CONTINUOUS REPRESENTATIONS IMPROVE ON DISCRETE ONES FOR SVG GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Current autoregressive SVG generation models represent coordinates as a sequence of discrete tokens, even though those values describe continuous, geometric positions. We propose an alternative, *continuous* approach that encodes coordinates through a value encoder and predicts them via regression through a value decoder, while retaining ordinary token prediction for drawing commands and markup. In evaluations, our approach achieves higher reconstruction quality than all evaluated discrete baselines. Moreover, data-scaling experiments with 5%, 25%, and 100% of the training data show a persistent advantage over discrete approaches at every training-set size, showing that our approach learns faster and peaks higher than discrete baselines. We investigate this advantage by analyzing *numerical representations*, *coordinate generalization*, and *coordinate precision*. For numerical representations, we find that our embeddings vary more smoothly and are more linearly decodable than discrete representations. For coordinate generalization, continuous predictions remain stable on held-out coordinates, while discrete predictions degrade. For coordinate precision, continuous predictions yield lower teacher-forced coordinate error than their discrete counterparts on fractional targets. Together, these results favor a more natural division of labor: continuous values for geometry and discrete tokens for drawing structure. We will publicly release our models and code.

1 INTRODUCTION

Moving an SVG point from $(37, 80)$ to $(38, 80)$ shifts it one unit to the right. The change is simple geometrically, but a model that generates SVG as text must express it through whatever digits or subwords its tokenizer provides. A dedicated coordinate vocabulary, as used by OmniSVG (Yang et al., 2025), can represent each point in one step, but treats the two points as separate classes. Neither choice explicitly tells the model how far the points are apart. Thus, the way that SVG generators (Rodriguez et al., 2025) represent numbers becomes a basic design choice: how should they learn geometry alongside discrete drawing instructions?

We propose a *continuous* approach that encodes coordinates as numerical values and predicts them by regression. We use a shared Fourier-feature encoder to map coordinates into the transformer's embedding space, while a value decoder predicts the next coordinate from its hidden state. Commands and markup retain ordinary token embeddings and next-token prediction. Each predicted coordinate is then re-encoded to condition the rest of the drawing. The encoder shares parameters across the coordinate domain, and the regression loss penalizes numerical distance directly. Together, these design choices give the model a geometric inductive bias without changing the SVG grammar.

This interface also separates numerical resolution from vocabulary size. For OmniSVG's quantized coordinates, the integer grid runs from 0 through the coordinate scale M, giving $M + 1$ positions per axis and $(M + 1)^2$ tokens for all coordinate pairs. With our approach, a value encoder and decoder handle all coordinates, so no additional vocabulary is needed. We study two primary variants: *scalar* prediction emits x then y in separate steps, while *paired* prediction emits both (x, y) together. Scalar prediction uses two coordinate steps per point rather than one. Since commands and markup are shared, its full sequences are about 50% longer overall. Paired generation is faster, while scalar prediction gives stronger raster-to-SVG quality. We also study a *standard* variant where we use the backbone's standard tokenizer and x and y are not necessarily aligned to distinct tokens. Building on

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.

xVal and FoNE (Golkar et al., 2024; Zhou et al., 2026), we use explicit numerical representations for SVG generation, where successive coordinate predictions must form a coherent drawing.

Our experiments examine what this representation learns and how it affects generation. Learned continuous embeddings are smoother and more linearly decodable than discrete ones. In our generalization test, when we exclude selected coordinates from training, continuous predictions remain stable, while discrete accuracy degrades. In a separate precision experiment, continuous predictions give lower teacher-forced error than their discrete counterparts on fractional targets, showing that we gain increased precision with continuous vs. quantized-integer coordinates.

For SVG generation, we compare standard, discrete, and continuous approaches using a shared transformer backbone and training on the same data. In our evaluations, continuous approaches beat discrete ones (both scalar and paired), achieving better values on our image-quality metrics: DINOv3, SSIM, LPIPS, and MSE. This shows that continuous outperforms, whether we prefer speed/conciseness (paired) or output quality (scalar). Comparisons across training-data scales of 5%, 25%, and 100% show that continuous approaches learn faster and peak higher than discrete baselines.

Our three core contributions are:

1. **Continuous SVG interfaces.** We introduce a continuous coordinate interface for autoregressive SVG generation that encodes coordinate values continuously and predicts them by regression, while retaining discrete token prediction for SVG commands and markup.
2. **Matched comparison.** We compare continuous interfaces with standard tokenization and discrete coordinate tokens under matched training conditions, assessing generation quality and efficiency across model scales and data budgets, as well as cross-dataset and text-to-SVG generalization.
3. **Analysis of coordinate behavior.** We analyze numerical embeddings, coordinate generalization, and fractional-target precision, finding smoother, more linearly decodable continuous embeddings, stable predictions on held-out coordinates, and lower error on fractional targets.

2 Related Work

Our work connects research on generating vector graphics with number representations in sequences.

Vector-graphics generation. Early vector-graphics methods used representations and decoders tailored to geometry. SVG-VAE learns sequential latent representations of vector graphics (Lopes et al., 2019), while DeepSVG quantizes coordinates and predicts their categories with a non-autoregressive decoder (Carlier et al., 2020). Im2Vec predicts Bézier control points with a path decoder trained through differentiable rendering (Reddy et al., 2021), and SVGformer encodes continuous geometry for representation learning rather than autoregressive SVG generation (Cao et al., 2023). A separate line optimizes graphical output directly: VectorFusion (Jain et al., 2023) and SVGDreamer (Xing et al., 2024) use diffusion guidance to optimize vector primitives, while NeuralSVG optimizes an implicit layered representation (Polaczek et al., 2025). Chat2SVG brings language models into this process by generating SVG templates and refining their paths (Wu et al., 2025). More recent systems instead generate SVG programs autoregressively, from tokenized paths in IconShop (Wu et al., 2023) to language-model-generated programs in LLM4SVG (Xing et al., 2025) and SVGen (Wang et al., 2025). This shift makes coordinate serialization part of the sequence-modeling problem. StarVector retains ordinary SVG text (Rodriguez et al., 2025), OmniSVG uses discrete coordinate-command tokens on a quantized grid (Yang et al., 2025), and HiVG compresses discrete sequences with learned segment tokens (Xing et al., 2026). These interfaces trade flexible text, explicit quantization, and learned sequence compression, motivating a comparison of coordinate representations within autoregressive SVG generation.

Numerical representations in sequence models. The same question arises more broadly in how sequence models represent numbers. With BPE and WordPiece, numbers are split into vocabulary units whose embeddings do not explicitly encode numerical distance (Sennrich et al., 2016; Wu et al., 2016). xVal makes the value explicit by scaling a learned number embedding by the scalar and using a regression head for numerical prediction (Golkar et al., 2024). Fourier features provide another continuous representation, mapping a value to sinusoidal features at multiple frequencies before

learned projection (Tancik et al., 2020). Lower frequencies capture broad changes in the value, while higher frequencies expose finer variation, giving the model features at multiple scales. FoNE applies Fourier features to single-token number embeddings, but predicts output digits in parallel (Zhou et al., 2026). Together, xVal and FoNE illustrate complementary ways to add numerical structure, through value-scaled embeddings or frequency features, while differing in how numerical outputs are predicted. For SVG generation, the open question is how coordinate representations behave when numerical values and drawing commands must be generated together.

We address this question by comparing standard tokenization, discrete coordinate tokens, and continuous numerical representations under controlled conditions.

3 A CONTINUOUS APPROACH TO SVG GENERATION

We keep SVG commands and markup discrete, while representing coordinates as continuous values that the model can encode and predict.

