A Scalable Vector Graphics Latent Space

Leonardo Zini[1], Elia Frigieri[1], and Lorenzo Baraldi[1]

University of Modena and Reggio Emilia, Italy
{name.surname}@unimore.it
aimagelab.github.io/svg_latent_space/

Abstract. Scalable Vector Graphics are a fundamental medium for resolution-independent visual content, yet the deep learning community lacks a continuous, dense, and invertible latent space for vector representations, the kind of foundational building block that Variational Autoencoders and their descendants have long provided for raster images. We introduce **SLS** (**S**VG **L**atent **S**pace), a Transformer-based autoencoder that learns compact dense representations of individual SVG paths, the atomic visual elements from which any SVG image can be composed. By modeling SVG commands, coordinate data, and visual properties within a unified BPE-based token vocabulary, SLS learns fixed-size latent representations that jointly capture structure and appearance, and can be decoded back into valid, style-consistent SVG paths with high fidelity. The resulting embedding space is robust, invertible, and structured: embeddings lie on a unit hypersphere, enabling efficient similarity search, composition, and downstream conditioning through simple vector-space operations. Finally, we demonstrate that SLS generalizes across diverse tasks reducing their FLOPs by over 150× compared to token-based approaches, and establishing a general-purpose latent foundation for vector graphics research.

Keywords: Scalable Vector Graphics · Latent Space · Representation Learning · Transformer Autoencoder · Vector Graphics Representation

1 Introduction

Scalable Vector Graphics have become a foundational medium for resolution-independent visual content, spanning iconography, UI design, data visualization, and beyond. Unlike raster images, SVGs describe geometry symbolically through sequences of drawing commands, coordinates, and style attributes – a representation that is inherently interpretable and editable, yet poses a fundamental challenge for deep learning: how do we build a continuous, compact, and semantically meaningful space over vector content?

This question has remained largely unanswered. Existing approaches either rely on rasterized renderings [13, 14, 36], indirectly learning representations in the pixel domain and thus discarding the underlying geometric structure, or process SVG markup as raw text sequences. The latter strategy, adopted by a

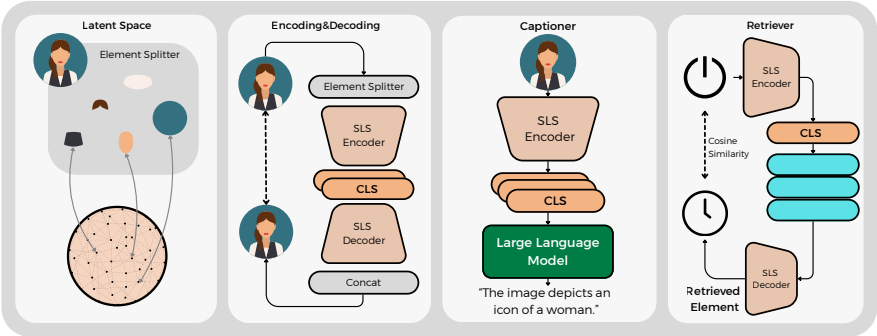

Fig. 1: Overview of the proposed approach: from left to right, we show *(i)* SLS path-level latent space, *(ii)* the encoding and decoding framework that learns structured dense representations, and *(iii)* and *(iv)* downstream applications including captioning with a large language model and element-level retrieval via similarity matching.

growing family of LLM-based methods [35, 37, 40, 45], directly generates or consumes SVG code, but does not learn a continuous and compact latent space over vector content. Within these approaches, SVG elements are serialized into long token sequences, which leads to sparse, high-dimensional representations that strain context windows and complicate downstream tasks such as retrieval or captioning at scale. The reliance on rasterization or long token sequences is not incidental: it reflects the lack of a sufficiently expressive and scalable invertible embedding space tailored to vector content. Without a compact and semantically structured space, models must either discard symbolic structure by operating in the pixel domain or process SVG markup as sparse text sequences, both of which hinder scalability and downstream reasoning. While the broader vision community has long benefited from latent embedding spaces for images, enabling retrieval, captioning, and generation through simple vector-space operations, an equivalent continuous and semantically structured embedding space for SVG content remains underexplored. Existing approaches to SVG representation learning introduce latent spaces that are constrained in terms of embedding fidelity, input versatility, and stylistic coverage.

We argue that the missing ingredient is a *path-level latent space*: a continuous embedding space where each individual SVG path is mapped to a single dense vector. SVG paths provide a canonical representation for vector graphics, since drawable primitives (*e.g.*, circles, rectangles, ellipses and polygons) can be losslessly converted into path descriptions. As a result, path-level representations form a complete and unified abstraction for SVG content. The analogy to the raster image world is instructive: the introduction of continuous latent spaces for images, most notably through Variational Autoencoders [15] and their descendants [8, 27], was a watershed moment that unlocked generation, interpolation, retrieval, and downstream conditioning at scale [2, 24, 29, 39]. Existing SVG dense encoders [5, 28, 34] are strongly limited in both quality of learned space and sequence representation. Alternatively, traditional raster encoders [21, 23] operate RGB space, discarding the symbolic structure that makes SVGs editable and

resolution-independent, and, in the case of raster VAEs, cannot reconstruct the original code from the latent representation, producing features with no path back to valid SVG markup. A path-level latent space should reduce sequence representations from thousands of tokens to a handful of embeddings, preserve both geometric structure and style attributes in a form amenable to vector-space reasoning, and critically decouple the complexity of downstream applications from the length of the underlying SVG markup. Equally important, the latent space must be *invertible*: a dense embedding should be sufficient to faithfully reconstruct the original SVG path, preserving both geometry and style with high fidelity – a property that any useful vector-domain latent space must provide.

To this end, we introduce **SLS** (**S**VG **L**atent **S**pace), a Transformer-based autoencoder that learns compact latent representations of individual SVG paths (Figure 1). A key design choice is the adoption of a data-driven BPE tokenizer trained directly on path corpora, which treats commands, coordinates, and style attributes, such as fill color, stroke width, and opacity – as a unified token vocabulary. Rather than enforcing rigid command-based boundaries [5, 34, 35], BPE discovers sub-word units whose boundaries emerge from corpus statistics, naturally accommodating the heterogeneous mix of geometric and stylistic information that rigid tokenization schemes tend to handle poorly or omit entirely. SLS maps the input sequences to dense vectors via a transformer-based encoder, while a lightweight autoregressive decoder reconstructs the original path from the latent vector with high fidelity. This invertibility is a defining property of SLS: the same dense embedding that enables efficient vector-space reasoning can be decoded back into a valid, style-complete SVG path, something raster encoders such as CLIP [23] or DINOv2 [21] fundamentally cannot do. SLS operates on single paths rather than full SVG files, reducing average sequence lengths by over two orders of magnitude compared to whole-image LLM-based methods. Although not explicitly enforced during training, the encoder outputs consistently exhibit a nearly constant ℓ_2-norm, resulting in latent representations that lie close to a hypersphere of stable radius. We exploit this property by normalizing embeddings at inference time, enabling efficient cosine-similarity search and composition of path embeddings into image-level representations.

We validate SLS on two downstream tasks and analyze the structure of the learned latent space. For SVG image captioning, path embeddings are projected into the token space of pre-trained language models, achieving strong performance across three LLM backbones while reducing training FLOPs by over 150×. For path retrieval, SLS enables scalable nearest-neighbor search over hundreds of thousands of paths, outperforming raster-based encoders and prior SVG-specific baselines. Finally, robustness analysis shows that the learned latent space remains stable under both Gaussian noise and angular perturbations.

The main contributions of this work can be summarized as follows:

- We introduce SLS, a scalable and invertible path-level latent space for SVG content that overcomes key limitations of existing approaches, which either lack embedding quality, restrict input flexibility, or omit stylistic attributes.

- We show that a data-driven BPE tokenizer applied uniformly to commands, coordinates, and style attributes enables richer and more flexible path representations than rigid command-based alternatives, capturing stylistic variation that prior approaches discard or handle separately.
- We demonstrate that SLS learns an invertible latent space that lies on a hypersphere: a single dense vector per path is sufficient to reconstruct the original SVG with high geometric and stylistic fidelity, a property that raster encoders cannot provide.
- We show that SLS embeddings generalize across diverse downstream tasks, such as retrieval, captioning, and embedding space analysis, through simple vector-space operations, without task-specific architectural changes.

2 Related Works

2.1 SVG Representation

Representing scalable vector graphics remains a central challenge in Computer Vision. Approaches like DeepSVG [5] introduce hierarchical formulations with fixed per-path control points but omit style attributes. While these methods simplify training and enable structured editing, they sacrifice representational flexibility due to their rigid parameterizations. Beyond methods that explicitly model SVGs, a growing body of work demonstrates that neural models trained on tasks involving vector graphics often learn implicit SVG-like representations. For instance, image-to-sequence or sketch generation models, although not constrained to produce valid SVGs, capture underlying geometric and structural patterns that closely resemble vector representations [10].

Image-to-Vector. Another line of work focuses on recovering vector graphics from raster images. Im2Vec [25] predicts parametric primitives to approximate shapes, while VectorGrimoire [7] improves geometric fidelity through richer curve modeling. Other methods, such as DualVector [19], DeepVecFont [20,34] leverage both image and vector features, focusing specifically on neural representations for font reconstruction and sketch vectorization. These approaches perform well in specialized domains but struggle to generalize to arbitrary SVG content. This also suggests that SVG-like abstractions emerge when models are required to reason about compositional, structured visual content, highlighting the pervasiveness and utility of vector representations in modern deep learning pipelines.

Large Language Model-based Methods. Recent text-to-vector models (*e.g.*, IconShop [35], vHector [45], LLM4SVG [37], OmniSVG [40], SVGen [33]) and DuetSVG [41] directly serialize entire SVG images as long token sequences and generate them via large language models. While effective for text-conditioned synthesis, these token-based approaches suffer from significant scalability limitations: even moderately complex graphics require hundreds to thousands of tokens, resulting in sparse, high-dimensional representations that strain context windows and complicate learning. Rodriguez *et al.* [26] extend this paradigm to image-to-SVG vectorization by treating SVG markup as text sequences, further

demonstrating the potential of language models for vector graphics generation. However, the context length problem remains acute – their approach requires even longer sequences for detailed images, making the representation increasingly sparse and computationally expensive. In contrast, our method addresses this fundamental limitation through compact path-level embeddings that encode geometric and stylistic information in a fixed-size dense representation.

Hybrid Approaches. Other works explore intermediate strategies. SVGFusion [38] combines fixed symbolic representations with learnable latent matrices to improve scalability, though it still relies on predefined structural constraints. SuperSVG [13] learns SVG representations of superpixel image regions for RGB vectorization, operating at a different granularity than whole-image methods. Other work, such as NeuralSVG [28], factorizes geometry and color while constraining each path to a fixed number of points.

2.2 Token-based Encoder-Decoder Architectures

Sequence-to-sequence architectures have proven effective for learning dense representations across diverse modalities. In machine translation, NLLB [30] and SONAR [22] construct unified multilingual embedding spaces that enable cross-lingual transfer through shared semantic representations. Similarly, LCM (Large Concept Models) [16] demonstrate that next-embedding prediction can learn structured representations for complex domains. Inspired by these successes, our work extends token-based embedding learning to the SVG domain. Unlike prior LLM-based methods that treat entire SVG images as sparse token sequences, we construct a dense path-level embedding space that captures both geometric and stylistic properties in a compact representation, effectively addressing the context length limitations of sequential approaches.

