# VecGlypher: Unified Vector Glyph Generation with Language Modeling

**Xiaoke Huang**[1,2,*], **Bhavul Gauri**[1], **Kam Woh Ng**[1], **Tony Ng**[1], **Mengmeng Xu**[1], **Zhiheng Liu**[1], **Weiming Ren**[1], **Zhaochong An**[1], **Zijian Zhou**[1], **Haonan Qiu**[1], **Yuyin Zhou**[2], **Sen He**[1], **Ziheng Wang**[1], **Tao Xiang**[1], **Xiao Han**[1]

[1]Meta AI, [2]UC, Santa Cruz
*Work done at Meta

Vector glyphs are the atomic units of digital typography, yet most learning-based pipelines still depend on carefully curated exemplar sheets and raster-to-vector postprocessing, which limits accessibility and editability. We introduce VecGlypher, a single multimodal language model that generates high-fidelity vector glyphs directly from text descriptions or image exemplars. Given style prompts or reference glyph images, and a target character, VecGlypher autoregressively emits SVG path tokens, avoiding raster intermediates and producing editable, watertight outlines in one pass. A typography-aware data and training recipe makes this possible: (i) a large-scale continuation stage on 39K noisy Envato fonts to master SVG syntax and long-horizon geometry, followed by (ii) post-training on 2.5K expert-annotated Google Fonts with descriptive tags and exemplars to align language and imagery with geometry; preprocessing normalizes coordinate frames, canonicalizes paths, de-duplicates families, and quantizes coordinates for stable long-sequence decoding. On cross-family OOD evaluation, VecGlypher substantially outperforms both general-purpose LLMs and specialized vector-font baselines for text-only generation, while image-referenced generation reaches a state-of-the-art performance, with marked gains over DeepVecFont-v2 and DualVector. Ablations show that model scale and the two-stage recipe are critical and that absolute-coordinate serialization yields the best geometry. VecGlypher lowers the barrier to font creation by letting users design with words or exemplars, and provides a scalable foundation for future multimodal design tools.

**Date:** February 7, 2026
**Project Page:** https://xk-huang.github.io/VecGlypher

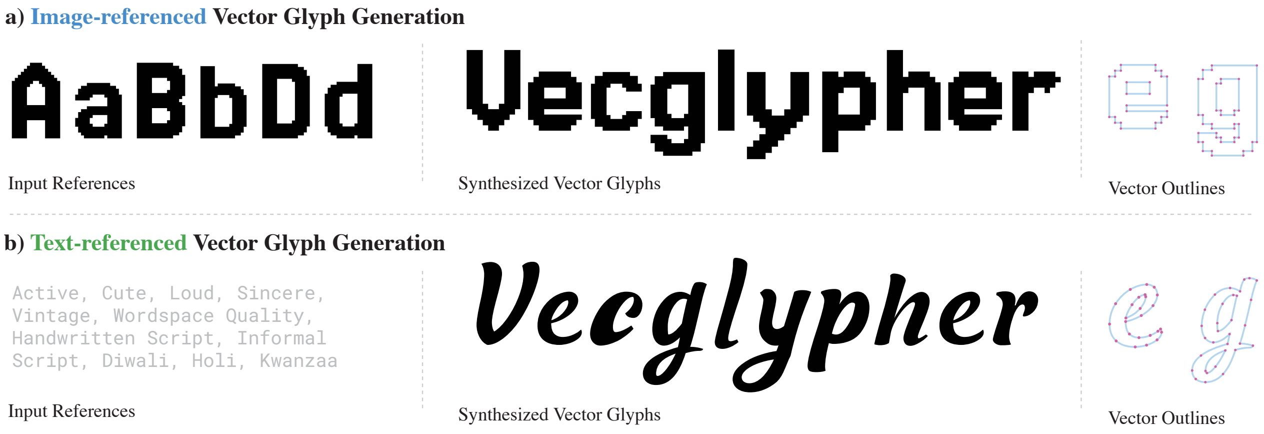

a) Image-referenced Vector Glyph Generation
AaBbDd
Input References
Vecglypher
Synthesized Vector Glyphs
Vector Outlines
b) Text-referenced Vector Glyph Generation
Active, Cute, Loud, Sincere, Vintage, Wordspace Quality, Handwritten Script, Informal Script, Diwali, Holi, Kwanzaa
Input References
Vecglypher
Synthesized Vector Glyphs
Vector Outlines

**Figure 1** **VecGlypher** generates high-fidelity vector glyphs directly as editable SVG outlines under two types of conditioning: (a) **image-referenced** generation, where a handful of exemplar glyph images specify the style and the model synthesizes new glyphs in the same visual form; and (b) **text-referenced** generation, where a natural-language prompt drives the synthesis without requiring exemplars. The figure shows the synthesized wordmark and sample vector outlines, highlighting one-pass generation of clean, controllable contours for typography workflows.

## 1 Introduction

Vector glyphs—the parametric curves that define letters and symbols in a font—are the atomic units of digital typography. They must be resolution-independent, compact, and precisely controllable for downstream typesetting, logo design, and UI rendering. Recent learning-based approaches have begun to automate glyph creation, yet the prevailing paradigm remains *image-referenced vector glyph generation*: given a few exemplar raster glyphs of a font, a model synthesizes the remaining characters by predicting vector outlines (Wang and Lian, 2021; Wang et al., 2023; Thamizharasan et al., 2024). While effective when exemplars are carefully prepared, this workflow assumes that users can already produce or collect representative glyph images for each new style. In practice, this requirement is a major bottleneck, especially for non-experts and for rapid ideation cycles in which designers would rather describe a desired style than draft a reference sheet.

We posit that natural language is a more universal and accessible interface for font creation. Designers and casual users routinely communicate typographic intent with concepts such as *"high-contrast, narrow, slightly condensed, art-deco, playful"*. Moreover, the output we ultimately seek, a sequence of vector drawing commands and coordinates, is itself textual (e.g., an SVG path). This observation suggests an appealing formulation: treat glyph generation as a **language modeling** problem and leverage multimodal **Large Language Models** (LLMs) (Bai et al., 2025; Team et al., 2025; Liu et al., 2023a) to map text descriptions or image exemplars directly to vector code. Besides simplifying the human interface, this formulation stands to inherit the strong text and image understanding of modern foundation models.

However, building an LLM that can actually *draw* credible typography is nontrivial. General-purpose LLMs (Achiam et al., 2023; Anthropic, 2024; Comanici et al., 2025; Yang et al., 2025a) and off-the-shelf vector-graphics LLMs (Rodriguez et al., 2023; Xing et al., 2025; Yang et al., 2025b) that produce icons or simple SVG drawings typically fail on glyphs. Typography imposes strict geometric and stylistic constraints: long sequences of precise coordinates; watertight topology; consistent stroke logic across characters; and subtle style factors (contrast, terminals, serifs, curvature) that must vary coherently with the target content. Beyond model capacity, practical data issues arise: paired text-glyph examples are scarce (Google LLC, 2025); tags in large repositories are noisy and often non-visual (Envato Pty Ltd, 2025); and coordinate systems vary across sources. Naively prompting existing LLMs to "write an SVG path for a serif V" often yields broken geometry, mismatched case, or degenerate paths.

**VecGlypher** addresses these challenges and unifies *text- and image-referenced* vector glyph generation within a single language model (Fig. 2). Our model is a multimodal decoder that consumes (i) a style description or image exemplars, and (ii) the target character identity, then autoregressively predicts SVG path tokens for the output glyph. The same architecture and decoding procedure handle both input modalities. At inference, generated tokens are de-tokenized into a valid SVG path and rasterized for preview.

**Table 1 SVG draw commands used in this work.** We serialize each glyph as a single SVG path using only *MoveTo*, *LineTo*, *Quadratic Bézier*, and *ClosePath*. Uppercase commands use absolute coordinates; lowercase are relative to the current point. Coordinates are quantized to one decimal; other attributes (e.g., "fill", "stroke", "transform") are unused.

| Command | Arguments | Description |
|---|---|---|
| `M / m` (MoveTo) | $x, y$ | Move the cursor to the end-point $(x, y)$ without drawing anything. |
| `L / l` (LineTo) | $x, y$ | Draw a line to the end-point $(x, y)$. |
| `Q / q` (Quadratic Bézier) | $q_x, q_y$ $x, y$ | Draw a quadratic Bézier curve with the control point $(q_x, q_y)$ and the end-point $(x, y)$. |
| `Z / z` (ClosePath) | $\varnothing$ | Close the path by moving the cursor back to the starting position (ignorable). |

Achieving reliable glyph drawing with an LLM requires both the right training recipe and typography-aware data engineering. We curate complementary corpora and adopt a two-stage scheme (Fig. 3): **Stage 1** performs large-scale continuation on noisy text-glyph pairs from a 39K-font Envato collection (Envato Pty Ltd, 2025) to *teach the model to draw*—establishing robust SVG syntax, long-horizon coordinate prediction, and character-conditioned geometry. **Stage 2** post-trains on a smaller but higher-quality 2.5K-font Google Fonts set (Google LLC, 2025) with expert descriptors and image exemplars, sharpening the mapping from textual concepts or image exemplers to glyph geometry and enabling *unified* generation for both