3.1 REPRESENTING SVG STRUCTURE AND COORDINATES

An SVG program combines discrete drawing instructions with numerical geometry. Continuous coordinates are normalized as $u = c/M$, where M is the coordinate scale set by the shared pre-processing. This normalization preserves real-valued coordinates rather than restricting them to fixed positions. We preserve commands and markup as ordinary tokens and represent coordinates with two continuous interfaces. Following OmniSVG's paired serialization (Yang et al., 2025), the *paired* variant replaces each coordinate pair with one `[POS]` marker, to which we attach both values. The *scalar* variant instead replaces the pair with two consecutive `[NUM]` markers, each carrying one value. For example, `L 37 80` becomes `L [POS]` with the attached pair $(37, 80)$, or `L [NUM] [NUM]` with the attached scalars 37 and 80. Figure 1a shows the paired token sequence and its attached coordinate pairs. Paired prediction therefore represents a complete point in one autoregressive step, whereas scalar prediction uses two steps and conditions the prediction of y on x. Appendix A.1 illustrates scalar pre-processing, training, and inference.

Let $s_{1:T}$ denote the token sequence, including numerical markers, and $c_t \in \mathbb{R}^d$ the coordinate attached to a marker at position t. Here $d = 2$ for paired prediction and $d = 1$ for scalar prediction; other positions have no numerical value. A transformer reads the conditioning input and this combined prefix to predict the next token; when that token is `[POS]` or `[NUM]`, the value decoder also predicts the value attached to it. The raster-to-SVG model conditions on an image, while the text-to-SVG variant conditions on a caption. The interface accepts integer and fractional values alike.

3.2 ENCODING NUMERICAL VALUES

The value encoder represents both a coordinate's magnitude and its variation at several scales using Fourier features (Tancik et al., 2020). For a scalar coordinate c, we write $u = c/M$ and construct Fourier features containing the normalized value and K sine and cosine bands:

$$\phi(u) = \left[u,\ (\sin(2\pi 2^k u))_{k=0}^{K-1},\ (\cos(2\pi 2^k u))_{k=0}^{K-1}\right]. \tag{1}$$

Scalar prediction uses $\Phi(c) = \phi(c/M)$. For paired prediction,

$$\Phi(x, y) = [x/M,\ y/M,\ \phi_{\mathrm{F}}(x/M),\ \phi_{\mathrm{F}}(y/M)], \tag{2}$$

where ϕ_{F} contains only the sine and cosine entries of ϕ.

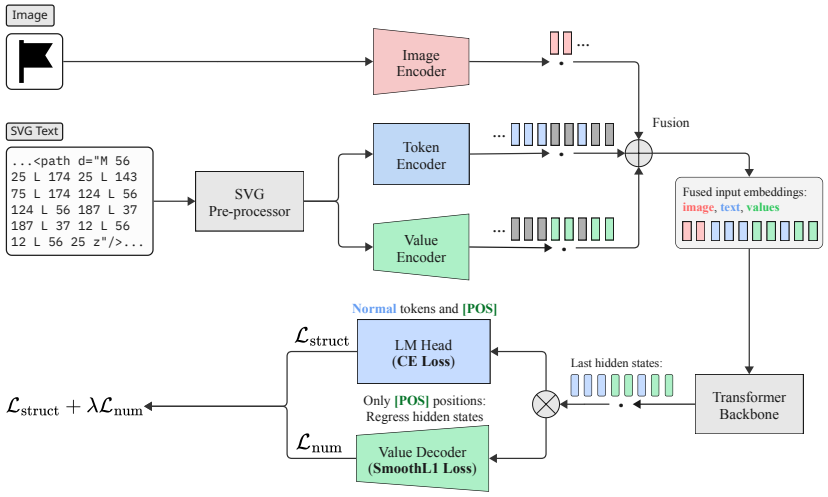
A shared MLP g_ψ maps these features to the transformer's embedding dimension. At a numerical position, its output is added to the token embedding of the marker, namely `[POS]` or `[NUM]`:

$$e_t = E(s_t) + g_\psi(\Phi(c_t)). \tag{3}$$

Non-numeric token positions retain $E(s_t)$. Each model shares its value encoder across coordinate positions; the scalar model also shares it across axes. The same function can encode any coordinate value, including fractions, without adding vocabulary entries. We use Fourier features because they perform best in our encoding ablation, which also compares raw coordinates, radial basis functions, learned splines, and multi-resolution grids (see Table 4). The value encoder and decoder are learned during training, while the Fourier frequencies remain fixed.

SVG Text	M	56	25	L	174	25	L	143	75	L	174	124	L	56	124	L	56	187	L	37	187	L	37	12	L	56	12	L	56	25	z
Tokens	M	[POS]		L	[POS]		L	[POS]		L	[POS]		L	[POS]		L	[POS]		L	[POS]		L	[POS]		L	[POS]		L	[POS]		z
Numbers	∅	(56, 25)		∅	(174, 25)		∅	(143, 75)		∅	(174, 124)		∅	(56, 124)		∅	(56, 187)		∅	(37, 187)		∅	(37, 12)		∅	(56, 12)		∅	(56, 25)		∅

(a) The SVG pre-processor takes SVG text and outputs two streams: tokens to the token encoder, and numbers to the value encoder. [POS] carries one coordinate pair and ∅ marks a position without a numerical attachment.



(b) The fused embeddings feed the backbone and train distinct language and value heads.

Figure 1: Paired pre-processing (top) and the corresponding training interface (bottom). Scalar pre-processing is shown in Table 5.

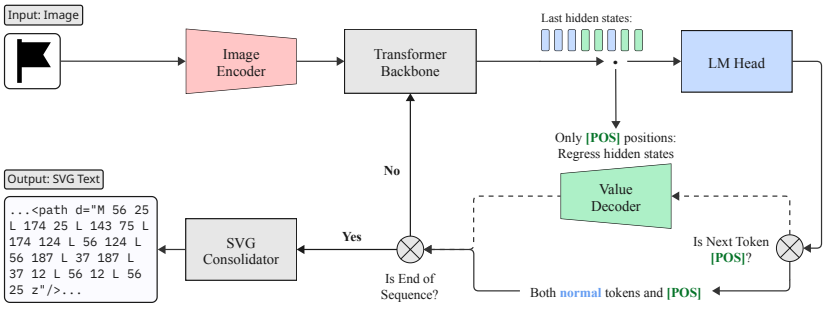


Figure 2: For paired prediction, ordinary tokens continue generation, while each predicted [POS] triggers the value decoder to predict a coordinate pair.

3.3 Predicting coordinates and generating SVGs

Let h_{t-1} be the final transformer state after reading the prefix through position $t-1$. The language head produces the next-token distribution p_t, conditioned on the image and preceding tokens and coordinates, while a separate MLP, which we call the value decoder f_θ, predicts the normalized coordinate:

$$p_t = \text{softmax}(Wh_{t-1}), \qquad \widehat{u}_t = f_\theta(h_{t-1}). \tag{4}$$

The coordinate prediction is used when s_t is [POS] or [NUM]. Both heads read the preceding state before the target coordinate enters the model. The paired head returns two values, and the scalar head returns one.

During generation, the predicted marker and its value $\widehat{c}_t = M\widehat{u}_t$ are appended together to the prefix (Figure 2). Their combined embedding then conditions the following token and coordinate predictions. This internal feedback retains floating-point values without rounding or clipping. After generation,

markers are replaced with their associated values to reconstruct SVG text, and the final formatting step rounds and clips values only when integer serialization is required.