3 Method

3.1 Preliminaries

We address the problem of learning compact vector representations of SVG paths, which are sequences of discrete drawing commands and associated parameters. We train a path-specific tokenizer that converts SVG elements into discrete tokens encompassing command types, numeric arguments, and stylistic attributes. To standardize inputs and reduce average sequence length, we adopt the preprocessing procedure introduced in [45]. The tokenizer operates at the individual path level, with compound SVG objects decomposed into independent path sequences. Unlike prior work [5, 42], which relies on specialized geometric encodings, we represent SVG paths as text sequences and train a purely text-based tokenizer that uniformly processes structural and stylistic information. Our tokenizer is based on Byte Pair Encoding (BPE) and is trained on the preprocessed dataset to learn a compact, subword-level vocabulary tailored to this domain. This reduces average sequence length while preserving geometric

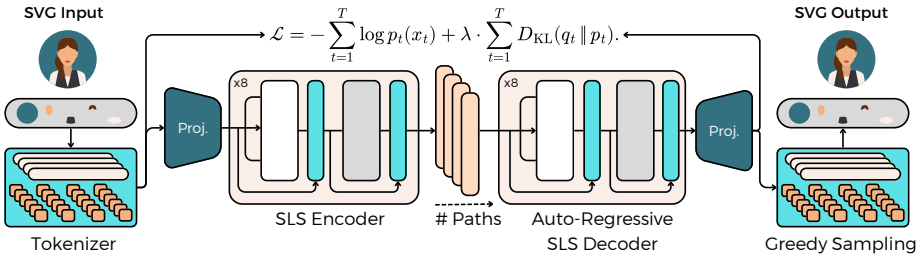

Fig. 2: Overview of SLS architecture. SVG paths are encoded into latent representations, with Gaussian noise injection during training for regularization. The autoregressive decoder reconstructs paths through greedy sampling decoding. Embeddings on the unit hypersphere enable efficient downstream applications: similarity-based path retrieval and image captioning via learned projection into language model token space.

fidelity. Given the resulting token sequence $\mathbf{x} = (x_1, \ldots, x_T)$, our goal is to learn an encoder that maps $\mathbf{x}$ to a dense latent representation, and a decoder that reconstructs the original sequence:

$$\mathcal{E}(\mathbf{x}) = \mathbf{z}, \quad \mathcal{D}(\mathbf{z}) \approx \mathbf{x}, \quad \mathbf{x} \in \mathcal{V}^T, \quad \mathbf{z} \in \mathbb{R}^d. \tag{1}$$

The encoder–decoder architecture follows the standard autoencoder framework for learning compressed representations, enabling downstream applications that benefit from latent vector-space reasoning over structured path data while reducing sparsity. This approach yields SLS, a SVG autoencoder architecture capable of producing dense representations of complete SVG images and paths while preserving style attributes and managing longer context windows.

3.2 SLS Architecture

Our proposed approach adopts a Transformer-based encoder-decoder architecture. The overall architecture is illustrated in Figure 2.

Encoder. Each SVG path sequence is tokenized and explicitly delimited by special `<s>` (BOS) and `</s>` (EOS) tokens, which are part of the vocabulary. Given an input sequence:

$$\mathbf{x} = [\texttt{<s>}, x_1, \ldots, x_T, \texttt{</s>}], \tag{2}$$

tokens are mapped to learned embeddings and combined with sinusoidal positional encodings. The sequence is processed by a stack of Transformer encoder layers. Instead of prepending a learnable [CLS] token, we use the final-layer hidden state corresponding to the EOS token as the global sequence representation. Formally, if $\mathbf{H} \in \mathbb{R}^{(T+2)\times d}$ denotes the encoder output, where T is the sequence length and d is the size of the model, the latent embedding is defined as:

$$\mathbf{z} = \mathbf{H}_{\mathrm{EOS}}, \tag{3}$$

where $\mathbf{H}_{\mathrm{EOS}} \in \mathbb{R}^d$ is the hidden state at the EOS position. This representation serves as a dense embedding summarizing the entire input path.

Decoder. The decoder reconstructs the original token sequence conditioned on the latent embedding $\mathbf{z} \in \mathbb{R}^d$, which matches the decoder model dimension. We inject $\mathbf{z}$ once as a prefix token and apply a causal Transformer over the autoregressive input. A linear language modeling head produces vocabulary logits at each timestep. Unlike natural language generation, where multiple valid continuations may exist and stochastic sampling strategies can improve diversity [9, 12], SVG path sequences are deterministic and syntactically strict: each token position admits a single correct value determined by the underlying geometric structure. Therefore, we employ greedy decoding, selecting the token with the highest probability at each step:

$$\hat{x}_t = \arg\max_{x \in \mathcal{V}} p(x \mid \mathbf{x}_{<t}, \mathbf{z}), \tag{4}$$

where $\mathcal{V}$ denotes the vocabulary. This deterministic strategy ensures syntactically valid path reconstructions and prevents malformed SVG generation.

Training. We train the autoencoder end-to-end using token reconstruction loss. To encourage a more robust and well-structured latent representation, we inject Gaussian noise into the latent embedding during training with standard deviation $\sigma = 1.0$, following [5]:

$$\mathbf{z}_{\text{train}} = \mathbf{z} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}). \tag{5}$$

Conditioned on the perturbed latent vector, the decoder autoregressively reconstructs the original token sequence. At each timestep, it produces vocabulary logits $\mathbf{y} \in \mathbb{R}^{T \times |\mathcal{V}|}$, which define the predicted token distribution:

$$p_t = \text{softmax}(\mathbf{y}_t). \tag{6}$$

The training objective combines the standard cross-entropy loss with a KL divergence term that enforces distributional consistency between the target and predicted token distributions. Specifically, we compute the KL divergence between the target distribution q_t associated with the ground-truth token and the predicted token distribution p_t:

$$\mathcal{L} = \underbrace{-\sum_{t=1}^{T} \log p_t(x_t)}_{\mathcal{L}_{\text{CE}}} + \lambda \cdot \underbrace{\sum_{t=1}^{T} D_{\text{KL}}(q_t \,\|\, p_t)}_{\mathcal{L}_{\text{KL}}}. \tag{7}$$

Here, λ corresponds to scaling factor of $\mathcal{L}_{\text{KL}}$. This regularizer stabilizes training by encouraging the predicted distribution to match a sharpened, self-generated target distribution.

Inference. At inference time, we observe that the trained encoder produces latent representations with consistent ℓ_2 norm with low standard deviation, indicating a well-regularized latent space. We leverage this property by normalizing the encoder output to unit norm for downstream applications, as

$$\mathbf{z}_{\text{norm}} = \frac{\mathbf{z}}{\|\mathbf{z}\|_2}. \tag{8}$$

This normalized representation $\mathbf{z}_{\text{norm}}$ enables efficient vector-space operations (*e.g.*, similarity search, interpolation) in downstream tasks, as all representations lie on the unit hypersphere, and further promotes numerical stability. Importantly, due to the intrinsic constant-norm structure of the learned latent space, this normalization does not discard meaningful magnitude information, thereby avoiding information loss. When reconstruction is required, we rescale by the empirical mean before feeding to the decoder:

$$\mathbf{z}_{\text{decoder}} = \mu_{\|\mathbf{z}\|} \cdot \mathbf{z}_{\text{norm}}, \tag{9}$$

where $\mu_{\|\mathbf{z}\|}$ indicates the mean ℓ_2 norm characteristic of the model. This decoupling between the normalized encoder output and the rescaled decoder input provides flexibility: downstream applications can reason with unit-norm vectors while the decoder receives representations in the expected magnitude range.

3.3 Downstream Applications

The normalized latent representations $\mathbf{z}_{\text{norm}} \in \mathbb{R}^d$ produced by our encoder enable various downstream applications that benefit from compact, semantically meaningful vector representations of SVG paths and images.

Path Retrieval. Given a query path encoded as $\mathbf{z}_q$, we retrieve the most similar paths from a database by computing cosine similarity in the normalized embedding space:

$$\text{sim}(\mathbf{z}_q, \mathbf{z}_i) = \frac{\mathbf{z}_q^\top \mathbf{z}_i}{\|\mathbf{z}_q\| \|\mathbf{z}_i\|}. \tag{10}$$

Since all embeddings lie on the unit hypersphere, cosine similarity reduces to the dot product enabling efficient retrieval with favoring nearest neighbor methods. This capability is particularly valuable for codebook-based applications and content-based search in large SVG databases.

SVG Image Captioning. We extend our approach to generate natural language descriptions of complete SVG images. Given an SVG image composed of multiple paths, we encode each path independently to obtain a sequence of path embeddings $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$. These embeddings are projected into the token embedding space of a pre-trained language model via a learned linear transformation:

$$\mathbf{e}_i = \mathbf{W}_{\text{proj}} \mathbf{z}_i, \tag{11}$$

where $\mathbf{W}_{\text{proj}} \in \mathbb{R}^{d_{\text{LLM}} \times d}$ is a learned matrix of parameters. The projected embeddings are prepended to the language model's input as prefix tokens, conditioning the autoregressive generation on the visual content:

$$p(\text{caption} \mid \text{SVG}) = \prod_{t=1}^{T_{\text{cap}}} p(w_t \mid w_{<t}, \mathbf{e}_1, \ldots, \mathbf{e}_N), \tag{12}$$

where w_t denotes the t-th word in the caption. This formulation treats SVG path embeddings as visual tokens, enabling the language model to ground generation in the geometric and stylistic properties encoded by SLS.

4 Experiments

4.1 Experimental setup

Dataset. To train and evaluate our model, we utilize a combination of publicly available SVG datasets, namely StarVector [26], HeisenVec [45], ColorSVG [6], and SVGX-Core [37]. In addition to SVG images, all datasets include captions automatically generated by a multimodal large language model (MLLM). Since our approach focuses on SVG path representations, we standardize and preprocess all data following the filtering procedure proposed in [45]. Specifically, we filter out paths exceeding 1024 tokens to ensure computational efficiency and maintain consistency across the training set, while leaving sufficient headroom for future context length extensions. This preprocessing pipeline yields a large-scale curated dataset comprising approximately 1.5M images for training, 31k for validation, and 16k for testing. At the path level, the dataset contains 25M training paths, 500k validation paths, and 250k test paths, providing sufficient data for robust SVG path encoding and generation tasks.

Training Details. We train SLS from scratch for 480k steps using a batch size of 192 and a learning rate of 1×10^{-4} with 6k warmup steps. The encoder and decoder share the same architecture with a hidden size of 1024, 8 attention heads, and an MLP ratio of 2, the overall number of parameters for both encoder and decoder is 135M. Training is performed on 4 NVIDIA A100 GPUs.

4.2 Reconstruction task

To validate the quality of SLS, we measured reconstruction performance at two levels: (i) *image level*, comparing original and reconstructed images using classical Computer Vision metrics (MSE-similarity, SSIM [44], LPIPS [43], DINOv2-Similarity [21]), and (ii) *path level*, analyzing token distribution with BLEU [32] and METEOR [4], and command coherency by computing the mean intersection over union (mIoU) of reconstructed paths over the original, without accounting for stylistic differences.

