A Scalable Vector Graphics Path Auto-Encoder

Anonymous CVPR submission

Paper ID 10880

Abstract

Scalable Vector Graphics (SVG) offer resolution-independent, interpretable, and editable visual content, yet their symbolic and structured nature poses challenges for deep learning methods. Existing Transformer-based approaches model entire SVG files as long textual sequences, limiting scalability and conflating geometry, style, and structure. In this work, we propose SPE (SVG Path Encoder), the first Transformer-based autoencoder designed to learn compact, path-level representations of SVGs. SPE is trained from scratch to learn a continuous latent space that captures both geometric and stylistic attributes while reducing sequence length by orders of magnitude. The learned embeddings lie on a normalized hypersphere, supporting efficient vector-space operations such as similarity search, interpolation, and composition. We demonstrate that SPE enables scalable and semantically meaningful downstream applications – including path retrieval, SVG captioning via language model conditioning, and smooth geometric manipulation – without relying on handcrafted geometric priors or raster intermediates. Experiments show that path-level embeddings preserve geometric fidelity, reduce computational cost, and provide a unified latent space bridging symbolic vector markup and continuous representation learning.

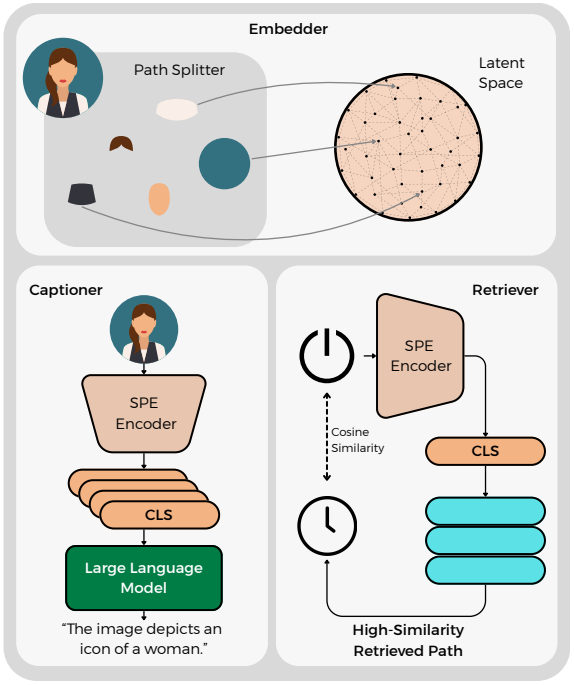

Figure 1. Overview of the proposed approach: SPE learns latent dense representations of SVG paths (top), which can be employed in a variety of downstream tasks *e.g.*, captioning and K-retrieving (bottom).

1. Introduction

Vector graphics have become a fundamental tool for scalable, resolution-independent visual design, powering everything from iconography and UI elements to data visualization and scientific illustrations. Unlike raster images, which discretize color over a pixel grid, vector graphics describe geometry symbolically through sequences of drawing commands, coordinates, and style attributes encoded in SVG (Scalable Vector Graphics) format. This symbolic nature makes them inherently interpretable and editable, but also poses unique challenges in the design of Deep Learning approaches for understanding and generation: SVG files are long, structured, and heterogeneous sequences that intertwine geometry with rendering semantics.

Recent progress in Large Language Models (LLMs) and sequence modeling has created a growing interest in treating SVG markup as a form of textual data. Works such as DeepSVG [5], IconShop [31], and vHector [15] have demonstrated that Transformer-based architectures can learn powerful generative priors over SVG code, producing coherent vector graphics or converting between text and SVG representations. However, these approaches take the entire SVG markup as input or output. This leads to substantial computational cost and limits scalability for applications such as retrieval, captioning, or reasoning over large corpora of vector content. Also, modeling SVGs as monolithic sequences conflates geometric primitives and struc-

CVPR #10880

tural hierarchy, hindering the emergence of more compact and semantically meaningful representations.

Drawing from these considerations, we shift the focus from whole-file to path-level representation learning, introducing a framework for learning compact embeddings of individual SVG paths. Each path corresponds to a coherent geometric stroke or shape, and can thus be treated as a fundamental compositional unit. By operating at this granularity, we enable efficient downstream processing, retrieval, captioning, or manipulation without requiring access to full-sequence SVG markup.