3.4 Training objective and coordinate noise

Training uses teacher forcing with clean next-token and coordinate targets. For the main raster-to-SVG models, we add zero-mean Gaussian noise to input coordinates, with its standard deviation specified in pre-normalization coordinate units and decreased over training, exposing the decoder to perturbed prefixes and improving robustness by mitigating exposure bias. Evaluation uses no added noise. Let $\mathcal{A}$ contain the supervised output-token positions, including numerical markers, and $\mathcal{C} \subseteq \mathcal{A}$ the numerical positions. We average token cross-entropy over $\mathcal{A}$ and coordinate error over the scalar components in $\mathcal{C}$:

$$\mathcal{L}_{\text{struct}} = -\frac{1}{|\mathcal{A}|} \sum_{t \in \mathcal{A}} \log p_t(s_t), \qquad \mathcal{L}_{\text{num}} = \frac{1}{d|\mathcal{C}|} \sum_{t \in \mathcal{C}} \sum_{j=1}^{d} \ell(\widehat{u}_{t,j} - u_{t,j}). \tag{5}$$

Here $u_t = c_t/M$, ℓ is Smooth L1 loss with transition width $1/M$, and both predictions use the perturbed training prefix. We optimize the combined objective $\mathcal{L} = \mathcal{L}_{\text{struct}} + \lambda \mathcal{L}_{\text{num}}$, where λ weights coordinate supervision. Training settings are given in Appendix A.2.

4 Experimental Setup

We evaluate the coordinate interfaces under a shared data pipeline, backbone, and generation protocol.

4.1 Data and pre-processing

We use SVG-Stack (Rodriguez et al., 2025), with the shared pre-processing described in Section 3.1. Following OmniSVG (Yang et al., 2025), discrete interfaces quantize each axis to integer positions from 0 to M; continuous interfaces use the same M for normalization but retain real-valued coordinates. At our scale, $M = 199$, giving 200 discrete positions per axis. We discard samples that fail parsing, rendering, or visual-preservation checks, leaving 1.7M training examples and a held-out test set of 5.71K examples. The data-scaling study uses fixed nested subsets containing 5%, 25%, and 100% of the training set. The reduced subsets contain 84.7K and approximately 423K examples.

We compare pre-processing schemes for the standard interface and use integer-quantized targets on the shared canvas, which gave the strongest reconstruction fidelity in this comparison. Appendix A.8 reports the pre-processing ablation.

4.2 Comparison methods and training

We compare standard tokenization, paired and scalar discrete tokens, and paired and scalar continuous coordinates. In standard tokenization, coordinates are written as decimal integers, so a coordinate can span several tokens. Paired discrete assigns one token to each integer (x, y) pair, while scalar discrete shares $M + 1$ tokens across axes and emits x followed by y. The continuous interfaces use the same paired or scalar grouping with the value encoder and decoder in Section 3. All five interfaces retain textual drawing commands and markup. We study Qwen2-VL-Instruct at the 2B and 7B scales (Wang et al., 2024), using one epoch and an effective batch size of 32. The data-scaling and numerical analyses use the 2B backbone unless stated otherwise. The comparisons isolate numerical interfaces under a shared backbone and grammar; standard tokenization and paired discrete are StarVector-style and OmniSVG-style representations, respectively, rather than reproductions of the complete systems (Rodriguez et al., 2025; Yang et al., 2025). The continuous value modules use FP32, while the transformer uses BF16. We also train all five interfaces for text-to-SVG generation using caption conditioning to test whether the comparison extends beyond raster conditioning.

4.3 Generation evaluation

Following StarVector (Rodriguez et al., 2025), we evaluate rendered reconstructions with DINOv3 similarity (Siméoni et al., 2025), SSIM (Wang et al., 2004), LPIPS (Zhang et al., 2018), and MSE.

Higher DINOv3 similarity and SSIM indicate closer reconstructions; lower LPIPS and MSE indicate smaller perceptual and pixel differences. Rendering and metric definitions are given in Appendix A.4. DINOv3 and SSIM are reported as percentages. For SVGs with transparent backgrounds, white-background evaluation can inflate image-quality scores for nearly empty outputs. We therefore report the mean over white- and black-background renders.

Following StarVector (Rodriguez et al., 2025), we generate five raster-to-SVG candidates ($T = 0, 0.25, 0.5, 0.75, 1$) and select the highest-DINOv3 match. We also follow StarVector in post-processing incomplete SVGs and truncate repetitions and repair the outputs before computing image-quality metrics. Generation time is computed from average serialized target length and measured batch-one latency per autoregressive step, for one candidate and excluding repair and selection. Because captions specify visual content but not an exact pixel layout, we use DINOv3 rather than SSIM as the primary metric for text-to-SVG, comparing each generation with its paired reference drawing.

4.4 Analysis protocols

We probe input embeddings after a complete coordinate has been read, measuring numerical decodability and smoothness. Within each representation, we withhold its atomic coordinate unit: exact (x, y) pairs for paired interfaces and scalar values from both x and y roles for scalar interfaces. Control models see these units during training, whereas holdout models do not, and both are evaluated under teacher forcing on the same per-interface test examples. Separate one-axis tests diagnose transfer of a scalar value between coordinate roles. Appendix A.5 specifies each construction.

For coordinate precision, we compare paired and scalar continuous predictions with their discrete counterparts under matched teacher-forced contexts. Their coordinate-error difference measures the precision gained by continuous prediction. For design ablations, we vary the coordinate encoder, replace regression with a depth-matched categorical head, and test coordinate-input noise. The categorical control retains the continuous input encoder and predicts independent $(M + 1)$-way distributions for x and y from one marker. Detailed settings and probe procedures are given in Appendix A.

5 Results and Analysis

We first compare generation quality and efficiency, then examine data scaling, cross-dataset transfer, coordinate behavior, and design choices.

5.1 Raster-to-SVG generation

At both the 2B and 7B scales, scalar continuous achieves the best DINOv3, SSIM, LPIPS, and MSE in Table 1. The two continuous interfaces lead the text-to-SVG DINOv3 comparison. Paired interfaces use fewer target tokens and have shorter generation times than scalar interfaces. Continuous methods have generation times close to their discrete counterparts, with lower parameter overhead than paired discrete and higher overhead than scalar discrete.

Visual quality and coverage. Figure 3 compares generations for five raster inputs, with more examples in Appendix A.13.

5.2 Training-data scaling and cross-dataset evaluation

Scalar continuous gives the highest DINOv3 and SSIM at every training-set size (Table 2). On MMSVGBench's clean test split of unseen raster inputs (Yang et al., 2025), it also achieves the highest mean SSIM at 2B (Table 1).

5.3 Numerical representations

We test whether numerical input embeddings expose coordinate magnitude and vary regularly between neighboring values. Ridge-probe MAE and R^2 measure linear decodability, while the mean second-difference norm measures local regularity (lower is smoother).

Table 1: Main generation results. Visual quality and efficiency are reported for raster-to-SVG generation; the rightmost columns report SSIM for cross-dataset (MMSVGBench) and DINOv3 for text-to-SVG evaluation. Full text-to-SVG metrics are reported in Appendix A.3. Parameter overhead is relative to the base models.

Method	*Visual quality*				*Efficiency*			*OOD*	*Text-to-SVG*
	SSIM (%) ↑	DINOv3 (%) ↑	LPIPS ↓	MSE ↓	Added params M (%)	Avg. target tokens ↓	Gen. time (s) ↓	SSIM (%) ↑	DINOv3 (%) ↑
2B									
Standard	83.2	82.7	0.159	0.100	0.0 (0.0%)	2065	49.6	83.9	71.6
Paired discrete	84.6	84.6	0.153	0.077	62.0 (2.8%)	610	**14.0**	83.6	69.6
Scalar discrete	85.4	85.3	0.135	0.076	0.9 (0.1%)	933	20.7	83.7	73.1
Paired continuous	85.1	85.5	0.147	0.090	19.5 (0.9%)	**606**	**14.0**	84.2	75.7
Scalar continuous	**86.4**	**85.8**	**0.132**	**0.072**	19.5 (0.9%)	924	21.2	**84.7**	**77.4**
7B									
Standard	87.8	87.2	0.111	0.060	0.0 (0.0%)	2081	52.5	–	–
Paired discrete	89.4	89.3	0.102	0.035	289.3 (3.5%)	604	**14.6**	–	–
Scalar discrete	90.6	90.1	0.083	0.032	4.0 (0.1%)	944	22.0	–	–
Paired continuous	90.3	90.4	0.092	0.044	105.5 (1.3%)	**600**	14.7	–	–
Scalar continuous	**91.7**	**90.9**	**0.076**	**0.024**	105.5 (1.3%)	930	22.4	–	–