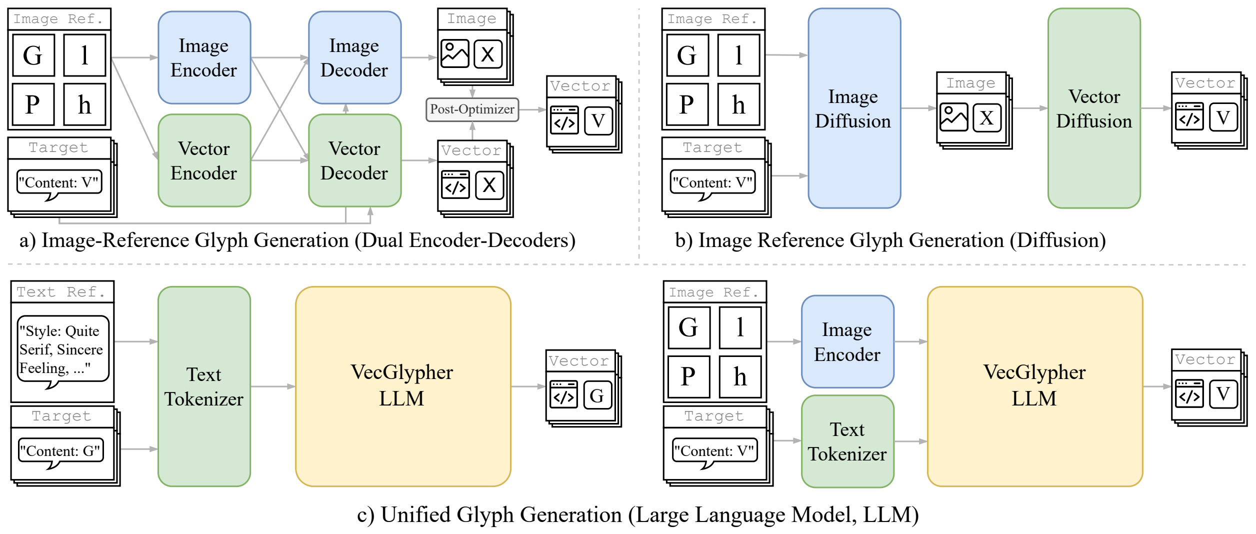


Image Ref.
G
l
P
h
Target
"Content: V"
Image Encoder
Vector Encoder
Image Decoder
Vector Decoder
Image
Vector
Post-Optimizer
a) Image-Reference Glyph Generation (Dual Encoder-Decoders)
Image Diffusion
Vector Diffusion
b) Image Reference Glyph Generation (Diffusion)
Text Ref.
"Style: Quite Serif, Sincere Feeling, ..."
"Content: G"
Text Tokenizer
VecGlypher LLM
c) Unified Glyph Generation (Large Language Model, LLM)

**Figure 2 Paradigm comparisons.** a) Prior image-referenced pipelines use separate image and vector encoder–decoders and a geometry post-optimizer. b) Diffusion-based approaches cascade image diffusion with a vector decoder. c) **VecGlypher** unifies both text- and image-referenced conditioning within a single LLM: given a style description or reference glyph images plus a target character, the model autoregressively emits SVG path tokens that detokenize to a valid SVG path. This formulation removes raster intermediates and exemplar-sheet requirements while producing directly editable vectors. A *practical workflow* is to first generate a few reference glyphs from text descriptions, then bootstrap with those images to synthesize the full font.

modalities.

To support stable long-sequence decoding, we design a typography-specific preprocessing pipeline: deduplicate near-identical fonts; remove malformed or excessively long paths; normalize all glyphs to a consistent coordinate system; serialize each glyph to a canonical SVG `<path>` representation; and tokenize commands and coordinates with fixed-precision quantization. These steps lower sequence complexity and reduce error propagation during decoding, turning a general-purpose multimodal LLM into a competent glyph drawer.

Fig. 2 summarizes the contrast with prior paradigms and our resulting pipeline. Compared with dual encoder-decoder systems tailored to image-referenced generation or cascaded image-then-vector diffusion approaches, **VecGlypher** collapses glyph synthesis into a single autoregressive program that emits vector tokens directly. This yields three practical advantages: (1) no exemplar sheet is required—text alone suffices—while still supporting exemplars when available; (2) avoiding raster intermediates eliminates vectorization artifacts by generating valid SVG in one pass; and (3) the language-modeling formulation scales naturally—larger backbones, more continuation data, and improved tokenization translate into stronger glyph geometry and generalization.

We evaluate these claims extensively. Both qualitatively and quantitatively, **VecGlypher** produces cleaner, more stylistically faithful glyphs than specialized *image-referenced* baselines, and succeeds on *text-referenced* tasks where state-of-the-art LLMs and vector-graphics LLMs fail to draw at all. Ablations show that (i) model scale is a primary driver of vector fidelity and style consistency, and (ii) large-scale continuation in Stage 1 delivers tangible out-of-distribution gains beyond what limited expert-annotated data alone can offer.

Our contributions are summarized as follows:

• **Unified formulation.** We recast vector glyph generation as language modeling over SVG path tokens and present **VecGlypher**, a single multimodal LLM that supports both text- and image-referenced conditioning while directly emitting vector code.

• **Two-stage training recipe.** We curate two complementary datasets (39K noisy fonts; 2.5K expert-annotated fonts) and show that large-scale continuation followed by targeted post-training is key to drawing high-quality glyphs from either modality.

**Table 2 Font filtering.**

| Filtering | Google Fonts | Envato Fonts |
|---|---|---|
| Total fonts | 3,403 | 82,343 |
| Invalid / Lengthy | 2,989 | 70,248 |
| Duplicated | 2,645 | 65,216 |
| MLLM OCR | 2,497 | 39,497 |

**Table 3 Font-level dataset statistics.**

| Split | Google Fonts | | Envato Fonts | |
|---|---|---|---|---|
| | Font Family | Font | Font Family | Font |
| Total | 1,117 | 2,497 | 23,543 | 39,497 |
| Train | 997 | 2,243 | 22,543 | 37,926 |
| Test | 120 | 254 | 1,000 | 1,571 |

**Table 4 Glyph-level dataset statistics.**

| Split | Google Fonts | | Envato Fonts | |
|---|---|---|---|---|
| | Original | Filtering | Original | Filtering |
| Total | 159,808 | 157,899 | 2,527,718 | 2,495,363 |
| Train | 143,552 | 142,148 | 2,427,174 | 2,394,819 |
| Test | 16,256 | 15,751 | 100,544 | 100,544 |

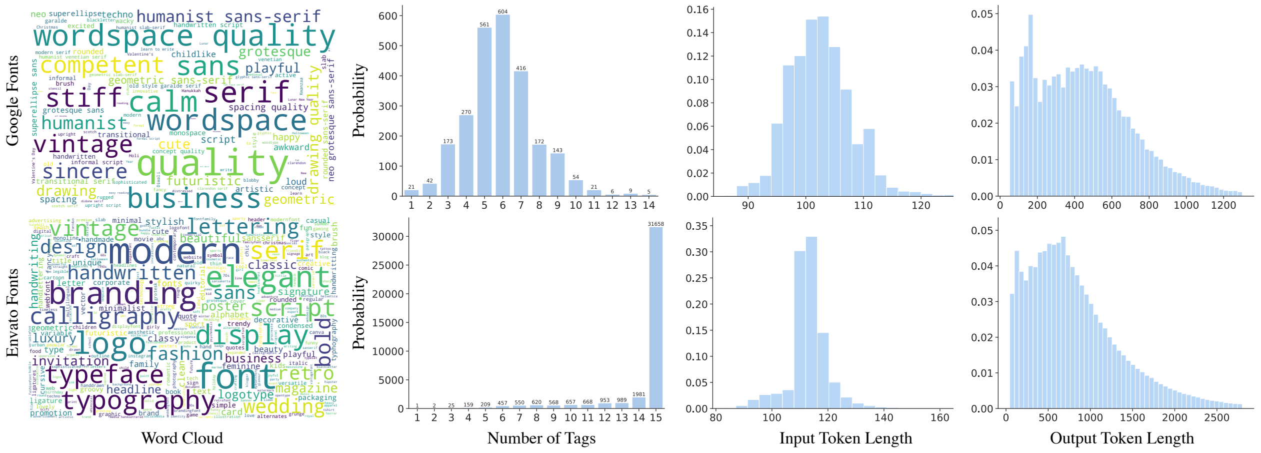


**Table 5 Dataset statistics.** Word clouds (left) visualize tag vocabularies for *Google Fonts* (expert-curated, appearance-focused) versus *Envato* (noisier, marketing-oriented). The middle plots show the per-font number of tags (Google: concise; Envato: capped at <15). Right: distributions of input token length (tags) and output token length (SVG path tokens), with Envato exhibiting heavier-tailed outputs. These differences motivate our two-stage recipe: large-scale continuation on Envato, then instruction alignment on Google Fonts.

• **Empirical validation.** Through comprehensive evaluations, we demonstrate substantial improvements over specialized image-referenced methods and show that general SVG-generation LLMs are insufficient for typography, whereas our approach follows textual descriptions and exemplar images with high fidelity.

Together, these results indicate that language models, properly trained and paired with typography-aware data engineering, can serve as a unified engine for vector glyph generation. **VecGlypher** lowers the barrier to font creation, lets users design with words or images, and provides a scalable foundation for future multimodal design tools.

## 2 Related Works

*Image-referenced glyph and font generation.* Early systems synthesize glyph bitmaps and transfer style across characters from a few exemplar images (Wang et al., 2020; Reddy et al., 2021; Li and Lian, 2024; Yang et al., 2023; He et al., 2024; Hayashi et al., 2019). While visually compelling, raster pipelines remain decoupled from vectorization, limiting editability and resolution-independent reuse. Vector-native methods instead predict parametric curves directly. (Lopes et al., 2019) sparked interest in structured, editable vector outputs; subsequent work predicts vector primitives from reference images to reconstruct style-consistent glyphs (Wang and Lian, 2021), improves contour fidelity and content-style disentanglement (Wang et al., 2023), jointly models curve geometry and filled regions (Liu et al., 2023b), represents glyphs with signed distance fields (Xia et al., 2023), and couples diffusion priors with a vector decoder for sparse-exemplar generation (Thamizharasan et al., 2024). These specialized architectures are typically trained on moderate-scale corpora (e.g., (Wang and Lian, 2021; Wang et al., 2023) ≈8K fonts; (Thamizharasan et al., 2024) ≈1.4K). VecGlypher differs by combining substantially larger supervision (39K+2.5K fonts) with a language-model backbone to unify text or image exemplar-conditioned synthesis and vector decoding, yielding stronger cross-script generalization and directly editable glyph outputs.