Our approach, which we term **SPE** (short for **S**VG **P**ath **E**ncoder), tokenizes path commands and arguments as discrete symbols, and learns an autoencoding Transformer that maps them into a continuous latent space. This latent space is trained from scratch to capture both the geometry and style of each path while reducing the sequence length by orders of magnitude. The resulting embeddings lie on a normalized hypersphere, allowing direct application of vector-space operations such as cosine similarity, interpolation, and linear composition.

Unlike prior work that relies on handcrafted geometric representations or raster intermediates, we treat SVG paths as text sequences and learn compact path-level representations in an end-to-end manner, using a reconstruction loss to preserve syntactic validity. During training, we inject Gaussian noise into the latent vectors to promote smoothness and regularization, and observe that the learned space exhibits highly consistent norms – a property we exploit for normalization and downstream applications. Through this design, SPE provides a dense, path-level embedding space for SVG graphics, connecting symbolic vector markup and continuous latent modeling.

We assess the quality of the SPE embedding space across a diverse set of tasks. In particular, we show that SPE enables efficient content-based search in path retrieval. When coupled with a pre-trained language model, SPE supports SVG image captioning, where path embeddings serve as visual tokens conditioning text generation. Finally, we show that latent interpolation and perturbation in the embedding space yield smooth geometric transformations, revealing structured and semantically meaningful manifolds.

In summary, the contributions of this work are as follows:

- We introduce SPE, the first Transformer-based autoencoder for SVG paths, which learns compact and expressive latent representations directly from tokenized path sequences.
- We propose a text-based tokenizer for path commands and arguments, enabling scalable training without handcrafted geometric priors.
- We demonstrate that the learned latent space supports retrieval, captioning, and manipulation of vector graphics through simple vector-space operations.
- We show that path-level embeddings substantially reduce sequence length and computational cost compared to full-SVG models, while preserving geometric and stylistic fidelity.

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 [9].

Image-to-Vector. Another line of work focuses on recovering vector graphics from raster images. Im2Vec [22] predicts parametric primitives to approximate shapes, while VectorGrimoire [7] improves geometric fidelity through richer curve modeling. Other methods, such as DualVector [17], DeepVecFont [30] and [18] 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 naturally 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 [31], vHector [15], LLM4SVG [32], OmniSVG [34], SVGen [29]) 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.* [23] 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 rep-

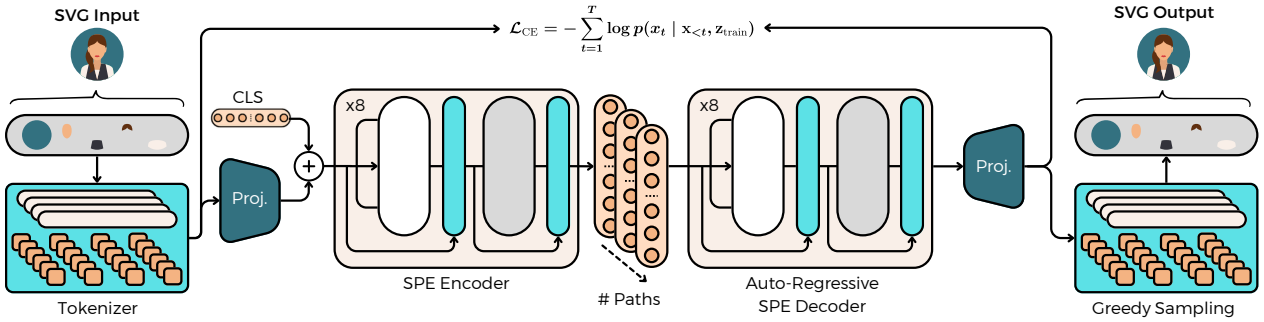

Figure 2. Overview of SPE architecture. SVG paths are encoded into latent representations via CLS tokens, with Gaussian noise injection during training for regularization. The autoregressive decoder reconstructs paths through greedy sampling decoding. Embeddings on the unit hypersphere enable efficient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