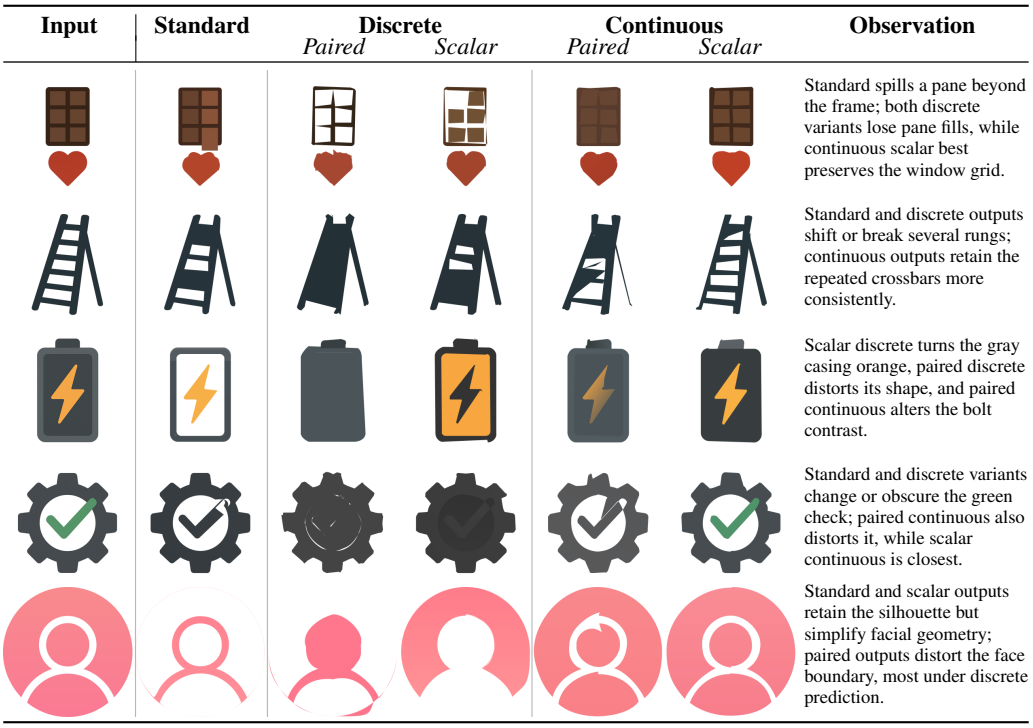


Figure 3: Sample raster-to-SVG generations. The first four rows show harder, OOD examples from MMSVGBench, and the final row shows an SVG-Stack test example; see Appendix A.13 for more examples.

Table 2: Data scaling: Reconstruction quality versus training-data size.

Method	*5% data*		*25% data*		*100% data*	
	SSIM (%) ↑	DINOv3 (%) ↑	SSIM (%) ↑	DINOv3 (%) ↑	SSIM (%) ↑	DINOv3 (%) ↑
Standard	78.7	76.0	81.7	81.0	83.2	82.7
Paired disc.	76.7	70.5	82.7	82.1	84.6	84.6
Scalar disc.	80.9	78.2	84.4	83.7	85.4	85.3
Paired cont.	78.5	74.7	83.0	82.0	85.1	85.5
Scalar cont.	**81.7**	**79.0**	**85.0**	**83.8**	**86.4**	**85.8**

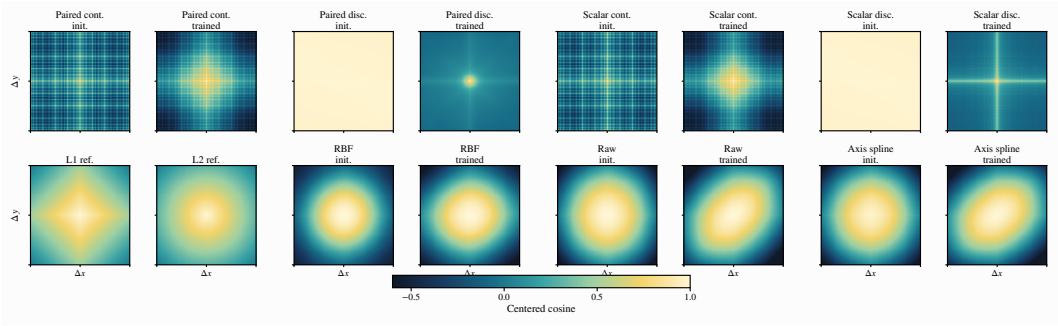


Figure 4: Mean centered-cosine similarity of numerical input embeddings by coordinate displacement $(\Delta x, \Delta y)$, before and after training. The first row compares continuous and discrete paired and scalar interfaces; the second shows L1 and L2 distance references and alternative continuous encoders. Lighter colors indicate greater similarity, and learned panels share a color scale.

Table 3: Consolidated numerical analyses. Embedding columns report linear coordinate decodability and successive-step smoothness; for generalization, paired interfaces use exact-pair holdout and scalar interfaces use both-axis value holdout; precision compares discrete and continuous coordinate MAE.

	Numerical embeddings			*Coordinate generalization*		*Coordinate precision*
Method	MAE ↓	R^2 ↑	Smooth. ↓	Seen MAE ↓	Held-out MAE ↓	MAE ↓
Paired interfaces						
Paired disc.	6.22	.980	.77	**4.62**	6.21	6.30
Paired cont.	**2.08**	**.998**	**.53**	4.84	**4.83**	**6.28**
Scalar interfaces						
Scalar disc.	3.02	.993	1.04	**2.49**	30.11	5.09
Scalar cont.	**1.19**	**.999**	**.66**	3.61	**3.79**	**5.07**

In Figure 4, the trained continuous embeddings have broad central peaks that fade with displacement, replacing much of the periodic structure visible at initialization. Paired discrete embeddings have a narrow central peak, while scalar discrete embeddings retain thin bands where one coordinate is unchanged. The lower probe error and smoother adjacent steps in Table 3 support this reading and help explain why continuous interfaces generalize to held-out coordinates through nearby values rather than separately learned tokens. Dimensionality statistics are given in Appendix A.12.

5.4 Coordinate generalization

We test how accurately each interface predicts coordinates withheld from training, using teacher-forced MAE. For paired interfaces, the held-out item is an exact (x, y) pair, although its individual x and y values may appear in other pairs. For scalar interfaces, we withhold coordinate values from both the x and y axes. Because these protocols hold out different items, we compare MAE only between discrete and continuous variants within each interface. We omit the standard interface because its tokenizer has no dedicated coordinate token to hold out.

Continuous interfaces maintain similar error on held-out coordinate units, while discrete interfaces show larger errors, especially under scalar-value holdout. These teacher-forced results diagnose coordinate generalization rather than complete-SVG generation.

5.5 Coordinate precision

We measure coordinate precision on a controlled fractional-coordinate benchmark under teacher forcing. For paired and scalar interfaces, we compare continuous coordinate predictions with their discrete counterparts on the same examples and teacher-forced contexts. This tests whether continuous prediction improves coordinate accuracy beyond the corresponding discrete representation. Continuous prediction lowers coordinate MAE for both paired and scalar interfaces, and both differences are statistically significant under a paired source-level bootstrap. These teacher-forced results measure coordinate precision rather than rendered fidelity.

5.6 Design ablations

Choice of numerical encoding. Smoothness alone does not rank the encoders: raw, RBF, and spline features also form broad neighborhoods in Figure 4, yet Fourier gives the best reconstruction quality (Table 4).