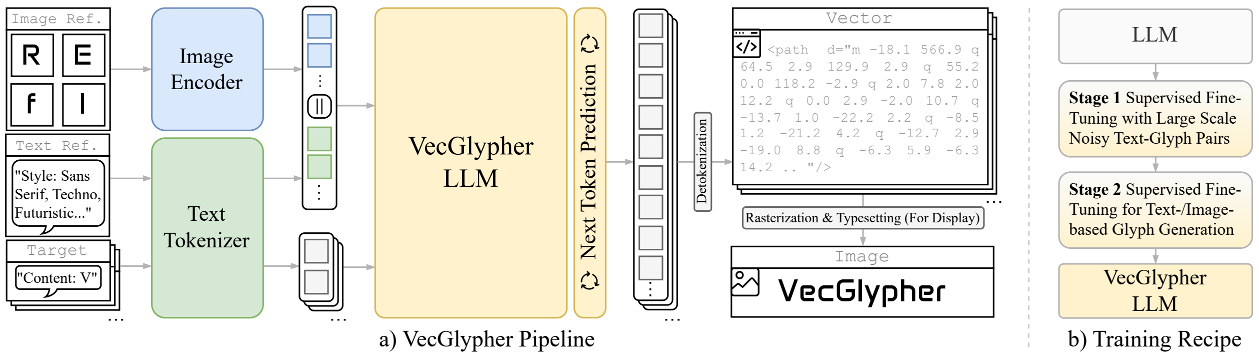


Image Ref.
R E f I
Text Ref.
"Style: Sans Serif, Techno, Futuristic..."
Target
"Content: V"
Image Encoder
Text Tokenizer
VecGlypher LLM
Next Token Prediction
Detokenization
Vector
Rasterization & Typesetting (For Display)
Image
VecGlypher
a) VecGlypher Pipeline
LLM
Stage 1 Supervised Fine-Tuning with Large Scale Noisy Text-Glyph Pairs
Stage 2 Supervised Fine-Tuning for Text-/Image-based Glyph Generation
VecGlypher LLM
b) Training Recipe

```
<path  d="m -18.1 566.9 q
64.5  2.9  129.9  2.9  q  55.2
0.0 118.2 -2.9 q 2.0 7.8 2.0
12.2  q  0.0  2.9  -2.0  10.7  q
-13.7  1.0  -22.2  2.2  q  -8.5
1.2  -21.2  4.2  q  -12.7  2.9
-19.0  8.8  q  -6.3  5.9  -6.3
14.2 .. "/>
```

**Figure 3 VecGlypher pipeline and training recipe.** a) A text tokenizer or image encoder condition the LLM (|| denotes mutually exclusive choice), which predicts the next SVG token until the path is produced; detokenization yields SVG paths that we rasterize for display only. b) Training is two-stage: 1) SFT on Envato (text-referenced only) teaches SVG syntax and long-horizon geometry; 2) SFT on Google Fonts (text- or image-referenced) aligns geometry to appearance instructions. No raster denoisers or post-optimizers are used.

*Text-referenced glyph and font generation.* A complementary line conditions font synthesis on language. Early datasets enabled modeling from style tags and textual attributes (Chen et al., 2019; O'Donovan et al., 2014). Systems generate fonts from user-specified "impressions" (Matsuda et al., 2021; Kang et al., 2022) or learn under incomplete tags (Matsuda et al., 2022). More recent work moves toward natural language, accepting free-form descriptions (Wang et al., 2024), guiding diffusion with text descriptors (Kang et al., 2024), and supporting multimodal, interactive optimization for iterative design (Tatsukawa et al., 2025). Most of these operate in the raster domain or emphasize retrieval or editing rather than producing structured, vector-native fonts with family-level consistency. In contrast, VecGlypher supports *both* text and exemplar conditioning in a single model and emits vector glyphs directly, linking language-level intent to geometry without a separate vectorization stage.

*Language models for vector graphics.* Large language models demonstrate strong compositional reasoning, code generation, and multimodal instruction following (Radford et al., 2018, 2019; Brown et al., 2020; Achiam et al., 2023; Comanici et al., 2025; Anthropic, 2024; Jaech et al., 2024). Text-to-vector approaches optimize or generate SVG-like structures from prompts, often via differentiable rendering and discrete primitives (Jain et al., 2022; Li et al., 2020; Xu et al., 2022; Xing et al., 2023; Wu et al., 2024). Vector-graphics LLMs go further by writing or editing SVG programs for icons and simple illustrations (Rodriguez et al., 2023; Wu et al., 2023; Xing et al., 2025; Yang et al., 2025b). Despite promising results, these efforts generally target generic shapes, rely on small curated corpora, and do not address family-level consistency, Unicode coverage, or the typographic constraints unique to fonts. To our knowledge, VecGlypher is the *first* unified LLM-based framework tailored to glyphs: it adapts an multimodal LLM to directly produce high-fidelity, editable vector glyphs and supports both text and image references within one model. At the 39K+2.5K data scale, this formulation closes the gap between language-conditioned intent and professional, vector-native type design.

## 3 Method

VecGlypher learns to autoregressively emit the SVG `<path>` string of a target glyph from either a textual style description or a set of reference glyph images. Prior pipelines (i) translate images to vectors with separate encoder–decoders and a geometry post-optimizer (Wang and Lian, 2021; Wang et al., 2023; Liu et al., 2023b), or (ii) cascade image and vector diffusions (Thamizharasan et al., 2024). In contrast, VecGlypher *unifies* style understanding and outline drawing within a single LLM conditioned on text or images. This removes intermediate mismatches and yields a *single training objective*: **next-token prediction on SVG**.

### 3.1 Preliminaries

A *glyph* is the vector outline of one character. We generate one glyph at a time as a single SVG `<path>`. SVG represents a path as a sequence of drawing commands with arguments; uppercase commands use absolute coordinates and lowercase commands use relative coordinates. Tab. 1 summarizes the commands we allow; no other SVG attributes (e.g., fill, stroke) are used. During training and inference, each glyph is serialized as a *string* of SVG drawing commands with *one-decimal* precision.

### 3.2 Data curation

*Envato fonts.* A large collection with up to fifteen *noisy retrieval tags* per font (often names, authors, or marketing phrases). We use this corpus to *teach* the model SVG syntax and geometry for glyph drawing.

*Google Fonts.* A smaller but *expert-tagged* set with appearance descriptors (e.g., humanist sans-serif, monoline, x-height). We use it for *instruction following*, i.e., learning to follow style tags and to imitate styles from image references.

*Dataset statistics.* According to Tab. 5, Google Fonts provides concise, informative tags and moderate output lengths; Envato exhibits a hard cap on tag counts and longer, heavy-tailed output sequences.

### 3.3 Dataset processing

*Font processing.* We apply four filters to increase the quality of the training data:

1. **Character coverage:** remove fonts lacking alphanumerics ("0–9, a–z, A–Z") such as symbols or pictograms.
2. **Length by pangram:** render the pangram "*The quick brown fox jumps over the lazy dog*" as SVG; discard fonts whose total path length exceeds the 0.9 quantile.
3. **Deduplication:** drop fonts with identical pangram renderings.
4. **Multimodal LLM sanity check:** render "GgAa" with a state-of-the-art open-weight multimodal LLM (Qwen3-VL-30A3B-Instruct) (Yang et al., 2025a) and remove unicase fonts (small caps or oversized lowercase) and failed or ambiguous renders. Tab. 2 reports resulting sizes.

We split *by font family* to measure cross-family generalization. In training we exclude extremely long glyphs ($>$0.9 quantile of tokenized path length). In testing we remove glyphs whose outlines duplicate those in training. Tab. 3 summarizes the splits.

*Glyph processing.* We extract each glyph as an SVG `<path>` and standardize as follows:

1. **Representation:** keep both absolute and relative commands; never elide consecutive command letters.
2. **Normalization:** rescale to UPM=1000 and vertically align to a shared baseline.
3. **Parsing:** retain only the `d=""` attribute of `<path>`; drop all other attributes (e.g., "fill", "stroke").
4. **Quantization:** round all coordinates to one decimal.

Tab. 4 gives glyph-level counts.

### 3.4 Problem formulation

*Training and inference.* Let $x^{\mathrm{t}}$ be the tokenized style description, $x^{\mathrm{i}} = (x^{\mathrm{i}}_1, \ldots, x^{\mathrm{i}}_N)$ a set of $N$ reference glyph images, and $x^{\mathrm{c}}$ the target character in "0–9a–zA–Z". The glyph is an SVG token sequence $y = (y_1, \ldots, y_T)$ whose concatenation forms a valid `d` string. VecGlypher (parameters $\theta$) models

$$p_\theta\left(y \mid (x^{\mathrm{t}} \parallel x^{\mathrm{i}}), x^{\mathrm{c}}\right) = \prod_{t=1}^{T} p_\theta\left(y_t \;\middle|\; y_{<t}, (x^{\mathrm{t}} \parallel x^{\mathrm{i}}), x^{\mathrm{c}}\right),$$

where $(x^{\mathrm{t}} \parallel x^{\mathrm{i}})$ denotes text- or image-referenced conditioning. We minimize next-token cross-entropy on the *SVG path text*. Consistent with (Rodriguez et al., 2023), specialized SVG tokenization (Xing et al., 2025; Yang et al., 2025b) adds complexity without measurable gains. At inference we use conservative decoding to favor syntactically valid SVG.