Table 1 shows SLS performance against that of DeepSVG [5] in terms of reconstruction capabilities. As DeepSVG is trained to embed entire SVG images rather than single paths, we compare against it only at the image level, evaluating its capabilities both in a zero-shot setting and after retraining it on our dataset. As can be seen, SLS outperforms DeepSVG by a significant margin on image-level reconstruction, even when retraining it, while respecting their limitation on maximum number of commands per path, which is one the its main limitations. Further, given that SLS produces path embeddings with constant norm, we also investigate whether magnitude information can instead be encoded during training through alternative mechanisms. We experimented with three techniques: (i) ℓ_2 regularization to minimize the CLS norm, (ii) multiplying the CLS token by a learnable logit scale to encode magnitude information in the scale factor rather than the embedding norm, and (iii) combining both approaches. Table 1 shows that these techniques do not learn better representations compared to

Table 1: Reconstruction performance at image and path levels.

Method	Image				Path		
	MSE↑	SSIM↑	LPIPS↓	DINO↑	mIoU↑	$BLEU_5$ ↑	MET↑
Baseline							
DeepSVG [5]	76.03	67.85	53.99	49.86	–	–	–
retrained	76.05	67.85	53.31	57.95	–	–	–
Ours							
w/ ℓ_2	86.51	81.05	28.63	72.41	66.79	62.05	90.02
w/ logit	71.98	58.63	62.38	36.83	5.63	**98.91**	77.45
w/ logit+ℓ_2	71.72	59.65	63.68	36.98	5.45	97.06	71.73
SLS (BPE)	**90.83**	**87.26**	**18.79**	**80.85**	**78.92**	96.21	**97.07**

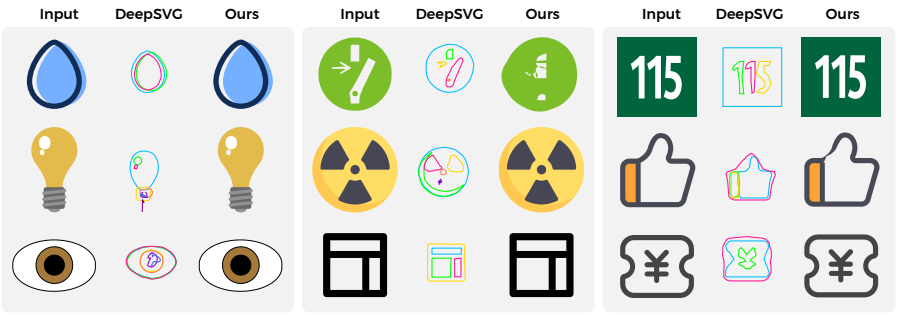

Fig. 3: Qualitative results on reconstruction of input image.

our final approach, confirming that constant ℓ_2 norm peculiarity emerges spontaneously. Figure 3 provides qualitative evidence that SLS consistently outperforms DeepSVG in reconstruction quality. This improvement is attributed to our model end-to-end design, which jointly processes style attributes and command/argument tokens at the input level, resulting in a latent representation that captures both geometric and visual properties.

Ablation studies. Table 2 further analyzes architectural and training choices, including projection dimensionality, pooling strategy (ViT-like [CLS] token vs. sequence modeling using the EOS representation), KL regularization, alternative tokenization schemes, and a block-wise image decomposition strategy. Results confirm that data-driven BPE tokenization and EOS-based pooling yield the best trade-off across reconstruction metrics, while command-based tokenizers significantly degrade performance. To assess whether fixed-length chunking could approximate path-level modeling, we experimented with multiple block sizes, consistently observing inferior performance compared to semantically coherent path decomposition. We find out that complete paths preserves both geometric structure and stylistic consistency better than fixed block chunking.

4.3 SVG Image Captioning

Experimental Setup. We then move to evaluating the quality of SLS embeddings through SVG-to-text generation. Following standard vision-language

Table 2: Architectural and training ablation studies. We evaluate the effect of *(i)* embedding projection size, *(ii)* training loss and pooling strategy (CLS vs. EOS, with/without KL), *(iii)* command-based tokenization approaches, and *(iv)* a block-wise image decomposition strategy instead of path-level modeling.

Method	Image				Path		
	MSE↑	SSIM↑	LPIPS↓	DINO↑	mIoU↑	$BLEU_5$ ↑	MET.↑
Projection size							
512	79.37	71.44	51.47	47.99	26.16	46.95	65.82
768	79.69	72.03	46.76	52.48	43.91	66.61	78.52
Pooling and training							
LAST	90.77	87.13	19.27	**81.08**	**79.17**	96.17	97.04
CLS	90.16	86.26	20.02	79.72	78.87	96.03	96.93
▷ w/ KL Loss	90.17	86.34	19.87	79.97	79.05	96.13	96.97
Tokenizers							
DeepSVG-like [5]	75.60	67.08	47.73	53.71	40.58	5.33	23.35
Iconshop-like [35]	79.26	67.61	46.01	58.34	21.86	1.68	16.57
Block size							
512 tokens	88.77	83.73	23.99	76.22	64.31	63.63	74.51
1024 tokens	89.21	84.13	24.06	76.84	61.33	61.46	70.10
SLS (Ours)	**90.83**	**87.26**	**18.79**	80.85	78.92	**96.21**	**97.07**

practices, we train lightweight projection layers that map SLS path embeddings into the input space of pre-trained language models. We experiment with three model scales representing different efficiency-capability trade-offs: Qwen3 0.6B [3], Llama 3.2 1B [1], and Gemma 2 2B [31].

The projection layer and language model are jointly trained end-to-end until convergence, allowing both components to adapt to the visual-language alignment task. All captioning models use a learning rate of 1×10^{-4} and batch size of 384. Additional training details are provided in the supplementary materials.

Evaluation Metrics. We evaluate caption quality using complementary metrics. ROUGE [18] emphasizes recall for longer texts, while BLEU [32] measures n-gram overlap with reference captions. METEOR [4] accounts for synonyms and paraphrasing. CLIP-Score [11] quantifies the semantic alignment between the generated caption and the rasterized SVG image using CLIP embeddings.

Results. To evaluate the semantic quality of our embedding space, we trained a set of LLMs to generate captions from SVG embeddings. Table 3 shows different encoders across three LLM backbones, and compares SLS to DeepSVG [5], the plain XML code, and CLIP [23]. Here, CLIP [23] serves as an upper-bound reference: while it produces semantically rich embeddings suitable for captioning, it cannot decode back to SVG, operating in a fundamentally different domain.

As can be seen, encoding SVG paths directly as XML text tokens performs poorly despite preserving complete information, as the extreme sequence lengths required make learning prohibitive. SLS approach achieves significantly better performance (+4.2 CLIP-Score, +29.01 $BLEU_5$) using dense embeddings, validating the effectiveness of our learned compression for downstream tasks. Further, SLS consistently outperforms DeepSVG across all metrics and LLMs, with

Table 3: SVG captioning performance with different encoders and language models. SLS surpasses invertible baselines (DeepSVG, XML) across all evaluation metrics, demonstrating superior semantic quality of learned embeddings. CLIP (gray) represents a non-invertible reference.

Encoder	SVG	Backbone	T2I ↑	$BLEU_5$↑	METEOR↑	$ROUGE_1$↑	$ROUGE_2$↑
CLIP [23]	✗	Qwen3 0.6B [3]	29.48	49.44	69.12	75.23	63.48
		Gemma2 2B [31]	29.03	24.78	53.72	55.70	41.40
		Llama 3.2 1B [1]	29.57	54.67	71.87	77.39	67.18
XML	✓	Qwen3 0.6B [3]	21.95	10.38	32.14	38.84	23.49
		Gemma2 2B [31]	22.73	12.62	34.48	44.31	26.67
		Llama 3.2 1B [1]	23.99	12.31	32.45	43.25	28.13
DeepSVG [5]	✓	Qwen3 0.6B [3]	25.42	27.30	40.55	48.96	37.02
		Gemma2 2B [31]	25.39	18.71	37.07	44.95	27.49
		Llama 3.2 1B [1]	23.25	18.77	35.82	44.77	27.22
SLS (Ours)	✓	Qwen3 0.6B [3]	**27.69**	45.60	64.95	70.43	59.06
		Gemma2 2B [31]	27.15	29.86	56.57	59.18	45.47
		Llama 3.2 1B [1]	**27.69**	**46.89**	**65.80**	**71.29**	**60.04**

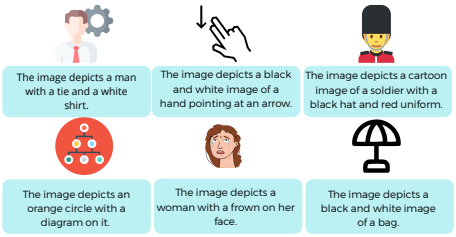

Fig. 4: Qualitative captioning samples of Llama 3.2 1B [1] as Large Language Model using SLS as SVG image encoder.

Fig. 5: Qualitative path retrieval results. Query paths (left) and top-ranked retrievals ordered by cosine similarity for SLS (Ours) and DeepSVG [5].

substantial average gains in CLIP-Score (+2.82), $BLEU_5$ (+19.19), and METEOR (+24.63), demonstrating superior semantic richness while maintaining invertibility. Qualitative results for SVG captioning are presented in Figure 4, where SLS serves as the visual encoder paired with Llama 3.2 1B as the language model, demonstrating strong caption quality.

Computational Efficiency. Table 4 reports the computational cost of SVG captioning with and without the SLS encoder. Replacing raw XML token sequences with our dense path embeddings reduces the average context length from 2432 to just 15.56 tokens, yielding a 156× speedup in context processing and a 167× reduction in training TFLOPs. This gain stems directly from our intrinsic learned compression: rather than feeding thousands of XML tokens per path to the language model, a single compact embedding captures the full geometric and stylistic content of each path. Crucially, this efficiency does not come at the cost of quality, SLS outperforms both DeepSVG and the raw XML baseline across all captioning metrics. While DeepSVG similarly avoids rasterization and could in principle offer comparable efficiency benefits, it lacks the stylistic and semantic richness encoded by SLS, which translates into consistently lower caption quality, showing its embedding limitations. These results highlight that compact path-level embeddings not only improve scalability, but also fundamen-

Table 4: Computational analysis of SVG image captioning with and without SLS. Our compact path-level representations reduce the average context length by 156× and the training cost by over 167× TFLOPs across three LLM backbones.

		Llama3 1B	Qwen3 0.6B	Gemma2 2B	Avg.	Speedup
Context length	w/o SLS	1636.55	2803.26	2856.27	2432.02	**156.29**
	w/ SLS	15.56	15.56	15.56	15.56	
Training TFLOPs	w/o SLS	9.82	10.09	34.27	18.06	**167.22**
	w/ SLS	0.09	0.05	0.18	0.108	

Table 5: Path retrieval performance on the SVG test set. SLS outperforms SVG-native encoders by retrieving more faithful SVG paths.

Model	Raster	MSE-Sim ↑	LAB Dist. ↓	BLEU_5 ↑	METEOR ↑
CLIP-B [23]	✓	98.67	11.66	10.58	34.12
DINOv2-B [21]	✓	98.86	14.68	10.77	34.55
DeepSVG [5]	✗	81.45	32.06	12.89	43.27
SLS (Ours)	✗	**89.95**	**25.27**	**20.91**	**47.78**

tally decouple downstream computational cost from the length and complexity of the underlying SVG markup.