Table 4: Encoding ablation with matched training. Each encoding is followed by an MLP; RBF denotes radial basis functions.

Encoder	DINOv3 (%) ↑	SSIM (%) ↑	LPIPS ↓	MSE ↓
Fourier	**73.6**	**76.9**	**0.25**	**0.15**
Raw	70.0	74.6	0.28	0.18
RBF	73.3	76.1	0.25	0.16
Axis spline	70.9	75.3	0.27	0.17
Multi-resolution grid	69.1	74.6	0.28	0.18

Output objective. The depth-matched categorical control predicts coordinates more accurately under teacher forcing, while regression better predicts ordinary tokens and path commands and completes more SVGs. Thus improving the next coordinate alone does not ensure a better complete drawing. Appendix A.9 separates coordinate prediction from SVG-structure prediction.

6 Limitations

Following OmniSVG's pre-processing convention (Yang et al., 2025), both discrete and continuous interfaces use a shared coordinate scale M. Unscaled coordinates performed substantially worse, so applying the current encoder outside this scale requires rescaling.

The value decoder produces a point estimate, not an explicit distribution over possible geometries. We also tried a Gaussian-mixture decoder, which performed worse than MLP regression (Appendix A.11).

Our custom value modules are not yet integrated with optimized inference systems such as vLLM (Kwon et al., 2023), so generation cannot fully use their batching and kernel optimizations.

The interface is designed for supervised SVG generation under the grammar used here. Applying it to other structured vector domains, such as CAD, would require adapting the representation and pre-processing pipeline.

7 Conclusion

Our continuous approach gives an autoregressive SVG model a shared value encoder and decoder while retaining token prediction for drawing structure. Across matched raster-to-SVG comparisons, the continuous interfaces attain the strongest results. The learned continuous embeddings are smoother and more linearly decodable than discrete embeddings. In coordinate-generalization tests, continuous interfaces retain low error on held-out coordinates, whereas the discrete interfaces degrade more sharply. In the coordinate-precision tests, continuous prediction produces a statistically significant reduction in teacher-forced coordinate MAE relative to the corresponding discrete interface. Beyond its empirical gains, the continuous approach offers a more natural representation of SVG geometry, preserving its numerical structure rather than reducing coordinates to categorical symbols.

AI Use Statement

We designed and conducted the experiments, assessed the claims, and wrote the manuscript. Generative AI tools assisted with code generation, experiment orchestration, and polishing our writing. We verified that the code and experiments were correct and take full responsibility for the work.

Reproducibility Statement

Section 3 specifies the numerical interface, training objective, and autoregressive feedback procedure. Section 4 describes the comparison protocols, and Appendix A provides implementation settings,

evaluation definitions, coordinate-holdout construction, and probe procedures. We retain training configurations and per-example evaluation outputs for reproducibility.

REFERENCES

Defu Cao, Zhaowen Wang, Jose Echevarria, and Yan Liu. SVGformer: Representation learning for continuous vector graphics using transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10093–10102, June 2023.

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, 2020.

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Parker, Bruno Régaldo-Saint Blancard, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. xVal: A continuous numerical tokenization for scientific language models, 2024. URL https://arxiv.org/abs/2310.02989.

Ajay Jain, Amber Xie, and Pieter Abbeel. VectorFusion: Text-to-SVG by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1911–1920, June 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

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)*, October 2019.

Sagi Polaczek, Yuval Alaluf, Elad Richardson, Yael Vinker, and Daniel Cohen-Or. NeuralSVG: An implicit representation for text-to-vector generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15458–15468, October 2025.

Pradyumna Reddy, Michaël Gharbi, Michal Lukáč, and Niloy J. Mitra. Im2Vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. URL https://arxiv.org/abs/2102.02798.

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.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, August 2016.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7537–7547, 2020.

Feiyu Wang, Zhiyuan Zhao, Yuandong Liu, Da Zhang, Junyu Gao, Hao Sun, and Xuelong Li. SVGen: Interpretable vector graphics generation with large language models. In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, pp. 9608–9617. ACM, October 2025. doi: 10.1145/3746027.3755011. URL http://dx.doi.org/10.1145/3746027.3755011.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution, 2024. URL https://arxiv.org/abs/2409.12191.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.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.

Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. IconShop: Text-guided vector icon synthesis with autoregressive transformers. *ACM Transactions on Graphics*, 42(6), 2023. doi: 10.1145/3618364.

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, 2025. doi: 10.1109/CVPR52734.2025.02206. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wu_Chat2SVG_Vector_Graphics_Generation_with_Large_Language_Models_and_Image_CVPR_2025_paper.html.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016. URL https://arxiv.org/abs/1609.08144.

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.

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.

Ximing Xing, Ziteng Xue, Zhenxi Li, Weicong Liang, Linqing Wang, Zhantao Yang, Tiankai Hang, Zijin Yin, Qinglin Lu, Chunyu Wang, and Qian Yu. Hierarchical SVG tokenization: Learning compact visual programs for scalable vector graphics modeling, 2026.

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. In *Advances in Neural Information Processing Systems*, 2025.

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 (CVPR)*, June 2018.

Tianyi Zhou, Deqing Fu, Mahdi Soltanolkotabi, Robin Jia, and Vatsal Sharan. FoNE: Precise single-token number embeddings via Fourier features. In *International Conference on Learning Representations (ICLR)*, 2026.

A Additional experimental detail

A.1 Scalar training and inference diagrams

The scalar interface follows the training and inference flow in Figures 1b and 2, but replaces each coordinate pair in Figure 1a with two consecutive [NUM] markers carrying x and y separately. Each

scalar prediction therefore occupies one autoregressive step. Table 5 shows the corresponding scalar pre-processing for the start of the same SVG path.

Table 5: Scalar pre-processing sends tokens to the token encoder and numbers to the value encoder, attaching x and y to consecutive `[NUM]` markers.

SVG Text	M	56	25	L	174	25	L	143	75	L	174	124	···
Tokens	M	[NUM]	[NUM]	L	[NUM]	[NUM]	L	[NUM]	[NUM]	L	[NUM]	[NUM]	···
Numbers	∅	56	25	∅	174	25	∅	143	75	∅	174	124	···

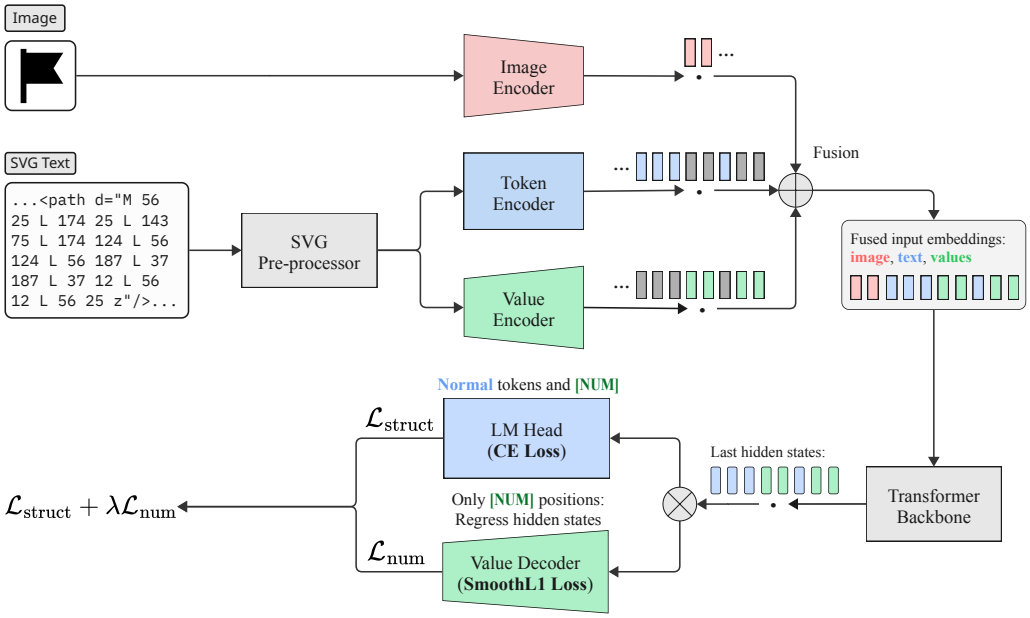


Figure 5: Scalar training encodes one value at each `[NUM]` marker and supervises its prediction through the value decoder.