*Output space.* The model outputs the exact text of a single SVG `<path>`. Because subword tokenization naturally covers command letters, separators, and numeric substrings, we simply detokenize to obtain the final `d` string and render if needed (Fig. 3a). We do *not* apply raster denoisers, vector post-optimizers, or outline simplifiers (Wang and Lian, 2021; Wang et al., 2023; Liu et al., 2023b).

*Prompts.* A strict system prompt enforces output structure and removes non-SVG tokens. For text-referenced generation, the input is a bag of style tags plus the target character; we randomly permute tag order during training to reduce positional bias. For image-referenced generation, we provide 1–8 reference glyph images from the same font, rendered at 192×192 with centering and uniform padding. The model transfers style and metrics from references to the requested content.

### 3.5 Training recipe

Our training uses two-stage supervised fine-tuning (Fig. 3b).

**Stage 1: learning to draw (Envato).** We perform supervised fine-tuning (SFT) on Envato (Envato Pty Ltd, 2025) using *text-referenced* samples only. Despite noisy tags, the scale and diversity force the model to master SVG glyph syntax and long-horizon geometry. Randomly sampled glyph-tag pairs are used as inputs; targets are gold `<path>` strings. This stage markedly improves closure, winding consistency, and control-point placement.

**Stage 2: instruction following (Google Fonts).** We then SFT on Google Fonts (Google LLC, 2025) with a *mixture* of text- and image-referenced samples to align geometry with appearance instructions. Empirically, Stage 1 is essential: models initialized with it generalize better to unseen families and produce more stable outlines.

## 4 Experiments

### 4.1 Implementation Details

We fine-tune with AdamW at a base learning rate (LR) of $1 \times 10^{-5}$, weight decay of 0.01, and scale the LR by the square root of the GPU scaling factor. A cosine schedule with 1% warm-up decays the LR to zero. Envato is trained for one epoch and Google Fonts for three. For the 4B setting we use 8×A100 with a global batch of 32; for 27B and 70B we use 32×A100 with a global batch of 128. Decoding uses greedy sampling to assess raw generation capability. Input and conditioning details are in section 3.4.

### 4.2 Datasets and Evaluation

*Corpora and splits.* We evaluate on Google Fonts, where expert-curated tags closely match glyph appearance, and use Envato as a large but noisy pre-training source. To test out-of-distribution (OOD) generalization, Google Fonts is split by *family* so that styles from the same family never appear in both train and test. Additional details are in section 3.2.

*Evaluation protocol.* Unless noted, we report results on Google Fonts *test families* only (cross-family OOD). Metrics:

1. **Relative OCR Accuracy (R-ACC).** We run Qwen3-VL-30A3B-Instruct (Yang et al., 2025a) as the OCR engine, prompting abstention for unrecognizable glyphs (Chang et al., 2025; Geng et al., 2025). Accuracy on generated glyphs is normalized by *OCR accuracy on the ground truths* and scaled by 100; scores can exceed 100 due to OCR variability (higher is better).

2. **Chamfer Distance (CD)** (Borgefors, 1986, 2002). We normalize both glyphs to $[-1, 1]$ (preserving aspect ratio), uniformly sample 200 points per glyph, and compute the symmetric Chamfer distance, scaled by 100 (lower is better).
3. **CLIP** (Radford et al., 2021). Text–image similarity (ViT-B/32) between the prompt tags and a rasterized preview (higher is better).
4. **DINO** (Oquab et al., 2023). Cosine *similarity* of DINOv2 ViT-B/14 features between the generated and ground-truth glyphs, scaled by 100 (higher is better).
5. **FID** (Heusel et al., 2017). Fréchet Inception Distance between generated and target glyph distributions (lower is better).

## 4.3 Ablations

*Text-referenced.* We train Gemma3 (Team et al., 2025) 4B and 27B on Google Fonts and study data sources at 27B; see Tab. 6.

• **Model size.** Scaling from 4B to 27B markedly improves geometry and fidelity: on Google-only training, R-ACC rises from 73.96/66.66 (relative/absolute coordinates) to 92.81/94.91, while CD drops from 4.29/3.75 to 2.31/1.98 and FID from 15.57/14.69 to 5.81/3.96. CLIP and DINO similarly improve.

• **Data source.** Using Envato only at 27B yields strong recognition (R-ACC ≈94) but weak geometry/fidelity, confirming that noisy supervision alone does not teach precise contours. Mixing Envato and Google in a single stage improves recognition yet remains inferior in geometry/fidelity to Google-only training. A two-stage recipe consistently performs best: with absolute coordinates we obtain R-ACC 101.0, CD 1.67, DINO 94.34, FID 3.47.

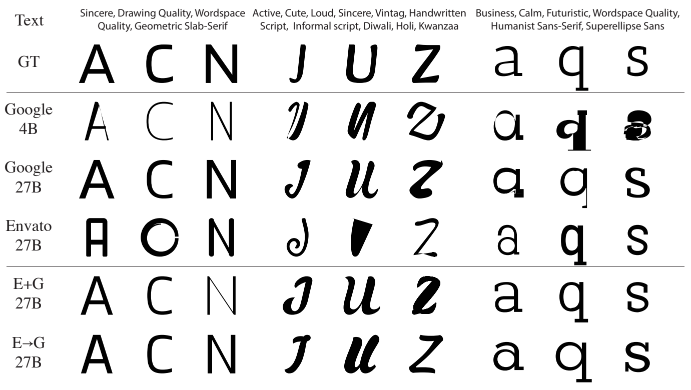

Text
Sincere, Drawing Quality, Wordspace Quality, Geometric Slab-Serif
Active, Cute, Loud, Sincere, Vintag, Handwritten Script, Informal script, Diwali, Holi, Kwanzaa
Business, Calm, Futuristic, Wordspace Quality, Humanist Sans-Serif, Superellipse Sans
GT
Google 4B
Google 27B
Envato 27B
E+G 27B
E→G 27B

**Figure 4 Text-referenced ablations.** Representative text-to-glyph generations across model sizes and data regimens: ground truth (GT), Google-only 4B and 27B models, Envato-only 27B, mixed E+G 27B, and two-stage E→G 27B. Using the same style tags across columns, scaling and the two-stage recipe yield cleaner closures, stable counters, and more faithful style. Please refer to the supplementary materials for comprehensive results.

• **SVG representation.** At 27B, absolute coordinates slightly but consistently outperform relative across metrics. At 4B, relative coordinates can yield marginally higher R-ACC but worse geometry, indicating absolute coordinates become preferable as capacity increases. We provide qualitative results in Fig. 4.

*Image-referenced.* We repeat the analysis for the image-referenced task; see Tab. 7.

• **Model size.** With Google-only training, 27B improves over 4B on all metrics (e.g., CD 1.41 vs. 2.11, FID 2.84 vs. 7.94 with absolute coordinates).

• **Data source.** Pre-training on *text-referenced* Envato and then fine-tuning on *image-referenced* Google improves geometry (CD 1.16–1.17) and fidelity (FID 2.51–2.55) versus Google-only. A joint second stage mixing text- and image-referenced Google is best: with absolute coordinates we reach R-ACC 99.12, CD 1.18, CLIP 26.07, DINO 95.82, and FID 2.32.

• **SVG representation.** As above, absolute coordinates are equal or better for 27B; 4B sometimes gains slightly higher R-ACC with relative coordinates but with worse geometry. The qualitative comparisons can be found in Fig. 5.

## 4.4 Comparisons to Baselines

**Table 6 Text-referenced ablations on data and model size.** We compare relative (R) vs. absolute (A) coordinate serialization and several data regimens: Google-only (G), Envato-only (E), mixed E+G, and two-stage E→G. Scaling and the two-stage E→G training with absolute coordinates yield the best geometry and fidelity.

| Data | Size | Repr. | R-ACC ↑ | CD ↓ | CLIP ↑ | DINO ↑ | FID ↓ |
|---|---|---|---|---|---|---|---|
| Google | 4B | R | 73.96 | 4.29 | 26.02 | 90.27 | 15.57 |
| | | A | 66.66 | 3.75 | 26.04 | 89.92 | 14.69 |
| | 27B | R | 92.81 | 2.31 | 26.38 | 93.01 | 5.81 |
| | | A | 94.91 | 1.98 | 26.43 | 93.43 | 3.96 |
| Envato | 27B | R | 93.99 | 3.89 | 26.43 | 89.79 | 30.68 |
| | | A | 93.84 | 3.63 | 26.45 | 90.24 | 20.43 |
| E + G | 27B | R | 94.39 | 2.56 | 26.39 | 93.17 | 5.83 |
| | | A | 95.59 | 2.14 | 26.45 | 93.66 | 4.86 |
| E → G | 27B | R | 99.88 | 1.71 | 26.52 | 94.07 | 3.57 |
| | | A | 101.0 | 1.67 | 26.53 | 94.34 | 3.47 |

**Table 7 Image-referenced ablations on data and model size.** With 1—8 reference glyphs, we study Google-only 4B/27B and two-stage variants that first SFT on Envato then fine-tune on Google ("E→G I" for image-only training; "E→G T,I" for mixed training). At 27B, absolute coordinates achieve the strongest overall results, outperforming single-stage training.