4.4 Path Retrieval

Experimental Setup. We evaluate the path retrieval performance of different encoders on a large-scale benchmark. Specifically, with respect to the constraint posed by DeepSVG, we randomly choose 2.5k query paths and 266k document paths from our validation dataset. Each path was then independently embedded using its respective encoder, and all embeddings were ℓ_2-normalized before computing pairwise similarities. We then calculated the cosine similarity matrix between query and database embeddings. For each query, we select the top-1 retrieved path (i.e., the database path with highest cosine similarity). Since DeepSVG and SLS can decode embeddings back to SVG, while raster-based encoders such as DINOv2 [21] and CLIP [23] cannot, we recover the corresponding original SVG paths using the retrieved embedding indices for fair comparison. Retrieval quality was assessed through both visual and syntactic metrics. Visual similarity was measured using MSE-Sim and distance in CIE-LAB space, which captures differences in the color attributes. Syntactic fidelity was evaluated using BLEU_5 and METEOR, quantifying token-level pairs overlap.

Results. Table 5 reports path retrieval performance compared to DeepSVG [5], CLIP [20], and DINOv2 [21]. Among SVG-based methods, SLS achieves the best overall performance, significantly outperforming DeepSVG in both visual and syntactic metrics. In particular, SLS attains higher MSE-Sim, lower LAB distance, and improved $BLEU_5$ and $METEOR$, indicating better visual consistency and stronger structural fidelity. While raster-based encoders (CLIP, DINOv2) achieve higher visual similarity due to image-level supervision, they are less sensitive to vector structure. In contrast, SVG-based models better capture geometric and syntactic variations. As illustrated in Figure 5, SLS consistently

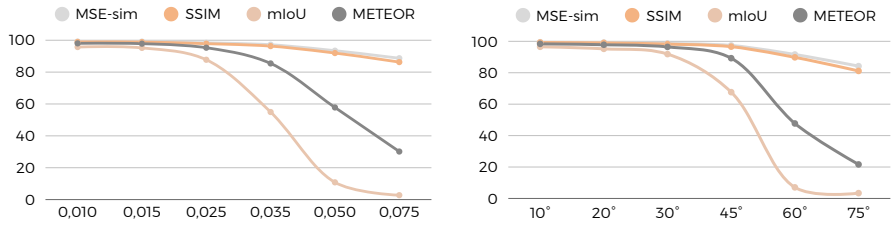

Fig. 6: Reconstruction performance of SLS under varying noise levels. Additive Gaussian noise (left) and rotational perturbations applied to input vectors (right).

retrieves more structurally similar paths, establishing it as the most effective encoder for vector path retrieval.

4.5 Space Robustness

To assess the stability of the learned embedding space, we systematically evaluate the effect of controlled perturbations applied directly to the latent representations. Two complementary regimes were considered: Gaussian noise and angular perturbations. Gaussian additive noise was introduced as

$$\tilde{\mathbf{z}} = \mathbf{z} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}), \tag{13}$$

with σ controlling the noise magnitude. In the angular regime, we perturbed the embeddings along the tangent space of the unit hypersphere, rotating them by increasing angular offsets θ, as follows:

$$\begin{aligned} \tilde{\mathbf{z}} &= \|\mathbf{z}\| \left[\cos(\Delta\theta)\, \hat{\mathbf{z}} + \sin(\Delta\theta)\, \hat{\mathbf{y}}\right], \\ \text{where } \hat{\mathbf{z}} = \frac{\mathbf{z}}{\|\mathbf{z}\|} \quad \hat{\mathbf{y}} &= \frac{\mathbf{r} - (\mathbf{r}^\top \hat{\mathbf{z}})\hat{\mathbf{z}}}{\|\mathbf{r} - (\mathbf{r}^\top \hat{\mathbf{z}})\hat{\mathbf{z}}\|}, \quad \mathbf{r} \sim \mathcal{N}(0, I_d); \end{aligned} \tag{14}$$

this simulates directional displacement within the latent manifold while preserving embedding norm. For each perturbed embedding $\tilde{\mathbf{z}}$, we decoded the corresponding SVG path with SLS decoder and compared it to the original using both visual and structural metrics. This setup provides a fine-grained view of how local perturbations in latent space affect geometric and syntactic consistency in the reconstructed paths. Results are visually reported in Figure 6, where we notice that the space learned with SLS is robust to rotations up to 30 degrees, and up to $\sigma = 0.02$ when applying Gaussian noise on ℓ_2-norm embeddings. This further attests the robustness of the embedding space.

4.6 Generalization Beyond Path-Level Reconstruction

We investigate whether the learned latent space generalizes across datasets and naturally extends to image-level representations. To assess cross-dataset generalization, we evaluate SLS on the UniSVG dataset [17], which is not included in the training mixture. As reported in Table 6, despite the domain shift, SLS exhibits

only a moderate degradation with respect to the in-domain evaluation, while maintaining high perceptual and structural reconstruction quality. These results indicate that the learned latent representations capture general properties of SVG paths rather than overfitting to the training distribution. Furthermore, although SLS is trained exclusively on individual path reconstruction, complete SVG images can be represented by aggregating the embeddings of their constituent paths. To evaluate these image-level representations, we train a lightweight Transformer classifier on the 20 most frequent classes of the ColorSVG [6] dataset, restricting each image to at most eight paths. As shown in Table 7, SLS nearly doubles the classification performance of DeepSVG across accuracy, precision, and recall, demonstrating that the learned path embeddings effectively compose into expressive image-level representations that generalize well to downstream image-level tasks.

Table 6: SLS performance in in-domain (ID) and out-of-domain (OOD) settings.

Model	MSE ↑	DINO ↑	LPIPS ↓	SSIM ↑
ID	90.83	80.85	18.79	87.26
OOD	86.67	76.12	30.18	78.44

Table 7: Multiple path representation.

Model	Acc. ↑	Prec. ↑	Rec. ↑
DeepSVG	15.36	17.89	15.02
SLS	**29.98**	**30.93**	**29.62**

5 Conclusion

We presented SLS, a Transformer-based autoencoder that learns compact, invertible latent representations of individual SVG paths. By treating path commands, coordinates, and style attributes as a unified token vocabulary through data-driven BPE tokenization, SLS produces dense embeddings that capture both geometric and stylistic information in a fixed-size vector — reducing sequence lengths by orders of magnitude compared to full-SVG token-based approaches. The learned space exhibits robustness to latent perturbations and generalizes to diverse downstream tasks, such as retrieval and captioning, through simple vector-space operations. By enabling compact, semantically structured representations of individual paths, we believe SLS lays the groundwork for future research in vector-native modeling, bridging symbolic vector representations with modern representation learning and large-scale foundation models.

Acknowledgements

This work has been conducted under a research grant co-funded by Doxee S.p.A. and supported by the EU Horizon project "ELLIOT - European Large Open Multi-Modal Foundation Models For Robust Generalization On Arbitrary Data Streams" (No. 101214398) and by the EU Horizon projects "ELIAS - European Lighthouse of AI for Sustainability" (No. 101120237). We further acknowledge the CINECA award, under the ISCRA initiative, for the availability of high-performance computing resources.

References

1. Aaron Grattafiori et al.: The Llama 3 Herd of Models. In: arXiv (2024) 11, 12
2. Alves, C., Traina, A.J.: Variational autoencoders for medical image retrieval. In: 2022 International conference on innovations in intelligent systems and applications (2022) 2
3. An Yang et al.: Qwen3 Technical Report (2025) 11, 12
4. Banerjee, S., Lavie, A.: Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In: Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization (2005) 9, 11
5. Carlier, Alexandre and Danelljan, Martin and Alahi, Alexandre and Timofte, Radu: Deepsvg: A hierarchical generative network for vector graphics animation. NeurIPS (2020) 2, 3, 4, 5, 7, 9, 10, 11, 12, 13
6. Chen, Zehao and Pan, Rong: SVGBuilder: Component-Based Colored SVG Generation with Text-Guided Autoregressive Transformers (2024) 9, 15
7. Cipriano, Marco and Feuerpfeil, Moritz and De Melo, Gerard: Vector Grimoire: Codebook-based Shape Generation under Raster Image Supervision (2025) 4
8. Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al.: Scaling rectified flow transformers for high-resolution image synthesis (2024) 2
9. Fan, Angela and Lewis, Mike and Dauphin, Yann: Hierarchical neural story generation. arXiv preprint (2018) 7
10. Ha, David and Eck, Douglas: A neural representation of sketch drawings. arXiv preprint (2017) 4
11. Hessel, Jack and Holtzman, Ari and Forbes, Maxwell and Le Bras, Ronan and Choi, Yejin: Clipscore: A reference-free evaluation metric for image captioning (2021) 11
12. Holtzman, Ari and Buys, Jan and Du, Li and Forbes, Maxwell and Choi, Yejin: The curious case of neural text degeneration. ICLR (2020) 7
13. Hu, Teng and Yi, Ran and Qian, Baihong and Zhang, Jiangning and Rosin, Paul L and Lai, Yu-Kun: Supersvg: Superpixel-based scalable vector graphics synthesis. In: CVPR (2024) 1, 5
14. Jain, A., Xie, A., Abbeel, P.: Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In: CVPR (2023) 1
15. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2013) 2
16. LCM team, Loïc Barrault, Paul-Ambroise Duquenne, Maha Elbayad, Artyom Kozhevnikov, Belen Alastruey, Pierre Andrews, Mariano Coria, Guillaume Couairon, Marta R. Costa-jussà, David Dale, Hady Elsahar, Kevin Heffernan, João Maria Janeiro, Tuan Tran, Christophe Ropers, Eduardo Sánchez, Robin San Roman, Alexandre Mourachko, Safiyyah Saleem, Holger Schwenk: Large Concept Models: Language Modeling in a Sentence Representation Space. In: arXiv (2024) 5
17. Li, Jinke and Yu, Jiarui and Wei, Chenxing and Dong, Hande and Lin, Qiang and Yang, Liangjing and Wang, Zhicai and Hao, Yanbin: Unisvg: A unified dataset for vector graphic understanding and generation with multimodal large language models. In: ACM MM (2025) 14
18. Lin, Chin-Yew and Och, Franz Josef: Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics (2004) 11
19. Liu, Ying-Tian and Zhang, Zhifei and Guo, Yuan-Chen and Fisher, Matthew and Wang, Zhaowen and Zhang, Song-Hai: Dualvector: Unsupervised vector font synthesis with dual-part representation. In: CVPR (2023) 4