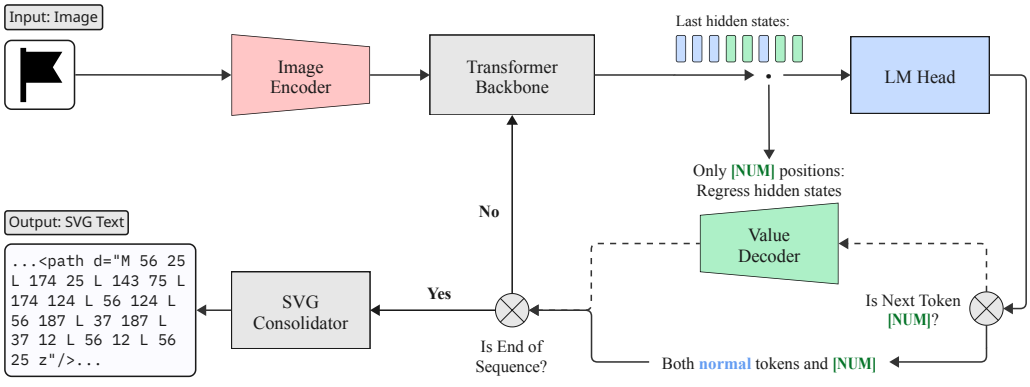


Figure 6: At inference, each `[NUM]` marker triggers one scalar prediction before the next autoregressive step.

A.2 Implementation and Training

Coordinate serialization. The principal generation comparisons use the same canonical SVG geometry on integer coordinates from 0 to M. The standard interface serializes integer values with the backbone tokenizer, scalar discrete uses a shared coordinate vocabulary across axes, and

paired discrete maps (x, y) to the index $x + (M + 1)y$. The continuous approach replaces the same coordinate positions with markers and attaches either one scalar or a pair of values. The remaining command and markup tokens are retained. All interfaces use the same canonical target geometry. Coordinate precision is evaluated using separately trained models with fractional targets.

Training configuration. Table 6 summarizes the training setup and continuous value modules.

Table 6: Training configuration.

Component	Setting	Configuration
Training		
Training	Epochs	1
Training	GPUs	4 L40S
Training	Per-device batch size	4
Training	Gradient accumulation	2 steps
Training	Effective global batch size	32
Training	Maximum sequence length	4,096
Optimization		
Optimization	Optimizer	AdamW
Optimization	Learning rate	3×10^{-5}
Optimization	Learning-rate schedule	Cosine decay
Optimization	Warmup	5% of training steps
Optimization	Weight decay	0
Optimization	Gradient clipping	Maximum norm 1
Coordinate-input noise		
Noise	Distribution	Zero-mean Gaussian perturbations to input coordinates
Noise	Schedule	Standard deviation decays linearly from 15 to 5 coordinate units over training
Continuous value modules		
Value encoder	Fourier bands	$K = 7$ per axis
Value encoder	Input shape	Scalar: $1 + 2K$; paired: $2 + 4K$ features
Value encoder	MLP shape and activation	Input $\to d_{\text{model}} \to d_{\text{model}} \to d_{\text{model}} \to d_{\text{model}} \to d_{\text{model}}$; GELU after the first four linear layers
Value decoder	MLP shape and activation	$d_{\text{model}} \to d_{\text{model}} \to d_{\text{model}} \to d_{\text{model}} \to d_{\text{model}} \to 1$ (scalar) or 2 (paired); GELU after the first four linear layers

Here, d_{model} denotes the transformer width.

Numerical computation. The value encoder, decoder, loss, and feedback values use FP32 while the transformer uses BF16. Teacher-forced evaluation uses the same coordinate serialization as training, including coordinate separators and scalar-marker conventions.

A.3 Text-to-SVG evaluation

Text-to-SVG generation uses caption conditioning and is evaluated without candidate selection against a reference image. Table 7 reports the full metric results; the main table includes DINOv3.

Table 7: Text-to-SVG results on caption-conditioned generations.

Method	DINOv3 (%) ↑	SSIM (%) ↑	LPIPS ↓	MSE ↓
Standard	71.6	71.9	0.30	0.19
Paired disc.	69.6	70.3	0.31	0.21
Scalar disc.	73.1	69.9	0.31	0.21
Paired cont.	75.7	75.7	0.28	0.17
Scalar cont.	**77.4**	**75.8**	**0.28**	**0.17**

Scalar continuous achieves the strongest score on each of the four image-quality metrics.

A.4 Evaluation and uncertainty

Denominators and decoding. Generation quality is evaluated on all 5.71k test examples, including failed and unfinished outputs. The main evaluation uses the five temperatures in Section 4.3, top-

$p = 0.9$, and a maximum of 4,096 generated tokens. Ablations use temperature 0.7, top-$p = 0.95$, and top-$k = 5$. Native validity requires successful termination and a renderable SVG without repair. Post-repair metrics evaluate every output after truncation and repair.

Rendering and image metrics. For transparent backgrounds, we compute each image-quality metric on white- and black-background renderings of the prediction and reference and report their mean. For DINOv3, prediction and target SVG are rendered to 512×512 canvases on each background and passed through the stock image processor. We mean-pool the final-layer tokens and report $(1 + \cos)/2$, following the StarVector pooling and display convention with a DINOv3 backbone (Rodriguez et al., 2025). Strict DINOv3 assigns zero to an invalid or unfinished output; it is distinct from the post-repair similarity in the main generation tables. For SSIM and pixel error, the reference is the pre-processed input raster on the corresponding background. The prediction is rendered in its declared viewport with longest side 1,024 and resized with Lanczos to the reference raster dimensions. LPIPS uses the aligned pair letterboxed onto a 512×512 canvas on each background. Pixel distances are computed on normalized RGB values; they do not receive artificial zero scores for failed generations.

Uncertainty. For generation metrics, paired bootstrap intervals resample prompts jointly across methods, keeping all seeds for each prompt together. Coordinate-level analyses resample all occurrences from each source SVG together; the fractional-coordinate intervals use 10,000 paired resamples of the 96 source icons. These intervals describe variability over evaluation inputs, not variability from retraining a model.

Validity and repair. Failed generations are truncated to a renderable prefix before image-quality evaluation. When several prefixes are renderable, recovery selects the prefix with the highest reference-image SSIM.

A.5 Coordinate-holdout construction

Exact pairs versus scalar values. An exact-pair holdout removes documents containing selected (x, y) pairs while allowing each component value to appear in other combinations. The pair-selection procedure divides the interior coordinate range into a 4×4 grid and selects one sufficiently frequent pair per cell, avoiding repeated x or y values within a set and repeated pairs across sets. We select frequent interior pairs using a fixed random procedure. Whole-document exclusion prevents a withheld target from reappearing elsewhere in the same training SVG. Paired-interface results are averaged over three exact-pair holdout sets.

Role-value design. All four interfaces are trained without coordinate noise on size-matched 5% subsets. Three role-value sets use the interior values 37, 103, and 169. The one-axis exclusions cycle through $(x = 37, y = 103)$, $(x = 103, y = 169)$, and $(x = 169, y = 37)$, leaving each value available in the opposite role. One-axis tests balance targets from the x and y roles, while both-axis tests balance occurrences in x, y, and both coordinates. Training and evaluation targets are disjoint across all holdout sets. These interventions address different questions: composition of familiar values, transfer across roles, and prediction of an individually unseen value. The main scalar-interface comparison uses the both-axis intervention because an individual coordinate value is the scalar representation's atomic output unit.