| Data | Size | Repr. | R-ACC ↑ | CD ↓ | CLIP ↑ | DINO ↑ | FID ↓ |
|---|---|---|---|---|---|---|---|
| Google | 4B | R | 83.48 | 2.36 | 25.90 | 93.26 | 9.38 |
| | | A | 81.94 | 2.11 | 25.89 | 93.11 | 7.94 |
| | 27B | R | 94.42 | 1.53 | 26.03 | 94.88 | 4.07 |
| | | A | 96.46 | 1.41 | 26.04 | 95.15 | 2.84 |
| E→G I | 27B | R | 98.90 | 1.17 | 26.06 | 95.69 | 2.51 |
| | | A | 97.88 | 1.16 | 26.06 | 95.84 | 2.55 |
| E→G T,I | 27B | R | 99.08 | 1.21 | 26.07 | 95.65 | 2.48 |
| | | A | 99.12 | 1.18 | 26.07 | 95.82 | 2.32 |

*Text-referenced.* We compare to general-purpose proprietary LLMs and our VecGlypher models (Tab. 8). Budget-tier models (e.g., GPT-5 Mini, Gemini-2.5 Flash) rarely output valid SVGs. Flagship models perform better but still struggle to follow the requested character and to close contours. VecGlypher substantially outperforms all baselines: VecGlypher-27B (Gemma3-27B) attains R-ACC 100.5, CD 1.72, DINO 94.22, and FID 3.46; VecGlypher-70B (Llama3.3-70B (Grattafiori et al., 2024)) further improves CD/FID to 1.68/3.34 with R-ACC 100.4. Relative to the strongest baseline (Claude Sonnet 4.5), VecGlypher-70B delivers ∼2.15× higher R-ACC, 68% lower CD, and 83% lower FID, while improving DINO by 6 points. These results highlight the current limitations of general LLMs for glyph generation and the effectiveness of our unified, two-stage training. Fig. 6 illustrates the qualitative comparison against the baselines.

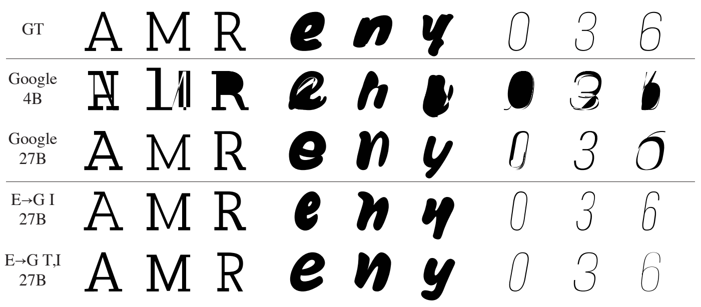


**Figure 5 Image-referenced ablations.** Given 1–8 reference glyphs from a font, we compare Google-only 4B/27B with two-stage variants (E→G I and E→G T,I). The two-stage settings at 27B best transfer style and preserve thin structures and closures; Google-only baselines underperform in geometry. Please refer to the supplementary materials for comprehensive results.

*Image-referenced.* We compare with dedicated vector-glyph methods DeepVecFont-v2 (Wang et al., 2023) and DualVector (Liu et al., 2023b) (Tab. 9). These approaches generalize poorly to unseen structures (e.g., thin strokes ). In contrast, VecGlypher-27B achieves R-ACC 99.12, CD 1.18, and FID 2.32, improving over

**Table 8 Text-referenced comparisons with general LLMs.** We evaluate several proprietary/budget LLMs against VecGlypher on cross-family OOD fonts. Budget models rarely emit valid SVG; flagship models improve but still trail our approach. VecGlypher-70B (A) attains state-of-the-art performance.

| Model | R-ACC↑ | CD↓ | CLIP↑ | DINO↑ | FID↓ |
|---|---|---|---|---|---|
| GPT-5 Mini | 4.17 | 10.70 | 24.75 | 79.65 | 63.86 |
| Gemini-2.5 Flash | 4.89 | 13.78 | 25.26 | 78.62 | 86.03 |
| Claude Haiku 4.5 | 32.38 | 7.60 | 25.61 | 84.69 | 35.07 |
| GPT-5 | 43.98 | 6.12 | 25.95 | 86.92 | 29.00 |
| Gemini 2.5 Pro | 24.04 | 8.22 | 25.57 | 83.77 | 47.82 |
| Claude Sonnet 4.5 | 46.65 | 5.28 | 25.99 | 88.31 | 19.59 |
| VecGlypher 70B T,R | 100.1 | 1.70 | 26.53 | 94.10 | 3.45 |
| VecGlypher 70B T,A | 100.4 | 1.68 | 26.54 | 94.28 | 3.34 |
| VecGlypher 27B T,I,R | 99.62 | 1.77 | 26.53 | 93.97 | 3.50 |
| VecGlypher 27B T,I,A | 100.5 | 1.72 | 26.53 | 94.22 | 3.46 |

**Table 9 Image-referenced comparisons with vector-font baselines.** Against DeepVecFont-v2 (Wang et al., 2023) and DualVector (Liu et al., 2023b), VecGlypher-27B (T,I) delivers substantially higher recognizability and drastically lower geometric and distributional errors, demonstrating strong generalization to unseen styles.

| Model | R-ACC↑ | CD↓ | CLIP↑ | DINO↑ | FID↓ |
|---|---|---|---|---|---|
| DeepVecFont-v2 | 37.86 | 14.58 | 24.81 | 79.41 | 115.5 |
| DualVector | 49.20 | 16.45 | 25.07 | 79.57 | 105.5 |
| VecGlypher 27B T,I,R | 99.08 | 1.21 | 26.07 | 95.65 | 2.48 |
| VecGlypher 27B T,I,A | 99.12 | 1.18 | 26.07 | 95.82 | 2.32 |

the best baseline by $\sim 2\times$ R-ACC, 92% lower CD, and 97.8% lower FID. Qualitative results show our outputs close contours, preserve thin structures, and follow target content faithfully. Fig. 7 illustrates the qualitative comparison against the baselines.

*More qualitative results.* We include additional full qualitative samples across the full test set. Please refer to our supplementary material for thorough examination.

## 5 Discussions and Conclusions

*Discussions.* VecGlypher exposes a sharp domain gap: LLMs that can write SVG icons rarely produce typographically valid glyphs. We posit this stems from training corpora that lack glyph programs, encouraging memorization of pictorial contours rather than learning vector-structural rules. Consequently, unified vector glyph generation is foremost a data problem: if next-generation LLMs ingest large, diverse corpora of vector glyph programs annotated with typographic constraints (closure, winding, counters, etc.), the capability should emerge with instruction tuning.

Scale still matters. Small models, even cost-efficient proprietary LLMs (see Tab. 8), remain brittle; current evidence points to $\sim$30B parameters for stable quality. That operating point could drop with improved vector tokenization, constrained decoding, and lightweight geometric adapters once glyph-centric data is available. Progress also depends on measurement: composite proxy rewards that mix vector/raster geometry, topology, typographic regularity, and recognizability (Tatsukawa et al., 2024) enable best-of-N sampling (Wang et al., 2022) and RL fine-tuning (Guo et al., 2025) from text or image cues.

Our current scope ("0–9, a–z, A–Z") highlights the limits of closed-set vector methods (Wang and Lian, 2021; Wang et al., 2023; Thamizharasan et al., 2024; Xia et al., 2023), which hinder the part sharing needed for diacritics, pictographs, and cursive writing. A path forward is to adopt compositional encodings and trajectory-aware stroke programs. With such encodings and the right data, extending from Latin alphanumerics to open-ended writing systems appears to be a problem of scale rather than principle.

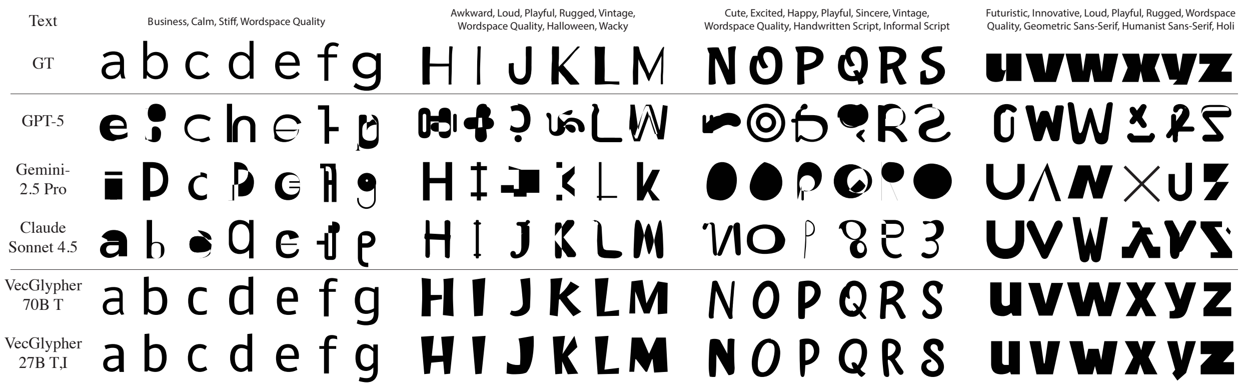

Text
Business, Calm, Stiff, Wordspace Quality
Awkward, Loud, Playful, Rugged, Vintage, Wordspace Quality, Halloween, Wacky
Cute, Excited, Happy, Playful, Sincere, Vintage, Wordspace Quality, Handwritten Script, Informal Script
Futuristic, Innovative, Loud, Playful, Rugged, Wordspace Quality, Geometric Sans-Serif, Humanist Sans-Serif, Holi
GT
GPT-5
Gemini-2.5 Pro
Claude Sonnet 4.5
VecGlypher 70B T
VecGlypher 27B T,I

**Figure 6 Text-referenced comparisons to general LLMs.** Rows show GT, three flagship/budget multimodal LLMs, and our VecGlypher models. General LLMs often fail to output valid, closed paths or the requested character, whereas VecGlypher (27B/70B) consistently produces watertight, stylistically faithful vectors for the same prompts. More results can be found in the supplementary materials.