20. Lopes, R.G., Ha, D., Eck, D., Shlens, J.: A learned representation for scalable vector graphics. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7930–7939 (2019) 4, 13
21. Maxime Oquab and Timothée Darcet and Théo Moutakanni and Huy V. Vo and Marc Szafraniec and Vasil Khalidov and Pierre Fernandez and Daniel HAZIZA and Francisco Massa and Alaaeldin El-Nouby and Mido Assran and Nicolas Ballas and Wojciech Galuba and Russell Howes and Po-Yao Huang and Shang-Wen Li and Ishan Misra and Michael Rabbat and Vasu Sharma and Gabriel Synnaeve and Hu Xu and Herve Jegou and Julien Mairal and Patrick Labatut and Armand Joulin and Piotr Bojanowski: DINOv2: Learning Robust Visual Features without Supervision (2024) 2, 3, 9, 13
22. Paul-Ambroise Duquenne and Holger Schwenk and Benoit Sagot: SONAR: Sentence-Level Multimodal and Language-Agnostic Representations. In: arXiv (2023) 5
23. Radford, Alec and Kim, Jong Wook and Hallacy, Chris and Ramesh, Aditya and Goh, Gabriel and Agarwal, Sandhini and Sastry, Girish and Askell, Amanda and Mishkin, Pamela and Clark, Jack and others: Learning transferable visual models from natural language supervision (2021) 2, 3, 11, 12, 13
24. Rafiei, M., Iosifidis, A.: Class-specific variational auto-encoder for content-based image retrieval. In: 2023 International Joint Conference on Neural Networks (2023) 2
25. Reddy, Pradyumna and Gharbi, Michaël and Lukac, Michal and Mitra, Niloy J.: Im2Vec: Synthesizing Vector Graphics without Vector Supervision. In: CVPR (2021) 4
26. Rodriguez, Juan A. and Puri, Abhay and Agarwal, Shubham and Laradji, Issam H. and Rodriguez, Pau and Rajeswar, Sai and Vazquez, David and Pal, Christopher and Pedersoli, Marco: Starvector: Generating scalable vector graphics code from images and text. In: CVPR (2025) 4, 9
27. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR (2022) 2
28. Sagi Polaczek and Yuval Alaluf and Elad Richardson and Yael Vinker and Daniel Cohen-Or: NeuralSVG: An Implicit Representation for Text-to-Vector Generation. In: arxiv (2025) 2, 5
29. Saha, S., Minku, L.L., Yao, X., Senhoff, B., Menzel, S.: Exploiting linear interpolation of variational autoencoders for satisfying preferences in evolutionary design optimization. In: 2021 IEEE Congress on Evolutionary Computation (2021) 2
30. Team, N., Costa-jussà, M.R., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., Kalbassi, E., Lam, J., Licht, D., Maillard, J., Sun, A., Wang, S., Wenzek, G., Youngblood, A., Akula, B., Barrault, L., Gonzalez, G.M., Hansanti, P., Hoffman, J., Jarrett, S., Sadagopan, K.R., Rowe, D., Spruit, S., Tran, C., Andrews, P., Ayan, N.F., Bhosale, S., Edunov, S., Fan, A., Gao, C., Goswami, V., Guzmán, F., Koehn, P., Mourachko, A., Ropers, C., Saleem, S., Schwenk, H., Wang, J.: No Language Left Behind: Scaling Human-Centered Machine Translation. In: arXiv (2022) 5
31. Team, Gemma and Riviere, Morgane and Pathak, Shreya and Sessa, Pier Giuseppe and Hardin, Cassidy and Bhupatiraju, Surya and Hussenot, Léonard and Mesnard, Thomas and Shahriari, Bobak and Ramé, Alexandre and others: Gemma 2: Improving open language models at a practical size. In: arXiv preprint (2024) 11, 12
32. vPapineni, Kishore and Roukos, Salim and Ward, Todd and Zhu, Wei-Jing: Bleu: a method for automatic evaluation of machine translation (2002) 9, 11

33. Wang, Feiyu and Zhao, Zhiyuan and Liu, Yuandong and Zhang, Da and Gao, Junyu and Sun, Hao and Li, Xuelong: SVGen: Interpretable Vector Graphics Generation with Large Language Models. In: ACM MM (2025) 4
34. Wang, Yuqing and Wang, Yizhi and Yu, Longhui and Zhu, Yuesheng and Lian, Zhouhui: Deepvecfont-v2: Exploiting transformers to synthesize vector fonts with higher quality. In: CVPR (2023) 2, 3, 4
35. Wu, Ronghuan and Su, Wanchao and Ma, Kede and Liao, Jing: Iconshop: Text-guided vector icon synthesis with autoregressive transformers. ACM TOG (2023) 2, 3, 4, 11
36. Xing, X., Zhou, H., Wang, C., Zhang, J., Xu, D., Yu, Q.: Svgdreamer: Text guided svg generation with diffusion model. In: CVPR (2024) 1
37. Xing, Ximing and Hu, Juncheng and Liang, Guotao and Zhang, Jing and Xu, Dong and Yu, Qian: Empowering llms to understand and generate complex vector graphics. In: CVPR (2025) 2, 4, 9
38. Xing, Ximing and Hu, Juncheng and Zhang, Jing and Xu, Dong and Yu, Qian: SVGFusion: Scalable Text-to-SVG Generation via Vector Space Diffusion (2024) 5
39. Xu, J., Liu, B., Zhou, Y., Liu, M., Yao, R., Shao, Z.: Diverse image captioning via conditional variational autoencoder and dual contrastive learning. ACM Transactions on Multimedia Computing, Communications and Applications (2023) 2
40. Yang, Y., Cheng, W., Chen, S., Zeng, X., Zhang, J., Wang, L., Yu, G., Ma, X., Jiang, Y.G.: OmniSVG: A Unified Scalable Vector Graphics Generation Model. NeurIPS (2025) 2, 4
41. Zhang, Peiying and Zhao, Nanxuan and Fisher, Matthew and Xu, Yiran and Liao, Jing and Liu, Difan: Duetsvg: Unified multimodal svg generation with internal visual guidance. In: CVPR (2026) 4
42. Zhang, Peiying and Zhao, Nanxuan and Liao, Jing: Text-to-Vector Generation with Neural Path Representation. ACM TOG (2024) 5
43. Zhang, Richard and Isola, Phillip and Efros, Alexei A and Shechtman, Eli and Wang, Oliver: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018) 9
44. Zhou Wang and Bovik, A.C. and Sheikh, H.R. and Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE TIP (2004) 9
45. Zini, L., Frigieri, E., Aloscari, S., Baraldi, L.: vHector and HeisenVec: Scalable Vector Graphics Generation Through Large Language Models. In: NeurIPS (2025) 2, 4, 5, 9

A Scalable Vector Graphics Latent Space
Supplementary Material

Leonardo Zini[1], Elia Frigieri[1], and Lorenzo Baraldi[1]

University of Modena and Reggio Emilia, Italy
{name.surname}@unimore.it

1 Method details

1.1 Latent Space Norm Characterization

Analysis of Magnitude Distribution. Through empirical analysis of the learned latent space, we observe a distinctive geometric property: the latent representations exhibit remarkably consistent ℓ_2-norms across the dataset. Specifically, measuring the distribution of latent vector magnitudes after training convergence, we find a mean norm of $\mu_{\|\mathbf{z}\|} \approx 214.943$ with a standard deviation of $\sigma_{\|\mathbf{z}\|} \approx 1.89$ for latent vectors $\mathbf{z} \in \mathbb{R}^{1024}$. This near-constant magnitude property (coefficient of variation $\text{CV} = \frac{\sigma}{\mu} < 0.01$) indicates that the learned representations naturally concentrate on an approximate hypersphere in the latent space. We hypothesize that the cross-entropy reconstruction objective, combined with the continuous nature of SVG representations, naturally induces an L2-spherical geometry in the learned embedding space where the autoencoder learns to encode information primarily in directional variations rather than magnitude variations.

Normalization-Denormalization Protocol. This consistent magnitude property enables a practical normalization scheme for downstream tasks. Given a latent representation $\mathbf{z}$ with $\|\mathbf{z}\|_2 \approx \mu_{\|\mathbf{z}\|}$, we can apply ℓ_2-normalization to obtain a unit-norm representation:

$$\hat{\mathbf{z}} = \mathbf{z}/\|\mathbf{z}\|_2 \quad \mathbf{z}' = \mu_{\|\mathbf{z}\|} \cdot \hat{\mathbf{z}} \tag{1}$$

The reconstruction error $\|\mathbf{z} - \mathbf{z}'\|_2$ is bounded by the magnitude variation $\sigma_{\|\mathbf{z}\|}$. We observe a consistently low reconstruction error across our testing dataset, corresponding to a very small relative deviation in latent-space coordinates. We confirm that the normalization–denormalization procedure introduces negligible impact on reconstruction quality. Rendering from the transformed latent representation instead of the original leads to only a minute change in Chamfer Distance [1], small enough to remain imperceptible in the final output.

Implications for Representation Learning. This property distinguishes our learned latent space from typical autoencoder representations, where magnitude often carries semantic information. The alignment to a hypersphere suggests that semantic variations in SVG structure in our space are encoded primarily in angular relationships rather than radial distance from the origin.

This geometric structure may reflect fundamental properties of SVG representation spaces and merits further investigation in future works on structured vector graphics generation and analysis.

1.2 Training and inference pseudocode

Training. During training, the model learns to encode SVG paths into compressed representations and reconstruct them in an autoregressive manner. Given a tokenized path sequence, the encoder processes the entire sequence bidirectionally and extracts semantic features from the hidden state corresponding to the last valid (non-padding) token, adopting a last-token pooling strategy similar to [3]. The decoder then learns to rebuild the path token-by-token in an autoregressive fashion.

Algorithm 1 SLS Training Forward Pass

Require: Tokenized path sequence $\mathbf{x} \in \mathbb{Z}^{B \times L}$, learnable CLS token $\mathbf{CLS} \in \mathbb{R}^{d_{\text{model}}}$
Ensure: Loss value and logits

$\mathbf{y} \leftarrow \text{ShiftLeft}(\mathbf{x})$ ▷ Next-token prediction targets

// **Encoding Phase (Non-Causal)**
$\mathbf{m} \leftarrow \text{PaddingMask}(\mathbf{x})$ ▷ 1 for PAD, 0 for valid tokens
$\mathbf{E} \leftarrow \text{Embed}(\mathbf{x})$
$\mathbf{E} \leftarrow \mathbf{E} + \text{PositionalEncoding}(\mathbf{E})$
$\mathbf{H} \leftarrow \text{SLS Encoder}(\mathbf{E}, \text{src\_key\_padding\_mask} = \mathbf{m})$
$\boldsymbol{\ell} \leftarrow L - \sum_{t=1}^{L} \mathbf{m}_{:,t}$ ▷ Valid lengths
$\mathbf{z} \leftarrow \text{LastTokenPool}(\mathbf{H}, \boldsymbol{\ell})$

// **Latent Noise Injection**
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ ▷ Gaussian perturbation
$\tilde{\mathbf{z}} \leftarrow \mathbf{z} + \boldsymbol{\epsilon}$

// **Decoding Phase (Autoregressive)**
$\mathbf{M}_{\text{causal}} \leftarrow \text{CausalMask}(L + 1)$ ▷ Classical autoregressive mask
$\mathbf{E}' \leftarrow \text{Embed}(\mathbf{x})$
$\mathbf{E}' \leftarrow [\tilde{\mathbf{z}}; \mathbf{E}']$ ▷ Prepend noisy CLS representation
$\mathbf{E}' \leftarrow \mathbf{E}' + \text{PositionalEncoding}(\mathbf{E}')$
$\mathbf{D} \leftarrow \text{SLS Decoder}(\mathbf{E}', \text{mask} = \mathbf{M}_{\text{causal}})$
$\mathbf{logits} \leftarrow \text{Linear}(\mathbf{D})$

// **Loss Computation**
$\mathcal{L} \leftarrow \text{CrossEntropyLoss}(\mathbf{logits}, \mathbf{y}) + 0.2 \cdot \text{KLDivLoss}(\mathbf{logits}, \mathbf{y})$
return $\mathcal{L}, \mathbf{logits}$

The training objective is to minimize the reconstruction loss, defined as a linear combination between cross-entropy and kl divergence loss, between the decoder output logits and the original token sequence. This encourages the model to compress path information into the latent representation $\mathbf{z}$ while ensuring accurate sequence synthesis.