A.6 Representation probes

Static geometry. Paired probes use the input embedding of each of the $(M + 1)^2$ coordinate pairs; scalar probes use the $M + 1$ individual numerical embeddings. The linear-probe splits hold out residues 0, 3, and 7 separately under $(17x + 31y) \bmod 10$ for pairs and $v \bmod 10$ for scalar values. Centered ridge regression uses a fixed penalty of 10^{-4} without selecting hyperparameters on the held-out coordinates. Finite differences measure local changes in the numerical representation, while covariance eigenvalues determine participation ratio and explained-variance dimensions. Before probing or measuring smoothness, we subtract the mean embedding and normalize each embedding to unit length. Scalar smoothness is the mean second-difference norm, $\|e(v + 1) - 2e(v) + e(v - 1)\|_2$, over interior values; lower values indicate less abrupt changes between successive embedding steps.

For paired embeddings, we compute the same second difference along each coordinate axis and average the two axis means.

Displacement figures. For each method, the grid-wide mean embedding is subtracted and each coordinate vector is normalized before cosine similarity is computed. Each displacement cell averages all coordinate pairs that fit within the grid; all learned-embedding panels share one color scale. The initialization panels show the numerical representations before training. The fixed Fourier basis is not itself trained; changes in the full numerical embedding arise through learned modules.

A.7 Coordinate precision

Fractional-coordinate analysis. We compare continuous and discrete predictions on the same fractional-coordinate benchmark under teacher forcing. Source-level bootstrap 95% intervals for the continuous-minus-discrete MAE difference are $[-0.022, -0.015]$ for paired interfaces and $[-0.031, -0.023]$ for scalar interfaces; both exclude zero.

A.8 Pre-processing ablation for the standard interface

We compare targets from the standard interface using a 512-pixel canvas, three-decimal coordinates on the shared canvas, and integer coordinates on the shared canvas. The integer-quantized variant gave the strongest reconstruction fidelity in this comparison and supplies the standard baseline.

Table 8: Coordinate pre-processing for the standard interface. Image metrics include outputs that completed without repair. Because the 512-pixel condition uses a different training subset, comparisons with that condition do not isolate coordinate serialization.

Pre-processing	DINOv3 (%) ↑	SSIM (%) ↑
512-pixel canvas	81.8	74.3
Shared canvas, three decimals	87.5	77.4
Shared canvas, integers	93.8	84.1

A.9 Output-objective control

The depth-matched categorical control replaces the paired regressor with two $(M+1)$-way categorical outputs while retaining the encoder, coordinate grouping, depth, and training budget. Classification reduces coordinate MAE from 5.381 to 5.181, whereas regression improves ordinary-token accuracy from 89.75% to 90.65% and path-command accuracy from 85.80% to 86.90%. On 5% of the training data, regression attains 44.8% native completion and 38.9% strict DINOv3, compared with 35.0% and 31.8% for classification. Thus lower one-step coordinate error does not ensure better SVG-structure prediction or completion.

A.10 Coordinate noise

On 5% of the training data, coordinate-input noise raises paired discrete native completion from 13.2% to 30.5%. Under the same noise schedule, paired continuous reaches 44.8% native completion.

A.11 Coordinate-decoder ablation

Table 9: Coordinate-decoder ablation. The table reports post-repair reconstruction quality, and the Gaussian-mixture model (GMM) uses deterministic mean decoding.

Decoder	DINOv3 (%) ↑	SSIM (%) ↑
MLP regression	**65.8**	**75.2**
GMM regression	60.4	72.9

A.12 Additional representation and holdout statistics

Tables 10 and 11 report additional representation and coordinate-holdout statistics.

Table 10: Static paired-coordinate representation geometry. Coordinate MAE and ridge R^2 measure numerical decodability. Participation ratio (PR) and 95% variance dimension describe the spread of embeddings.

Interface	PR	95% dims	Ridge R^2	MAE
Paired disc.	50.2	639	.980	6.22
Paired cont.	10.1	24	.998	2.08

Table 11: Seen versus held-out teacher-forced performance, averaged over three holdout sets. Paired interfaces use exact-pair holdout, scalar interfaces use both-axis value holdout, and MAE is reported in coordinate units.

Interface	Seen MAE	Held-out MAE
Paired interfaces		
Paired disc.	4.62	6.21
Paired cont.	4.84	4.83
Scalar interfaces		
Scalar disc.	2.49	30.11
Scalar cont.	3.61	3.79

A.13 Additional qualitative examples



Figure 7: Additional examples from MMSVGBench and the raster-to-SVG test set.

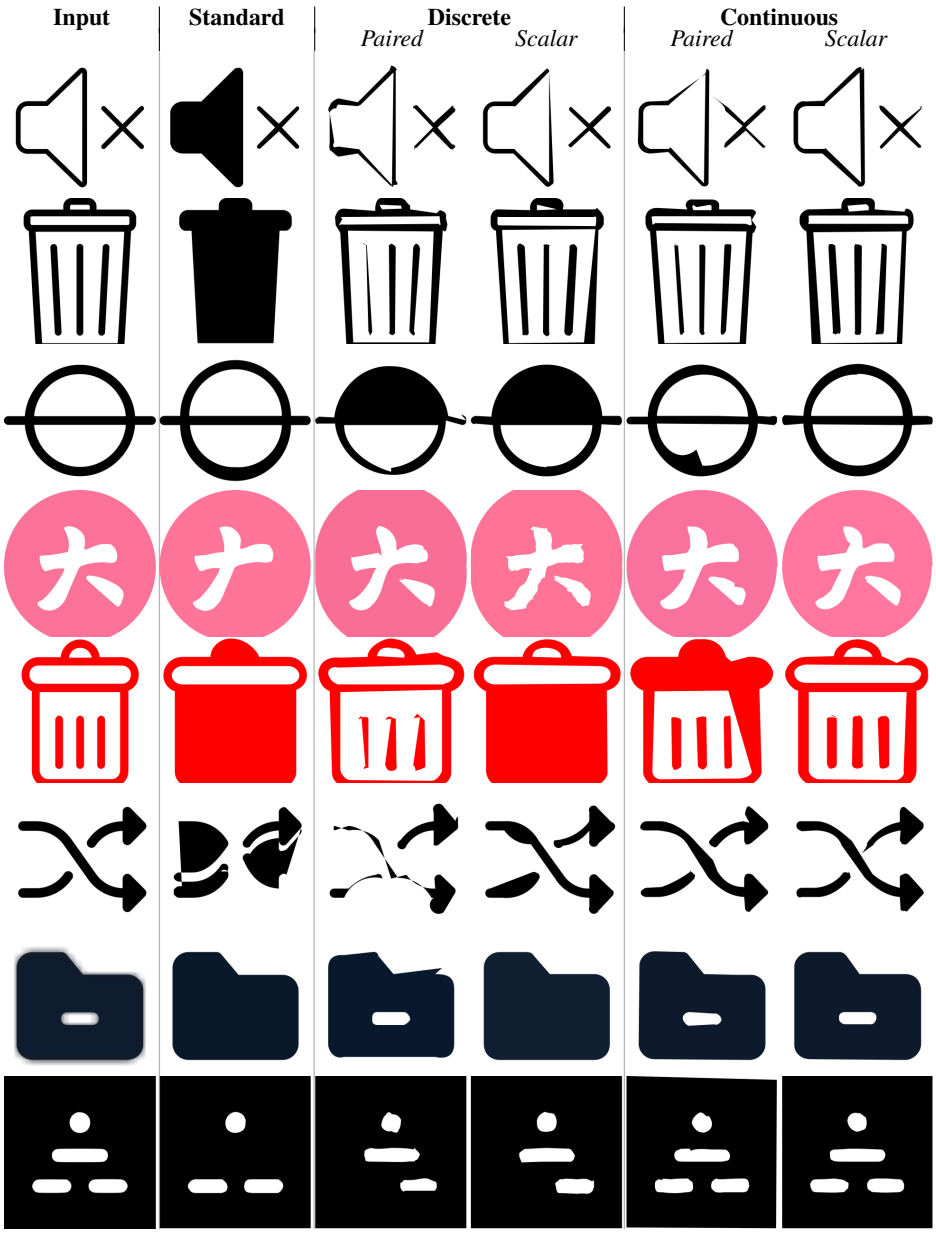


Figure 8: Additional raster-to-SVG test examples.