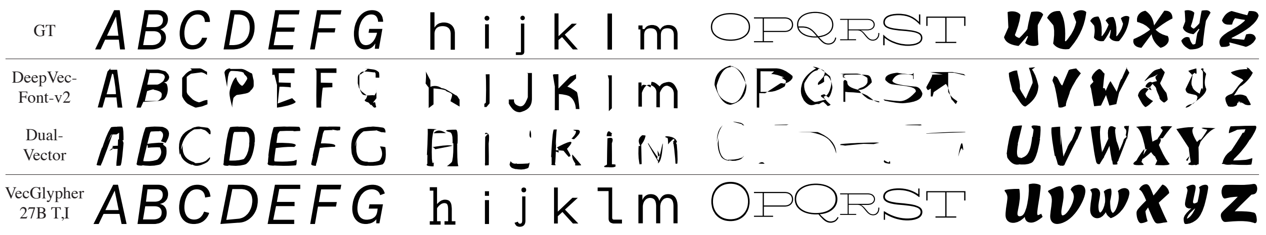


**Figure 7 Image-referenced comparisons to vector-font baselines.** On unseen font families, DeepVecFont-v2 (Wang et al., 2023) and DualVector (Liu et al., 2023b) struggle with thin strokes and contour closure. VecGlypher-27B (T,I) preserves delicate structures, follows target content, and closes contours. Please refer to the supplementary materials for the comprehensive comparisons.

*Conclusions.* We introduced VecGlypher, a unified multimodal language model that directly generates vector glyphs from text or exemplar images. Built on a dedicated data pipeline and a two-stage training recipe that combines large-scale noisy continuation with expert-tagged post-training, VecGlypher achieves substantially improved recognizability, geometry, and distributional fidelity than both general-purpose LLMs and specialized baselines in cross-family OOD evaluations. Ablation studies underline the importance of model scale, absolute-coordinate serialization, and staged supervision for stable SVG decoding and faithful style transfer. By lowering the barrier to font creation and coupling language, imagery, and vector geometry, VecGlypher positions LLMs as a scalable foundation for future multimodal design tools.

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

AI Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1(1):4, 2024.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Gunilla Borgefors. Distance transformations in digital images. *Computer vision, graphics, and image processing*, 34(3): 344–371, 1986.

Gunilla Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 10(6):849–865, 2002.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Jingjing Chang, Yixiao Fang, Peng Xing, Shuhan Wu, Wei Cheng, Rui Wang, Xianfang Zeng, Gang Yu, and Hai-Bao Chen. Oneig-bench: Omni-dimensional nuanced evaluation for image generation. *arXiv preprint arXiv:2506.07977*, 2025.

Tianlang Chen, Zhaowen Wang, Ning Xu, Hailin Jin, and Jiebo Luo. Large-scale tag-based font retrieval with generative feature learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9115–9124, 2019.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Envato Pty Ltd. Fonts — envato elements, 2025. Subscription library of commercial fonts.

Zigang Geng, Yibing Wang, Yeyao Ma, Chen Li, Yongming Rao, Shuyang Gu, Zhao Zhong, Qinglin Lu, Han Hu, Xiaosong Zhang, et al. X-omni: Reinforcement learning makes discrete autoregressive image generative models great again. *arXiv preprint arXiv:2507.22058*, 2025.

Google LLC. Google fonts, 2025. Free and open-source font library.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Hideaki Hayashi, Kohtaro Abe, and Seiichi Uchida. Glyphgan: Style-consistent font generation based on generative adversarial networks. *Knowledge-Based Systems*, 186:104927, 2019.

Haibin He, Xinyuan Chen, Chaoyue Wang, Juhua Liu, Bo Du, Dacheng Tao, and Qiao Yu. Diff-font: Diffusion model for robust one-shot font generation. *International Journal of Computer Vision*, 132(11):5372–5386, 2024.

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. *Advances in neural information processing systems*, 30, 2017.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. arXiv preprint arXiv:2211.11319, 2022. Uses Score Distillation Sampling for text-to-SVG.

Jihun Kang, Daichi Haraguchi, Seiya Matsuda, Akisato Kimura, and Seiichi Uchida. Shared latent space of font shapes and their noisy impressions. In *Advances in Multimedia Information Processing - MMM 2022*, pages 146–157. Springer, 2022.

Lei Kang, Fei Yang, Kai Wang, Mohamed Ali Souibgui, Lluis Gomez, Alicia Fornés, Ernest Valveny, and Dimosthenis Karatzas. Grif-dm: Generation of rich impression fonts using diffusion models. arXiv preprint arXiv:2408.07259, 2024. Accepted to ECAI 2024.

Hua Li and Zhouhui Lian. Hfh-font: Few-shot chinese font synthesis with higher quality, faster speed, and higher resolution. arXiv preprint arXiv:2410.06488, 2024. Accepted to SIGGRAPH Asia 2024.

Tzu-Mao Li, Michal Lukač, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics*, 39(6):193:1–193:15, 2020. Proceedings of SIGGRAPH Asia 2020.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023a.

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*, pages 14193–14202, 2023b.

Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7929–7938, 2019.

Seiya Matsuda, Akisato Kimura, and Seiichi Uchida. Impressions2font: Generating fonts by specifying impressions. In *Document Analysis and Recognition - ICDAR 2021*, pages 739–754. Springer, 2021.

Seiya Matsuda, Akisato Kimura, and Seiichi Uchida. Font generation with missing impression labels. arXiv preprint arXiv:2203.10348, 2022. Accepted to ICPR 2022.

Peter O'Donovan, Janis Libeks, Aseem Agarwala, and Aaron Hertzmann. Exploratory font selection using crowdsourced attributes. *ACM Transactions on Graphics*, 33(4):1–9, 2014. Proceedings of SIGGRAPH 2014.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

Pradyumna Reddy, Zhifei Zhang, Matthew Fisher, Hailin Jin, Zhaowen Wang, and Niloy J. Mitra. A multi-implicit neural representation for fonts. arXiv preprint arXiv:2106.06866, 2021. NeurIPS 2021.

Juan A. Rodriguez, Abhay Puri, Shubham Agarwal, Issam H. Laradji, Pau Rodriguez, Sai Rajeswar, David Vázquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images and text. arXiv preprint arXiv:2312.11556, 2023. Multimodal LLM for SVG generation.

Yuki Tatsukawa, I-Chao Shen, Anran Qi, Yuki Koyama, Takeo Igarashi, and Ariel Shamir. Fontclip: A semantic typography visual-language model for multilingual font applications. *Computer Graphics Forum*, 43(2), 2024. Eurographics 2024.

Yuki Tatsukawa, I-Chao Shen, Mustafa Doga Dogan, Anran Qi, Yuki Koyama, Ariel Shamir, and Takeo Igarashi. Fontcraft: Multimodal font design using interactive bayesian optimization. arXiv preprint arXiv:2502.11399, 2025. CHI 2025.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

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)*, pages 7943–7952, 2024. Official CVPR 2024 paper.

Pan Wang, Xun Zhang, Zhibin Zhou, Peter Childs, Kunpyo Lee, Maaike Kleinsmann, and Stephen Jia Wang. Typeface generation through style descriptions with generative models. In *Proceedings of the 2024 International Conference on Virtual Reality Continuum and its Applications in Industry (VRCAI)*, pages 1–12, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Yizhi Wang and Zhouhui Lian. Deepvecfont: synthesizing high-quality vector fonts via dual-modality learning. *ACM Transactions on Graphics*, 40(6):1–15, 2021. Proceedings of SIGGRAPH Asia 2021.

Yizhi Wang, Yue Gao, and Zhouhui Lian. Attribute2font: creating fonts you want from attributes. *ACM Transactions on Graphics*, 39(4):1–13, 2020. Proceedings of SIGGRAPH 2020.

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)*, pages 18320–18328. IEEE, 2023.

Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. Iconshop: Text-guided vector icon synthesis with autoregressive transformers. arXiv preprint arXiv:2304.14400, 2023. Text-guided vector icon synthesis.

Ronghuan Wu, Wanchao Su, and Jing Liao. Chat2svg: Vector graphics generation with large language models and image diffusion models. arXiv preprint arXiv:2411.16602, 2024. Hybrid LLM and diffusion framework for text-to-SVG.

Zeqing Xia, Bojun Xiong, and Zhouhui Lian. Vecfontsdf: Learning to reconstruct and synthesize high-quality vector fonts via signed distance functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1848–1857, 2023. Official CVPR 2023 paper.

Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. Svgdreamer: Text guided svg generation with diffusion model. arXiv preprint arXiv:2312.16476, 2023. Accepted to CVPR 2024.

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 Computer Vision and Pattern Recognition Conference*, pages 19487–19497, 2025.

Fangzhou Xu, Ke Sun, Dayu Duan, Yiqi Zhong, Ziqi Li, Zhizhao Zhang, Gui-Song Xia, Xiaoyong Shen, Baihua Zhang, and Jiaya Jia. Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16047–16057, 2022. LIVE.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Yiying Yang, Wei Cheng, Sijin Chen, Xianfang Zeng, Fukun Yin, Jiaxu Zhang, Liao Wang, Gang Yu, Xingjun Ma, and Yu-Gang Jiang. Omnisvg: A unified scalable vector graphics generation model. arXiv preprint arXiv:2504.06263, 2025b. Proposes MMSVG-2M dataset and multimodal SVG generation.