Inference. At inference time, each SVG path is processed independently. The model encodes each path into its latent representation and subsequently decodes it autoregressively, generating tokens sequentially. The final vector graphic is reconstructed by stacking the decoded paths in their original input order.

Algorithm 2 SLS Image Reconstruction

Require: Image paths $\mathcal{P} = \{p_1, \ldots, p_N\}$, max length $L_{\max}$, BOS token t_{bos}, EOS token t_{eos}, mean norm $\mu_{\|\mathbf{z}\|}$
Ensure: Reconstructed image paths $\mathcal{P}_{\text{gen}}$

$\mathcal{P}_{\text{gen}} \leftarrow \emptyset$
for $i = 1$ **to** N **do** ▷ Process each path independently
  $\mathbf{x} \leftarrow \text{Tokenize}(p_i)$ ▷ Convert path to tokens
  $\mathbf{x} \leftarrow \text{PadOrTruncate}(\mathbf{x}, L_{\text{enc}})$ ▷ Optional: encoder length

  // **Encoding Phase**
  $\mathbf{m} \leftarrow \text{PaddingMask}(\mathbf{x})$ ▷ 1 for PAD, 0 for valid tokens
  $\boldsymbol{\ell} \leftarrow \sum_{t=1}^{L_{\text{enc}}} (1 - \mathbf{m}_t)$ ▷ Valid length
  $\mathbf{E} \leftarrow \text{Embed}(\mathbf{x})$
  $\mathbf{E} \leftarrow \mathbf{E} + \text{PositionalEncoding}(\mathbf{E})$
  $\mathbf{H} \leftarrow \text{SLS Encoder}(\mathbf{E}, \text{src\_key\_padding\_mask} = \mathbf{m})$
  $\mathbf{z}_i \leftarrow \text{LastTokenPool}(\mathbf{H}, \boldsymbol{\ell})$ ▷ Last valid token
  $\hat{\mathbf{z}}_i \leftarrow \mathbf{z}_i / \|\mathbf{z}_i\|_2$ ▷ L2-normalize latent vector

  // **Autoregressive Decoding**
  $\mathbf{z}'_i \leftarrow \mu_{\|\mathbf{z}\|} \cdot \hat{\mathbf{z}}_i$ ▷ Scale to mean norm
  $\mathbf{x}_{\text{gen}} \leftarrow [t_{\text{bos}}]$ ▷ Initialize with BOS token
  finished $\leftarrow$ False
  while ¬finished **and** $|\mathbf{x}_{\text{gen}}| < L_{\max}$ **do**
    $\mathbf{M}_{\text{causal}} \leftarrow \text{CausalMask}(|\mathbf{x}_{\text{gen}}| + 1)$
    $\mathbf{E}' \leftarrow \text{Embed}(\mathbf{x}_{\text{gen}})$
    $\mathbf{E}' \leftarrow [\mathbf{z}'_i; \mathbf{E}']$
    $\mathbf{E}' \leftarrow \mathbf{E}' + \text{PositionalEncoding}(\mathbf{E}')$
    $\mathbf{D} \leftarrow \text{SLS Decoder}(\mathbf{E}', \mathbf{M}_{\text{causal}})$
    logits $\leftarrow \text{Linear}(\mathbf{D}[-1, :])$ ▷ Predict next token
    $t_{\text{next}} \leftarrow \text{Sample}(\textbf{logits})$ ▷ Greedy or sampling
    $\mathbf{x}_{\text{gen}} \leftarrow [\mathbf{x}_{\text{gen}}; t_{\text{next}}]$
    if $t_{\text{next}} = t_{\text{eos}}$ **then**
      finished $\leftarrow$ True
    end if
  end while
  $p_i^{\text{gen}} \leftarrow \text{Detokenize}(\mathbf{x}_{\text{gen}})$
  $\mathcal{P}_{\text{gen}} \leftarrow \mathcal{P}_{\text{gen}} \cup \{p_i^{\text{gen}}\}$
end for
return $\mathcal{P}_{\text{gen}}$ ▷ All reconstructed paths form the image

2 Training Details

2.1 SLS Training Details

Model Architecture. Our autoencoder is based on a Transformer architecture with hidden dimension 1024, 8 attention heads across 8 encoder and decoder layers each, and a feedforward dimension of 2048. The model operates on a vocabulary of 448 tokens with a maximum sequence length of 1024 tokens, comprising approximately 135M total parameters (overall for SLS: encoder + decoder).

Optimization. Training is performed using the AdamW optimizer with learning rate 1×10^{-4} following a constant schedule after 6,000 warmup steps, weight decay of 0.01, and gradient clipping with maximum norm of 1.0. We use a batch size equal to 192 and employ bfloat16 mixed precision training for computational efficiency. Early stopping with a patience of 5 evaluation checkpoints is applied to prevent overfitting, monitoring validation loss as the primary metric.

Data Augmentation. To improve robustness to syntactic variations in SVG formatting, we apply a stochastic attribute-order augmentation during training. With probability 0.5 per path, we swap the order of the `style` and `d` attributes within the SVG `<path>` element, without altering the rendered geometry. This encourages the model to learn representations that are invariant to attribute ordering and superficial formatting differences in the SVG source.

2.2 Captioner training details

Model Architecture. Our vision-language model combines the pretrained SLS encoder with a language model backbone. The SVG encoder produces 1024-dimensional representations directly fed to the language model that are linearly projected to match the language model dimension. The language model processes text sequences with a maximum length of 512 tokens and is fine-tuned on the captioning task, generating natural language descriptions of SVG images. We experiment three language model backbones: Llama-3.2-1B, Gemma-2-2B, and Qwen3-0.6B, using identical training configurations across all variants to ensure fair comparison. We further truncate each SVG up to 64 paths.

Optimization. Training is performed using the AdamW optimizer with learning rate 1×10^{-4} following a constant schedule after 2,000 warmup steps and weight decay of 0.1. We use a batch size of 384. Training employs bfloat16 mixed precision for computational efficiency. Early stopping with a patience of 3 evaluation checkpoints is applied to prevent overfitting, monitoring validation loss as the primary metric.

3 Tokenizer Vocabulary Size Selection

To determine the optimal vocabulary size for our SVG path tokenizer, we conduct an ablation study using tokenizers trained on 4M path samples. We evaluate vocabulary sizes from 128 to 1024 tokens, measuring average sequence length, compression ratio, token distribution concentration (% top50), and vocabulary utilization. Once the best configuration is chosen, we trained our tokenizer on a corpus of 35M paths.

We select a vocabulary size of 448 tokens, which balances compression efficiency and vocabulary utilization. This configuration achieves an average tokens-per-sequence equal to 110, a 2.29× compression ratio, with the top-50 most frequent tokens covering 63.6% of occurrences. Only 139 tokens (31.03% of vocabulary) are needed to reach 80% coverage, indicating efficient vocabulary utilization without excessive sparsity.

4 Sampling techniques

We conduct a comprehensive evaluation of different decoding strategies for SLS. Table 1 presents quantitative results across seven evaluation metrics, comparing greedy decoding against three stochastic sampling methods: Nucleus (Top-P) [2], Top-K, and multinomial sampling.

Sampling Methods. Greedy decoding selects the highest probability token at each step, ensuring deterministic outputs. Multinomial sampling draws tokens randomly from the full probability distribution. Top-K sampling restricts sampling to the K most probable tokens at each step, while Nucleus sampling dynamically selects from the smallest set of tokens whose cumulative probability exceeds the threshold p.

Results. Greedy decoding consistently outperforms all stochastic sampling alternatives across nearly all metrics, achieving the highest scores on MSE-sim, DINOv2-Sim, SSIM, mIoU, BLEU_5, and METEOR. The deterministic nature of greedy decoding is particularly well-suited for SVG code reconstruction, where syntactic correctness and precise token selection are critical. These results demonstrate that for SVG reconstruction, greedy decoding is not merely preferred but required. The exact, structured nature of SVG code, where a single incorrect token can invalidate the entire sequence, is fundamentally incompatible with stochastic sampling. Even minor deviations from the optimal token sequence lead to syntactic errors and corrupted outputs, making deterministic decoding essential for this task.

5 Reconstruction Quality vs Path Length

Motivation. Unlike raster-based generation models that operate on fixed-resolution pixel grids, SVG reconstruction inherently depends on sequence length, as each path is represented as a variable-length string of commands and coordinates. In the main paper, we noted that path length plays a critical role in reconstruction quality due to the autoregressive nature of the decoder, which generation is conditioned by a single embedding. Here, we provide a systematic analysis of how reconstruction metrics degrade as a function of maximum path length within SVG images.

Experimental Setup. We partition the test dataset into three bins according to the maximum character length of the longest path in each SVG. Let

$$L(\text{SVG}) = \max_{p \in \text{paths}} |\text{chars}(p)| \tag{2}$$

denote the maximum path length within a given SVG. We then define three disjoint subsets:

$$\mathcal{D}_{(a,b]} = \{\text{SVG} \mid a < L(\text{SVG}) \leq b\}, \quad (a,b] \in \{(0,512], (512,1024], (1024,2048]\}. \tag{3}$$

Table 1: Reconstruction performance of SLS with different sampling methods during decoding phase.

Method	Image				Path		
	MSE↑	SSIM↑	LPIPS↓	DINO↑	mIoU↑	$BLEU_5$ ↑	MET↑
Top-K	89.08	84.66	22.61	77.01	78.54	95.47	96.62
Nucleus	89.41	85.18	21.80	77.95	78.87	95.76	96.81
Multinomial	88.87	84.35	22.98	76.57	78.54	95.43	96.59
Greedy	**90.83**	**87.26**	**18.79**	**80.85**	**78.92**	**96.21**	**97.07**

Table 2: Reconstruction quality of SLS as a function of maximum path length. Metrics marked with ↑ indicate higher is better, ↓ lower is better.

Path Length	Image				Path		
	MSE↑	SSIM↑	LPIPS↓	DINO↑	mIoU↑	$BLEU_5$ ↑	MET↑
0 – 512 chars	97.40	96.30	5.76	94.37	88.85	98.62	99.10
512 – 1024 chars	86.29	81.07	28.29	71.83	77.53	95.19	96.28
1024 – 2048 chars	79.69	71.72	40.85	57.69	64.86	93.38	94.78
Overall	**90.83**	**87.26**	**18.79**	**80.85**	**78.92**	**96.21**	**97.07**

Here, 2048 characters correspond to the maximum path length observed in the dataset. For each subset, we computed reconstruction quality using both pixel-based metrics (MSE-sim, DINOv2-sim, SSIM, LPIPS) and structure-aware metrics (mIoU, $BLEU_5$, METEOR). The MSE-similarity is defined as:

$$\text{MSE-sim} = 1 - \frac{\text{MSE}(\mathbf{I}_{\text{orig}}, \mathbf{I}_{\text{rec}})}{\text{MSE}_{\text{max}}} \tag{4}$$

where $\mathbf{I}_{\text{orig}}$ and $\mathbf{I}_{\text{rec}}$ are the original and reconstructed rasterized images, and MSE_{max} is a normalization constant (typically the maximum possible squared error). Likewise, DINOv2-sim measures cosine similarity in DINOv2 feature space.