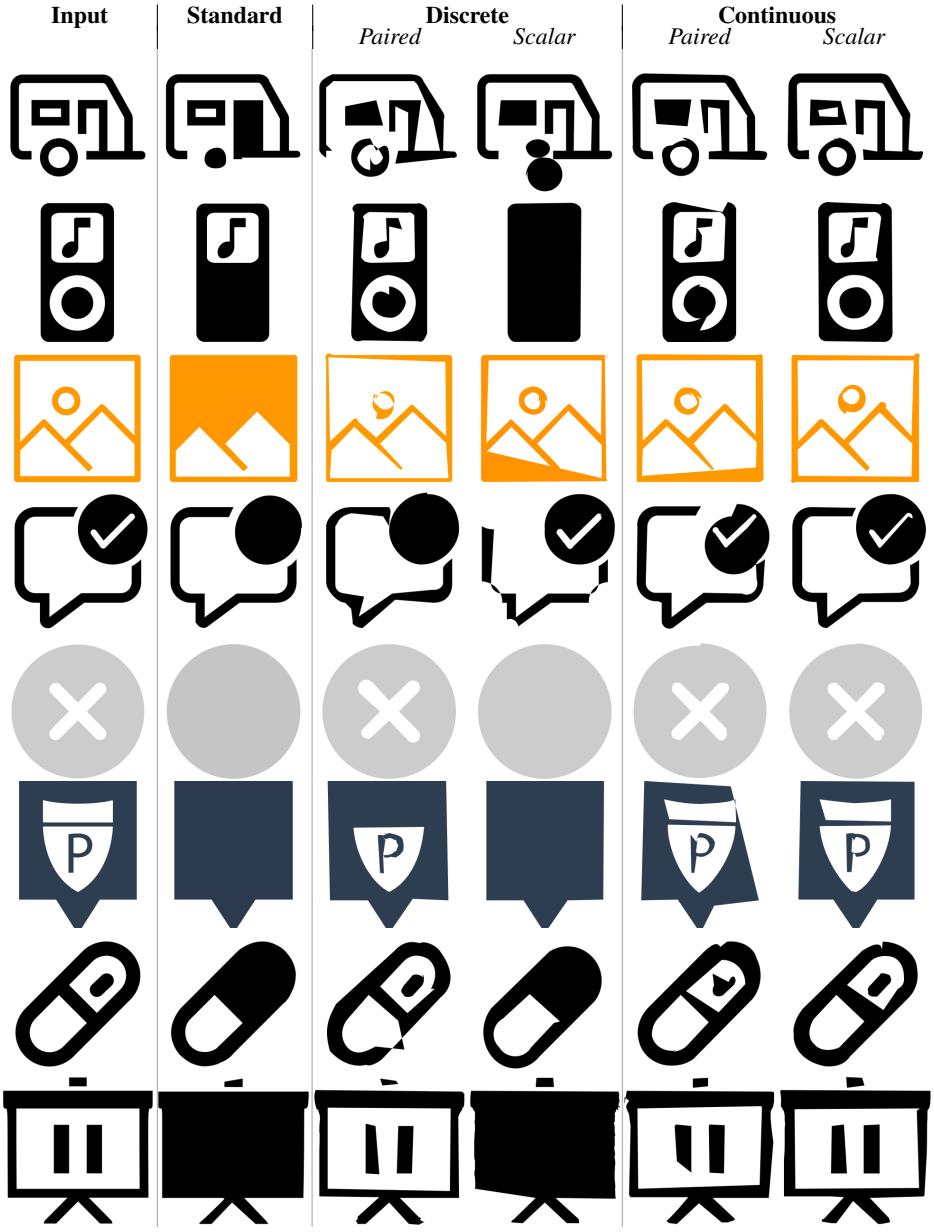


Figure 9: Additional raster-to-SVG test examples.

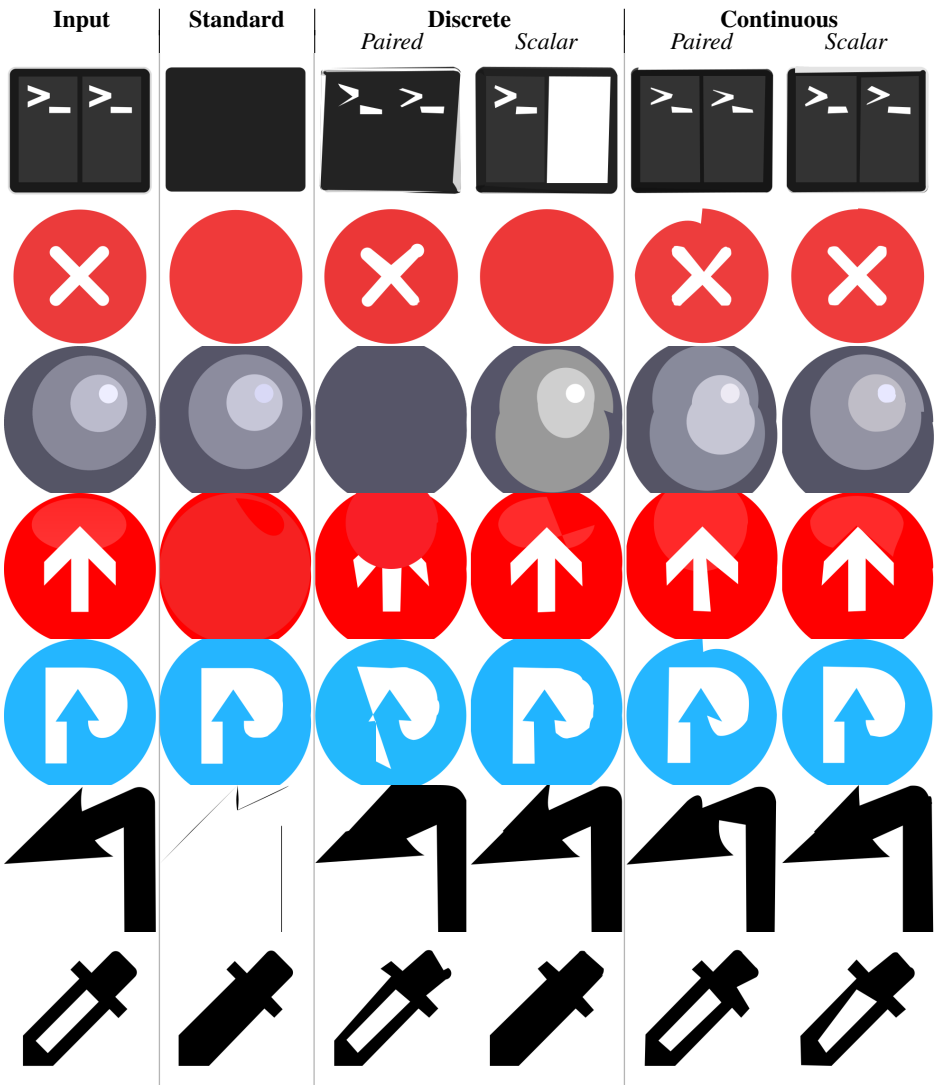


Figure 10: Additional raster-to-SVG test examples.