resentation 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 [33] combines fixed symbolic representations with learnable latent matrices to improve scalability, though it still relies on predefined structural constraints. SuperSVG [12] learns SVG representations of superpixel image regions for RGB vectorization, operating at a different granularity than whole-image methods. Other work, such as NeuralSVG [25], 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 [26] and SONAR [20] construct unified multilingual embedding spaces that enable cross-lingual transfer through shared semantic representations. Similarly, LCM (Large Concept Models) [14] 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 consist of sequences of discrete drawing commands and parameters. We first train a path-specific tokenizer to convert SVG path elements into discrete tokens, covering both command types and numeric arguments. The tokenizer operates on individual paths; compound SVG objects are decomposed into independent path sequences. Given the resulting token sequence $\mathbf{x} = (x_1, \ldots, x_T)$, our goal is to learn an encoder $E(\cdot)$ that maps $\mathbf{x}$ to a dense latent representation $\mathbf{z} \in \mathbb{R}^d$, and a decoder $D(\cdot)$ that reconstructs the original sequence. To standardize and reduce the average path lengths we followed the same tokenizer architecture proposed in [15].

The encoder-decoder architecture follows the standard autoencoder framework for learning compressed representations, enabling downstream applications that benefit from vector-space reasoning over structured path data, reducing sparsity. Unlike previous works [5, 35], which employ specialized geometric representations, we treat SVG paths as text sequences and train a purely text-based tokenizer that handles commands, coordinates, and style attributes uniformly. We then rely on the encoder-decoder architecture to learn appropriate representations of the underlying geometric and stylistic structure. This approach yields the first 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. SPE Architecture

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

Encoder. The encoder embeds token sequences using learned token embeddings and sinusoidal positional encodings. We prepend a learnable [CLS] token to produce an augmented sequence, following the ViT approach:

$$\tilde{\mathbf{x}} = [\text{CLS}; x_1; \ldots; x_T]. \quad (1)$$

CVPR #10880

The encoder processes this sequence through multiple Transformer layers, and we extract the final hidden state corresponding to the [CLS] token as our latent representation $\mathbf{z}$.

Decoder. The decoder reconstructs the original token sequence conditioned on $\mathbf{z}$. We prepend $\mathbf{z}$ as a prefix token and apply a causal Transformer over the autoregressive input. A linear language modeling head predicts vocabulary logits at each timestep. Unlike natural language generation, where multiple valid continuations exist and sampling strategies can improve output diversity [8, 11], SVG path sequences are deterministic and syntactically strict: each token position has 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:

$$\hat{x}_t = \arg\max_{x \in \mathcal{V}} p(x \mid \mathbf{x}_{<t}, \mathbf{z}), \tag{2}$$

where $\mathcal{V}$ is the vocabulary. This deterministic approach ensures syntactically valid path reconstructions and avoids the risk of sampling errors that could produce malformed SVG paths.

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

$$\mathbf{z}_{\text{train}} = \mathbf{z} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}). \tag{3}$$

The decoder then predicts token logits $\mathbf{y} \in \mathbb{R}^{T \times N}$, and we apply cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\sum_{t=1}^{T} \log p(x_t \mid \mathbf{x}_{<t}, \mathbf{z}_{\text{train}}). \tag{4}$$

This noise injection regularization promotes smoothness in the latent space and improves generalization.

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

$$\mathbf{z}_{\text{norm}} = \frac{\mathbf{z}}{\|\mathbf{z}\|_2}. \tag{5}$$

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. When reconstruction is required, we rescale by the empirical mean before feeding to the decoder:

$$\mathbf{z}_{\text{decoder}} = \mu_{\|\mathbf{z}\|} \cdot \mathbf{z}_{\text{norm}}, \tag{6}$$

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 magnitude range it was trained on.