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. arXiv preprint arXiv:2312.12142, 2023. Accepted to AAAI 2024.

## Appendix

## A Dataset Statistics (Cont')

*Training data before filtering.* Figure 1 visualizes the token-length distributions of the *training* corpora *before* applying the typography-specific filtering described in Sec. 3.3 of the main paper.

For each font, we compute:

- **Input token length**: the number of tokens in the serialized style description (for text-referenced samples).
- **Output token length**: the number of tokens in the tokenized SVG `<path>` string for each glyph.

The top row of Fig. 1 shows the distributions for Google Fonts; the bottom row shows Envato.

*Heavy-tailed SVG sequences.* On both corpora, input style tokens are relatively concentrated around a narrow range (roughly one to two dozen tags per font, consistent with Fig. 2 in the main text).

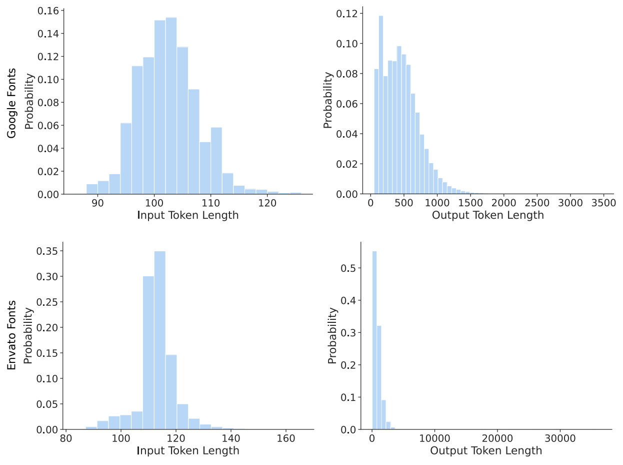
In contrast, the *SVG path* token lengths are strongly long-tailed, especially for Envato: a small fraction of fonts contain glyphs whose serialized paths span tens of thousands of tokens. These extremely long sequences mostly correspond to malformed outlines, redundant contour duplication, or decorative symbols.

Such heavy tails are problematic for autoregressive training: they increase sequence entropy, cause unstable gradients, and exacerbate error accumulation during decoding. This motivates the length-based pruning strategy ("Length by pangram") described in Sec. 3.3 of the main paper.

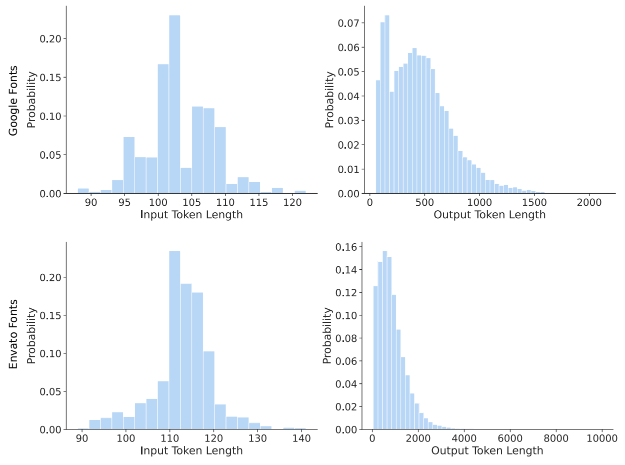
*Filtered test-set statistics.* Fig. 2 reports analogous statistics for the *testing* split, after all filtering steps and split-by-family partitioning. Compared to Fig. 1, the output token distributions are significantly better behaved:

- The vast majority of glyphs fall into a moderate length regime, enabling stable training and evaluation.
- Google Fonts remains more compact than Envato, consistent with its expert-curated nature and more regular outlines.

These plots confirm that the preprocessing pipeline successfully removes pathological fonts while retaining a broad diversity of typographic structure: after filtering we keep 2,497 Google Fonts and 39,497 Envato fonts, with 157,899 and 2,495,363 glyphs respectively. We exclude Envato fonts from testing because their tags are generally noisy and lack meaningful visual descriptions.



**(a)** Dataset Statistics for training set before filtering.



**(b)** Dataset Statistics for testing set.

## B Prompt Templates and Samples

We use a strict prompting scheme to encourage syntactically valid SVG paths and to cleanly separate role instructions, font style descriptions, and target content. The main paper briefly mentions a "strict system prompt" for SVG-only outputs; here we describe the full templates, which are summarized in Table 1 of the supplementary material. The system prompt and the instruction prompts for both text- and image-referenced vector glyph generation. The "{{FONT STYLE}}" is the bag of style tags, and the "{{GLYPH CHARACTER}}" is a single given character in "0–9, a–z, A–Z".

**System Prompt**

You are a specialized vector glyph designer creating SVG path elements.
Critical requirements:
- Each glyph must be a complete, self-contained <path> element, in reading order of the given text.
- Terminate each <path> element with a newline character.
- Output ONLY valid SVG <path> elements.

**Instruction Prompt for Text-Referenced Vector Glyph Generation**

Font design requirements: {{FONT STYLE}}.
Text content: {{GLYPH CHARACTER}}.

**Instruction Prompt for Image-Referenced Vector Glyph Generation**

Font design requirements: faithfully match the provided reference images for style and metrics.
Text content: {{GLYPH CHARACTER}}.

**Samples of instruction prompts for text-referenced vector glyph generation 1**

Font design requirements: stiff, wordspace quality, superellipse sans, rounded sans-serif, 400 weight, sans-serif, normal style, display, neo grotesque sans-serif, drawing quality, cute.
Text content: V.

**Samples of instruction prompts for text-referenced vector glyph generation 2**

Font design requirements: wordspace quality, rounded sans-serif, sans-serif, competent, grotesque sans, 900 weight, italic style, calm.
Text content: b.

**Samples of instruction prompts for text-referenced vector glyph generation 3**

Font design requirements: fancy, 400 weight, vintage, sophisticated, handwritten script, brush, wordspace quality, handwriting, sincere, informal script, drawing quality, excited, normal style, cute, artistic.
Text content: 6.

## C Additional Metrics

The main paper evaluates VecGlypher and baselines with Relative OCR Accuracy (R-ACC), Chamfer Distance (CD), CLIP similarity, DINO similarity, and FID.

Here we introduce several additional metrics that capture complementary aspects of glyph quality and provide full results in Tables 3–6 of the supplementary.

All raster-based metrics are computed on 192x192 grayscale renderings, using the same rasterization pipeline as for the qualitative figures.

We further introduce:

- **FID (C).** In addition to FID computed in Inception space, we report **FID (C)**, where both real and generated glyphs are embedded with CLIP ViT-B/32 features before computing the Fréchet distance. This emphasizes semantic similarity between the stylized glyph images and is more sensitive to high-level appearance than to low-level pixel noise.
- **R-ACC (U)**: R-ACC in the main paper measures OCR accuracy normalized by the accuracy on ground-truth glyphs, allowing scores slightly above 100 because of OCR variability.
- **R-ACC (U)**: The OCR outputs are case-normalized: upper- and lowercase characters that share the same identity (e.g., "a" vs "A") are treated as correct. This disentangles shape-level recognition errors from case mismatches.
- **CD (T) and CD (ST)**: Our primary Chamfer Distance (CD) computes a symmetric distance between two point clouds sampled from the SVG outlines after normalizing each glyph to the $[-1, 1]$ box.
- **CD (T)**: we first align prediction to ground truth via Iterative Closest Point (ICP) optimizing only a 2D translation. This compensates for small global shifts, e.g., from imperfect baseline alignment.
- **CD (ST)**: we run ICP optimizing both translation and isotropic scale. This is suitable for italic or script fonts where rotation is ambiguous but overall scale may drift. We intentionally do *not* optimize rotation in either variant, since many italic fonts are skewed and rotational alignment could distort their intended slant.
- **L2**: L2 is the mean squared error between rasterized predictions and ground truths, averaged over pixels and glyphs. It captures dense per-pixel discrepancies but is insensitive to human perception.
- **LPIPS**: We report Learned Perceptual Image Patch Similarity (LPIPS) using a standard VGG backbone. LPIPS measures perceptual distance between images by comparing deep feature activations, correlating better with human judgement than L2.
- **PSNR and SSIM**: We additionally report Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM), both computed on grayscale rasterizations. PSNR summarizes global reconstruction fidelity; SSIM emphasizes local luminance, contrast, and structural agreement.

## D Additional Baselines

To complement the proprietary LLMs evaluated in the main paper, we benchmark two classes of publicly available models on text-referenced glyph generation:

- **Open-weight multimodal LLMs**: Llama 3.3 70B Instruct, Gemma 3 27B IT, and Qwen 3 30B A3B Instruct (2507). These models are strong general-purpose multimodal assistants.
- **Vector-graphics LLM**: OmniSVG, an open-source model trained specifically for SVG generation and image vectorization. We skip LLM4SVG and StarVector since their text-to-SVG models are not publicly released.

Overall, we observe:

- **Extremely low recognizability.** R-ACC and R-ACC(U) remain close to zero for all open-weight LLMs and for OmniSVG, meaning that the OCR engine rarely recognizes the intended characters.
- **Poor geometry and image quality.** CD, CD(T), and CD(ST) are an order of magnitude higher than those of VecGlypher, while L2, LPIPS, and SSIM indicate severe geometric distortions or failure to render the glyph at all.
- **Limited benefit from SVG specialization.** OmniSVG, despite being trained for SVG icons and vectorization, produces particularly poor results for glyphs: a large fraction of outputs are invalid paths or generic shapes unrelated to the requested character or style.

These trends mirror the conclusions of Sec. 1 and Sec. 5 in the main paper: off-the-shelf LLMs and vector-graphics LLMs that perform well on icons or simple SVG drawings do not transfer to typography, which imposes stricter geometric, stylistic, and topological constraints.