Quantitative Results. Table 2 presents the reconstruction quality across different path length ranges. Several key trends emerge:

- **Strong performance for short paths** (0–512 chars): SLS achieves high reconstruction quality across all metrics for short paths. Image-based scores are particularly strong (MSE: 97.40, SSIM: 96.30, DINO: 94.37), while path-level metrics are near-perfect (mIoU: 88.85, $BLEU_5$: 98.62, MET: 99.10). This indicates that the model reliably reconstructs both geometric structure and textual path representation when sequence lengths are moderate.
- **Monotonic degradation with increasing length**: As path length increases, all image- and geometry-based metrics degrade smoothly. For example, SSIM drops from 96.30 (0–512) to 71.72 (1024–2048), a decrease of 24.58 percentage points. Similarly, mIoU decreases from 88.85 to 64.86 ($\Delta = 23.99$). This trend highlights the growing difficulty of accurately decoding longer and more complex path sequences.
- **Perceptual sensitivity to long sequences**: LPIPS increases substantially from 5.76 to 40.85 as path length grows, indicating a noticeable rise

in perceptual dissimilarity. DINO similarity also drops significantly (94.37 $\rightarrow$ 57.69), suggesting that high-level semantic features become increasingly distorted for long paths.
- **Text-based metrics are more robust**: $BLEU_5$ and METEOR exhibit only moderate degradation across bins ($BLEU_5$: 98.62 $\rightarrow$ 93.38; METEOR: 99.10 $\rightarrow$ 94.78). Compared to image-based metrics, this smaller drop indicates that the textual path syntax remains largely consistent, even when rendered geometry accumulates noticeable distortions.
- **Overall trend**: The results demonstrate that reconstruction quality scales inversely with path length. While short and medium-length paths are reconstructed with high fidelity, long sequences introduce compounding decoding errors that primarily affect perceptual and fine-grained structural metrics, with textual similarity being comparatively more resilient.

5.1 Syntactic Error Analysis

To assess the structural validity of the generated SVGs, we performed a syntactic analysis over all reconstructed files. Out of 16,159 generated SVG files, 15,369 are valid and directly renderable, corresponding to a validity rate of 95.11%. A total of 790 files exhibit syntax errors, resulting in an overall error rate of 4.89%. Importantly, none of the valid files were classified as non-visible, meaning that all syntactically valid SVGs also produced visible renderings.

Analysis. The vast majority of errors (783 out of 790) are caused by malformed XML structures. Only 3 cases are due to invalid path commands, and 4 cases fall into an unknown category. No truncated sequences were detected, indicating that generation rarely stops prematurely. Similarly, numeric and style-related errors were not observed. The dominance of malformed XML errors suggests that failures primarily arise from minor structural inconsistencies (e.g., tag mismatches or improper nesting) rather than from incorrect geometric reasoning. The extremely small number of invalid path commands (3 cases, $< 0.02\%$ of all generations) indicates that the model has learned the SVG path grammar.

Overall, the results demonstrate strong syntactic robustness, with more than 95% of generations being valid and renderable. Remaining errors are predominantly low-level formatting issues rather than semantic or geometric failures.

6 SLS Embedding Space Analysis

Beyond evaluating reconstruction quality under noise (Figure 6 of the main paper), we conducted a comprehensive visual analysis of how controlled perturbations in the embedding space affect the decoded SVG paths. This analysis provides qualitative/quantitative insights into the smoothness and semantic structure of the learned latent manifold.

Experimental Setup. We extracted per-path embeddings, using SLS encoder, for each of the images from our testing dataset. Then, two types of perturbations have been applied:

1. **Rotational perturbations**: Embeddings were rotated by varying angles, ranging from small perturbations (10°) to stronger rotations (75°), within the tangent space of the unit hypersphere. Given an embedding $\mathbf{z}$, we compute:

$$\tilde{\mathbf{z}} = \|\mathbf{z}\|_2 \left(\cos(\Delta\theta)\hat{\mathbf{z}} + \sin(\Delta\theta)\hat{\mathbf{y}}\right) \tag{5}$$

   where $\hat{\mathbf{z}} = \mathbf{z}/\|\mathbf{z}\|_2$ is the normalized embedding direction, and $\hat{\mathbf{y}}$ is a random orthogonal direction obtained via Gram-Schmidt:

$$\hat{\mathbf{y}} = \frac{\mathbf{r} - (\mathbf{r}^\top \hat{\mathbf{z}})\hat{\mathbf{z}}}{\|\mathbf{r} - (\mathbf{r}^\top \hat{\mathbf{z}})\hat{\mathbf{z}}\|_2}, \quad \mathbf{r} \sim \mathcal{N}(0, \mathbf{I}_d) \tag{6}$$

   This simulates directional displacement within the latent manifold while preserving the embedding norm.
2. **Gaussian noise**: additive noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ was applied with different σ to ℓ_2-normalized embeddings,ensuring the perturbed embedding remains on the unit hypersphere:

$$\tilde{\mathbf{z}} = \frac{\mathbf{z} + \boldsymbol{\epsilon}}{\|\mathbf{z} + \boldsymbol{\epsilon}\|_2} \tag{7}$$

Each perturbed embedding was decoded back to an SVG path, rasterized to 512×512 RGBA image, and compared against the original using both pixel-level metrics (MSE, SSIM, LPIPS) and structural metrics (IoU, BLEU_5, METEOR). The last two metrics were computed by tokenizing SVG path commands and style attributes, providing a syntax-aware similarity measure.

SVG Tokenization for Structural Metrics. For text-based structural evaluation (BLEU and METEOR), each SVG path element is tokenized into a sequence of discrete symbols. Given a path string, we separate:

- **Geometry tokens**: path commands (e.g., `M`, `L`, `C`, `A`, `Z`) and their associated numeric coordinates.
- **Style tokens**: attributes such as `fill`, `rgb`, individual color values, opacity, and stroke-width.

BLEU-n is computed using standard n-gram precision with brevity penalty, and we report BLEU_5 with smoothing to avoid zero-count issues. METEOR is computed using unigram alignment with precision–recall balancing and a fragmentation penalty, making it more sensitive to structural ordering differences.

Intersection over Union for Vector Paths. To measure geometric agreement, we rasterize both the original and reconstructed SVG paths into binary masks at fixed resolution and compute the Intersection over Union (IoU) between them. This metric captures overlap of the rendered shapes and is particularly sensitive to boundary misalignments and coordinate errors.

Qualitative Observations. Visual inspection of decoded paths under latent perturbations reveals a gradual degradation pattern:

- **Small perturbations** ($\theta \leq 20°$ or $\sigma \leq 0.02$) lead to minor geometric variations (e.g., small shifts or smooth deformations), while preserving overall structure and semantics. This suggests a locally smooth latent manifold.

- **Medium perturbations** ($20° < \theta \leq 45°$ or $0.02 < \sigma \leq 0.05$) introduce noticeable geometric and stylistic changes, though the global shape category typically remains recognizable.
- **Large perturbations** ($\theta > 45°$ or $\sigma > 0.05$) produce substantial structural deviations, often resulting in different shapes or styles, consistent with moving farther away in latent space.

Quantitative Trends. As shown in Figure 6 of the main paper, reconstruction metrics degrade gracefully with increasing perturbation magnitude. The smooth degradation curves, particularly for IoU and BLEU_5 , suggest that the embedding space does not contain sharp discontinuities or mode collapses. The relatively high robustness up to $\theta \approx 30$ and $\sigma \approx 0.02$ indicates that the learned representations are stable under realistic amounts of noise, which is beneficial for applications such as retrieval and captioning.

Comparison with Geometric Metrics. Interestingly, pixel-based metrics (MSE-sim, SSIM, LPIPS) degrade more rapidly than structural metrics (IoU, BLEU_5, METEOR) under rotational perturbations. This suggests that rotations in embedding space often correspond to transformations that preserve syntactic structure (path commands) while modifying geometric details (vertex positions). In contrast, Gaussian noise affects both aspects more uniformly, as evidenced by the more parallel degradation of all metrics.

Qualitative comparison. We qualitatively assess robustness under perturbations in Figure 4, where SVG paths are subjected to rotations and Gaussian noise. SLS preserves high-fidelity reconstructions up to 30° rotation and noise with standard deviation $\sigma = 0.02$, indicating that the learned embeddings capture stable geometric semantics. These results complement the quantitative analysis and highlight the robustness of the latent representation.

7 SLS Reconstruction Qualitatives

Figure 1 presents comprehensive qualitative examples comparing reconstruction quality between SLS and DeepSVG across a diverse set of SVG paths. As it can be seen, SLS consistently produces high-fidelity reconstructions with accurate preservation of geometric details and stylistic attributes. The breadth of examples demonstrates the robustness and generalization capability of our learned representations across varied path complexities and structures. Overall, these qualitative results complement the quantitative reconstruction metrics in the main paper and validate the effectiveness of the proposed method.

Failure Cases. While the model occasionally exhibits reduced accuracy on particularly intricate or highly irregular paths, these deviations are typically localized and do not compromise the underlying semantic structure or geometric intent. In more challenging examples, the reconstructions remain structurally coherent, and the embeddings still accurately capture the dominant shape characteristics. These cases outline natural boundaries of the current formulation while illustrating that the learned representation remains stable and meaningful even when exact token-level recovery is difficult.

8 Captioning Additional Qualitatives

Figure 2 shows qualitative captioning results obtained by conditioning Llama 3.2 1B, Gemma 2 2B, and Qwen3 0.6B on SLS embeddings. Across models of different scales, the generated captions remain consistent and well aligned with the visual semantics of the input SVGs, capturing object identity, structure, and stylistic cues. Overall, captions generated from SLS embeddings demonstrate strong semantic alignment with SVG content and significantly outperform those based on DeepSVG's encoder, which often fails to produce meaningful descriptions. While CLIP achieves superior caption quality due to large-scale pretraining, SLS remains competitive, showing that compact path-level embeddings effectively support SVG-to-text generation.

9 Latent Space Interpolation and Sampling

Although SLS is trained solely for reconstruction, the learned latent space supports meaningful interpolation and local sampling. Figure 5 shows spherical linear interpolation (SLERP) between pairs of path embeddings, producing smooth transitions in both geometry and appearance. Furthermore, randomly sampling embeddings from the local neighborhood of real paths yields valid and visually plausible SVG paths (Figure 6), demonstrating that the learned latent space is locally smooth and robust.

10 Retrieval Additional Qualitatives

Figure 7 presents qualitative retrieval examples. Compared to DINOv2, CLIP, and DeepSVG, SLS yields more semantically consistent top-5 retrieval results.

