validity↑	FID↓	LPIPS↓	L1↓	RMSE↓	SSIM↑
Standard Sampling	2.49%	99.86%	99.67%	2.759	0.201	0.142	0.361	0.741
Loop Aware Rollback	0.05%	99.95%	99.86%	2.682	0.202	0.143	0.362	0.740

E Additional Qualitative Results and Limitations

Figure 13 compares our outputs with the retrained raster baselines under matched inputs. Figures 14 and 15 present selected generations on unseen fonts, with seen (UFSC) and unseen (UFUC) target characters. Each example shows the Heiti structure input, style references from the target font, the ground-truth glyph, and our rendered SVG prediction. These examples illustrate both successful transfer and remaining errors without estimating their frequency.

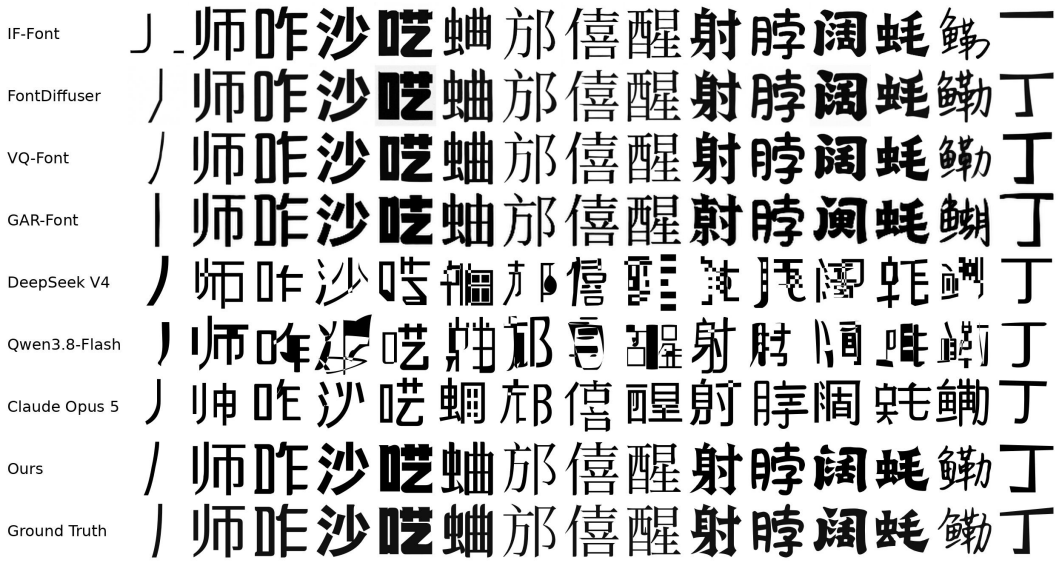
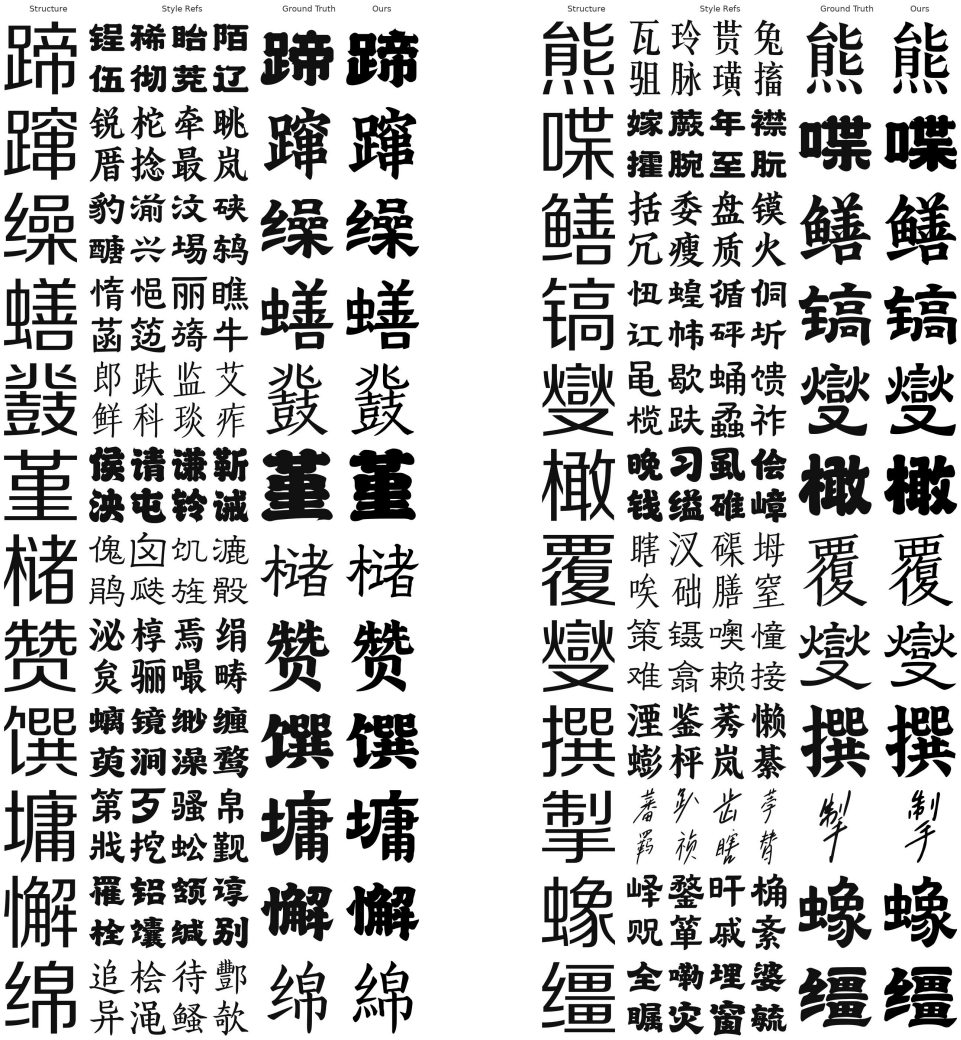


Figure 13: **Qualitative comparison with raster-domain methods on unseen fonts.** Each row shows matched structure and style references, target glyphs, and predictions. Our SVG outputs are rasterized for display using the same rendering settings.

Figure 15 illustrates representative cases with visible differences in character geometry or stroke appearance. The examples do not imply a measured prevalence of either error type.



(a) UFSC (b) UFUC

Figure 14: **Successful generations on unseen fonts.** UFSC and UFUC examples compare our rendered SVG predictions with ground truth under the shown structure input and style references.



(a) UFSC



(b) UFUC

Figure 15: **Representative failures on unseen fonts.** UFSC and UFUC examples show geometry or stroke-appearance deviations relative to the ground truth.