## E Additional Qualitative Results

Please refer to the HTML pages for comprehensive results. They include both ablation studies and comparisons for text-referenced and image-referenced vector-glyph generation tasks.

**Table 10** Text-referenced ablations on data and model size.

| Data | Size | Repr. | R-ACC | CD | CLIP | DINO | FID | FID (C) | R-ACC (U) | CD (T) | CD (ST) | L2 | LPIPS | PSNR | SSIM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Google | 4B | Rel. | 73.96 | 4.29 | 26.02 | 90.27 | 15.57 | 1.84 | 75.51 | 3.83 | 4.40 | 0.196 | 0.245 | 8.30 | 0.643 |
| | 4B | Abs. | 66.66 | 3.75 | 26.04 | 89.92 | 14.69 | 2.19 | 68.68 | 3.36 | 3.85 | 0.205 | 0.251 | 8.14 | 0.631 |
| | 27B | Rel. | 92.81 | 2.31 | 26.38 | 93.01 | 5.81 | 0.440 | 93.40 | 1.96 | 2.32 | 0.158 | 0.212 | 9.21 | 0.679 |
| | 27B | Abs. | 94.91 | 1.98 | 26.43 | 93.43 | 3.96 | 0.250 | 95.30 | 1.69 | 2.00 | 0.152 | 0.204 | 9.50 | 0.686 |
| Envato | 27B | Rel. | 93.99 | 3.89 | 26.43 | 89.79 | 30.68 | 1.63 | 94.86 | 3.35 | 3.90 | 0.211 | 0.265 | 7.46 | 0.621 |
| | 27B | Abs. | 93.84 | 3.63 | 26.45 | 90.24 | 20.43 | 1.31 | 95.18 | 3.22 | 3.64 | 0.197 | 0.254 | 7.78 | 0.636 |
| E + G | 27B | Rel. | 94.39 | 2.56 | 26.39 | 93.17 | 5.83 | 0.458 | 94.69 | 2.20 | 2.57 | 0.147 | 0.200 | 9.60 | 0.692 |
| | 27B | Abs. | 95.59 | 2.14 | 26.45 | 93.66 | 4.86 | 0.289 | 96.27 | 1.84 | 2.14 | 0.141 | 0.194 | 9.81 | 0.698 |
| E $\rightarrow$ G | 27B | Rel. | 99.88 | 1.71 | 26.52 | 94.07 | 3.57 | 0.142 | 99.90 | 1.44 | 1.71 | 0.141 | 0.194 | 9.74 | 0.697 |
| | 27B | Abs. | 101.0 | 1.67 | 26.53 | 94.34 | 3.47 | 0.135 | 100.72 | 1.40 | 1.65 | 0.138 | 0.191 | 9.88 | 0.700 |

**Table 11** Image-referenced ablations on data and model size.

| Data | Size | Repr. | R-ACC | CD | CLIP | DINO | FID | FID (C) | R-ACC (U) | CD (T) | CD (ST) | L2 | LPIPS | PSNR | SSIM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Google | 4B | Rel. | 83.48 | 2.36 | 25.90 | 93.26 | 9.38 | 1.08 | 84.66 | 1.97 | 2.45 | 0.155 | 0.206 | 9.60 | 0.684 |
| | 4B | Abs. | 81.94 | 2.11 | 25.89 | 93.11 | 7.94 | 1.07 | 83.53 | 1.77 | 2.19 | 0.156 | 0.205 | 9.58 | 0.682 |
| | 27B | Rel. | 94.42 | 1.53 | 26.03 | 94.88 | 4.07 | 0.293 | 95.30 | 1.24 | 1.50 | 0.127 | 0.179 | 10.65 | 0.716 |
| | 27B | Abs. | 96.46 | 1.41 | 26.04 | 95.15 | 2.84 | 0.147 | 97.03 | 1.12 | 1.34 | 0.121 | 0.172 | 10.79 | 0.722 |
| E-G (I) | 27B | Rel. | 98.90 | 1.17 | 26.06 | 95.69 | 2.51 | 0.116 | 99.26 | 0.910 | 1.09 | 0.109 | 0.159 | 11.22 | 0.736 |
| | 27B | Abs. | 97.88 | 1.16 | 26.06 | 95.84 | 2.55 | 0.101 | 98.43 | 0.910 | 1.09 | 0.107 | 0.156 | 11.36 | 0.740 |
| E-G (T+I) | 27B | Rel. | 99.08 | 1.21 | 26.07 | 95.65 | 2.48 | 0.120 | 99.44 | 0.950 | 1.13 | 0.112 | 0.162 | 11.38 | 0.734 |
| | 27B | Abs. | 99.12 | 1.18 | 26.07 | 95.82 | 2.32 | 0.095 | 99.82 | 0.930 | 1.10 | 0.110 | 0.158 | 11.35 | 0.736 |

**Table 12** Text-referenced comparisons with general LLMs.

| Model | R-ACC | CD | CLIP | DINO | FID | FID (C) | R-ACC (U) | CD (T) | CD (ST) | L2 | LPIPS | PSNR | SSIM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPT-5 mini | 4.17 | 10.70 | 24.75 | 79.65 | 63.86 | 13.78 | 4.96 | 10.49 | 10.74 | 0.382 | 0.412 | 4.45 | 0.432 |
| Gemini-2.5 Flash | 4.89 | 13.78 | 25.26 | 78.62 | 86.03 | 12.77 | 7.15 | 13.38 | 13.78 | 0.364 | 0.379 | 4.85 | 0.480 |
| Claude Haiku 4.5 | 32.38 | 7.60 | 25.61 | 84.69 | 35.07 | 5.85 | 38.80 | 7.17 | 7.60 | 0.289 | 0.332 | 5.96 | 0.545 |
| GPT-5 | 43.98 | 6.12 | 25.95 | 86.92 | 29.00 | 3.71 | 48.03 | 5.26 | 6.26 | 0.301 | 0.346 | 5.68 | 0.521 |
| Gemini 2.5 Pro | 24.04 | 8.22 | 25.57 | 83.77 | 47.82 | 7.39 | 27.06 | 7.72 | 8.22 | 0.319 | 0.347 | 5.48 | 0.518 |
| Claude Sonnet 4.5 | 46.65 | 5.28 | 25.99 | 88.31 | 19.59 | 2.81 | 52.99 | 4.58 | 5.34 | 0.253 | 0.305 | 6.62 | 0.580 |
| VecGlypher 27B T,I,R | 99.62 | 1.77 | 26.53 | 93.97 | 3.50 | 0.139 | 99.58 | 1.51 | 1.76 | 0.143 | 0.197 | 9.64 | 0.695 |
| VecGlypher 27B T,I,A | 100.5 | 1.72 | 26.53 | 94.22 | 3.46 | 0.134 | 100.34 | 1.44 | 1.68 | 0.140 | 0.193 | 9.83 | 0.698 |
| VecGlypher 70B T,R | 100.1 | 1.70 | 26.53 | 94.10 | 3.45 | 0.140 | 100.57 | 1.43 | 1.67 | 0.140 | 0.193 | 9.80 | 0.699 |
| VecGlypher 70B T,A | 100.4 | 1.68 | 26.54 | 94.28 | 3.34 | 0.136 | 100.71 | 1.40 | 1.66 | 0.139 | 0.192 | 9.85 | 0.699 |

**Table 13** Image-referenced comparisons with vector-font baselines.

| Model | R-ACC | CD | CLIP | DINO | FID | FID (C) | R-ACC (U) | CD (T) | CD (ST) | L2 | LPIPS | PSNR | SSIM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DeepVecFont-v2 | 37.86 | 14.58 | 24.81 | 79.41 | 115.5 | 16.45 | 49.29 | 14.58 | 14.58 | 0.243 | 0.320 | 6.70 | 0.599 |
| DualVector | 49.20 | 16.45 | 25.07 | 79.57 | 105.5 | 14.73 | 65.59 | 16.44 | 16.44 | 0.190 | 0.293 | 7.99 | 0.653 |
| VecGlypher 27B T,I,R | 99.08 | 1.21 | 26.07 | 95.65 | 2.48 | 0.120 | 99.44 | 0.950 | 1.13 | 0.112 | 0.162 | 11.38 | 0.734 |
| VecGlypher 27B T,I,R | 99.12 | 1.18 | 26.07 | 95.82 | 2.32 | 0.095 | 99.82 | 0.930 | 1.10 | 0.110 | 0.158 | 11.35 | 0.736 |

**Table 14** Performance of the open weight LLMs and the vector graphic LLM.

| Model | R-ACC | CD | CLIP | DINO | FID | FID (C) | R-ACC (U) | CD (T) | CD (ST) | L2 | LPIPS | PSNR | SSIM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Llama3.3 70B Instruct | 0.08 | 38.28 | 24.84 | 75.85 | 143.73 | 21.49 | 0.08 | 38.26 | 38.30 | 0.374 | 0.388 | 4.64 | 0.480 |
| Gemma3 27B IT | 0.12 | 17.90 | 24.78 | 74.26 | 162.89 | 23.24 | 0.11 | 17.44 | 17.90 | 0.407 | 0.397 | 4.28 | 0.450 |
| Qwen3 30B A3B Instruct 2507 | 0.39 | 26.15 | 24.66 | 74.03 | 148.78 | 22.47 | 0.55 | 26.07 | 26.15 | 0.418 | 0.422 | 4.219 | 0.427 |
| OmniSVG | 0.01 | 63.70 | 23.26 | 69.35 | 229.38 | 35.75 | 0.01 | 63.69 | 63.69 | 0.496 | 0.452 | 3.50 | 0.356 |