11 Limitations

Our method operates at the path level and does not explicitly model interactions across multiple paths. This is a deliberate design choice, as modeling cross-path dependencies falls outside the scope of the present work. Exploring mechanisms to aggregate path-level embeddings into a dense, image-level representation – such as through a dedicated aggregation module – constitutes a promising direction for future research. In addition, paths are truncated to a maximum length of 2048 characters for computational efficiency, and extending the approach to longer or more complex sequences is an avenue for future work.

For captioning, we employ a lightweight linear adapter to assess the intrinsic quality of the learned embeddings; more expressive cross-modal fusion strategies may yield additional gains. Finally, the reconstruction-based objective focuses on geometric fidelity and latent compactness, and integrating complementary semantic or contrastive objectives could enhance alignment in downstream tasks.

References

1. Bakshi, Ainesh and Indyk, Piotr and Jayaram, Rajesh and Silwal, Sandeep and Waingarten, Erik: Near-linear time algorithm for the chamfer distance. In: NeurIPS (2023)
2. Holtzman, Ari and Buys, Jan and Du, Li and Forbes, Maxwell and Choi, Yejin: The curious case of neural text degeneration. ICLR (2020)
3. Zhang, Y., Li, M., Long, D., Zhang, X., Lin, H., Yang, B., Xie, P., Yang, A., Liu, D., Lin, J., Huang, F., Zhou, J.: Qwen3 embedding: Advancing text embedding and reranking through foundation models (2025)

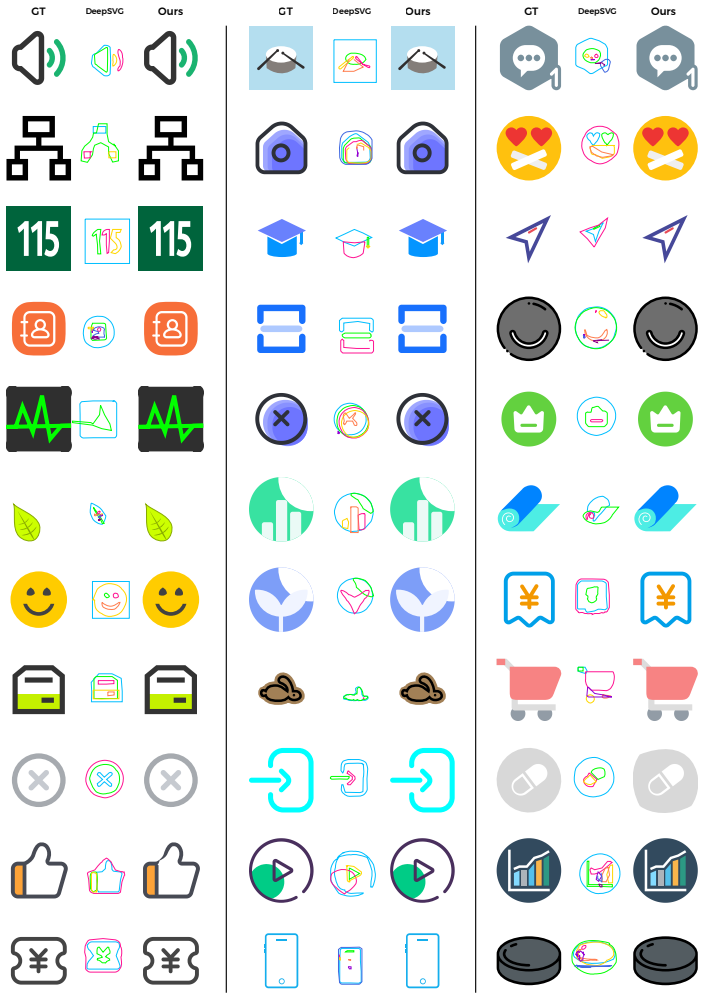

Fig. 1: Qualitative results on image reconstruction. Each separated column shows three samples from left to right: ground truth (GT) SVG images, DeepSVG reconstructions, and SLS (Ours) reconstructions.

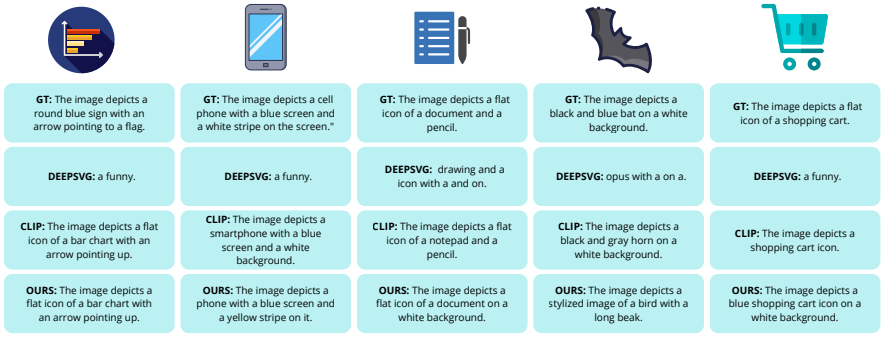

Fig. 2: Qualitative results on SVG captioning. We use Llama3.2 1B (first images row), Qwen3 0.6B (second images row) and Gemma2 2B (last images row) as language model and compare different image encoders: ground truth SVG (first row), DeepSVG, CLIP, and SLS (Ours). Each column represents a different example.

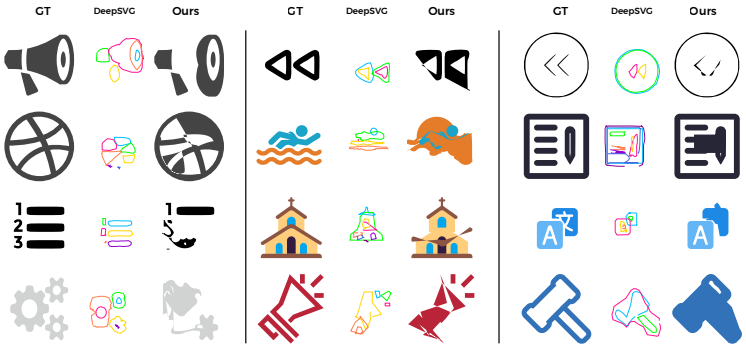

Fig. 3: Failure cases of SLS reconstruction. Each separated column shows three samples from left to right: ground truth (GT) SVG images, DeepSVG reconstructions, and SLS (Ours) reconstructions.

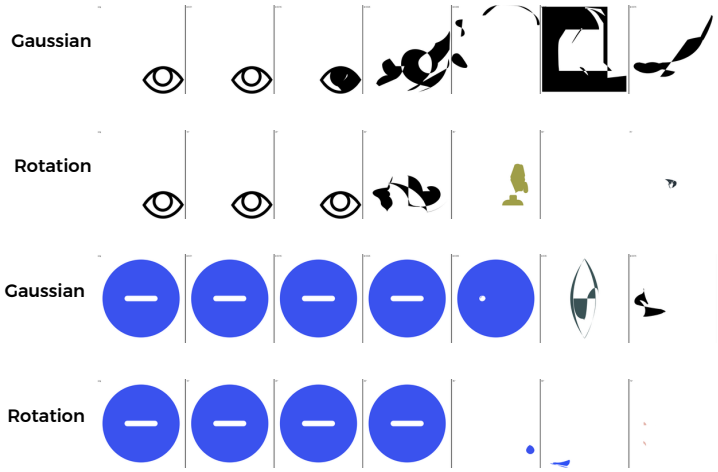

Fig. 4: Qualitative results on noise robustness. SVG paths subject to increasing rotation (up to 30°) and Gaussian noise ($\sigma = 0.02$). From left to right: original SVG, perturbed inputs with increasing gaussian noise magnitude and rotation angle. Results demonstrate robust geometric invariances within the learned representation, maintaining reconstruction fidelity under significant perturbations.

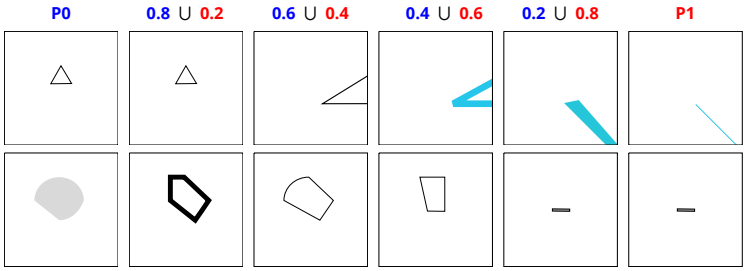

Fig. 5: Qualitative latent-space interpolation using spherical linear interpolation (SLERP). Each row interpolates between two SVG path embeddings (P_0 and P_1). The decoded paths evolve smoothly in geometry and style, illustrating the continuity of the learned latent space.

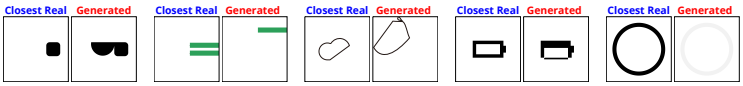

Fig. 6: Generation from latent-space neighborhoods. For each example, we randomly sample an embedding from the local neighborhood of a real path and decode it using the pretrained decoder. Generated paths remain valid and visually plausible while exhibiting meaningful variations with respect to their closest real counterparts.

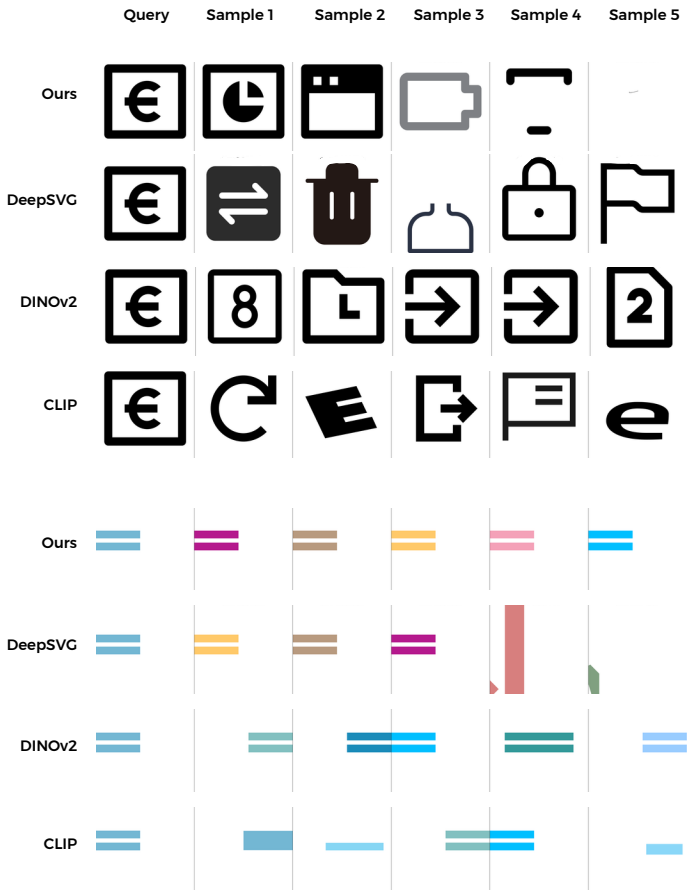

Fig. 7: Qualitative results on SVG retrieval. Query SVG path (left column) with top-5 ranked retrievals from our method compared to DeepSVG, DINOv2, CLIP baselines. Our method produces more semantically consistent results, demonstrating superior retrieval quality.