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:

$$\text{sim}(\mathbf{z}_q, \mathbf{z}_i) = \frac{\mathbf{z}_q^\top \mathbf{z}_i}{\|\mathbf{z}_q\| \|\mathbf{z}_i\|}. \tag{7}$$

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:

$$\mathbf{e}_i = \mathbf{W}_{\text{proj}} \mathbf{z}_i, \tag{8}$$

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:

$$p(\text{caption} \mid \text{SVG}) = \prod_{t=1}^{T_{\text{cap}}} p(w_t \mid w_{<t}, \mathbf{e}_1, \ldots, \mathbf{e}_N), \tag{9}$$

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

4. Experiments

4.1. Experimental setup

Dataset. To train and evaluate our model, we utilize a combination of publicly available SVG datasets, namely StarVector [23], HeisenVec [15], ColorSVG [6],

CVPR #10880

Table 1. Reconstruction performance comparison at image and path levels, with varying architectural choices.

Method	Image Level				Path Level		
	MSE-sim ↑	SSIM ↑	LPIPS ↓	DINOv2 ↑	mIoU ↑	$BLEU_5$ ↓	METEOR ↓
DeepSVG [5]	76.03	67.85	53.99	49.86	–	–	–
• retrained	76.05	67.85	53.31	57.95	–	–	–
SPE (Ours)	**94.73**	**92.58**	**10.48**	**89.44**	**81.74**	43.88	**1.32**
• ℓ_2 regularization	86.51	81.05	28.63	72.41	66.79	**37.95**	9.98
• logit scale	71.98	58.63	62.38	36.83	5.63	98.91	77.45
• logit scale + ℓ_2 norm	71.72	59.65	63.68	36.98	5.45	97.06	71.73

and SVGX-Core [32]. 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 [15]. 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 SPE from scratch for 290k steps using a batch size of 2048 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 expansion ratio of 2, the overall number of parameters for both encoder and decoder is 135M. Training is performed on 32 NVIDIA A100 GPUs.

Tokenization. We trained a data-driven tokenizer using SentencePiece [13] with Byte-Pair Encoding (BPE) on 43.75% of the total path corpus. The vocabulary size of 448 tokens was selected to optimize the trade-off between compression ratio and information density, given the limited variability of SVG path commands. We also augment the vocabulary with a set of special tokens, namely `<s>`, `</s>`, `<pad>`, `<unk>`, and `<mask>`.

Crucially, we adopt a data-driven tokenization strategy where token boundaries emerge from corpus statistics rather than being constrained by SVG command structure. This allows the model to learn optimal sub-word units without human bias: tokens may represent complete commands, command types, or frequently co-occurring numeric patterns based on their prevalence in the training data.

4.2. Reconstruction task

To validate the quality of SPE, we measured reconstruction performance at two levels: (i) *image level*, comparing original and reconstructed images using classical Computer Vision metrics (MSE-similarity, SSIM [37], LPIPS [36], DINOv2-Similarity [19]), and (ii) *path level*, analyzing token distribution with BLEU [28] and METEOR [3], 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 SPE 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, SPE 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.

Further, given that SPE 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 our final approach, confirming that constant ℓ_2 norm peculiarity emerges spontaneously. Figure 3 provides qualitative evidence that SPE consistently outperforms DeepSVG in reconstruction quality. This improvement is attributed to our model's direct end-to-end approach, which fuses style management and command/argument embedding components into a unified representation.

4.3. 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

CVPR #10880

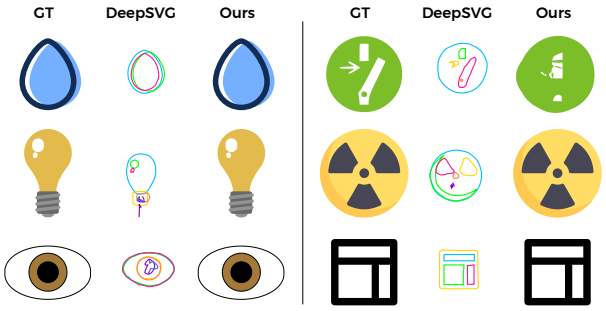

Figure 3. Qualitative results on image reconstruction. GT columns refer to the starting SVG image.

Table 2. Retrieval performance of SPE against other encoders.

Model	Raster	MSE-Sim ↑	LAB Dist. ↓	$BLEU_5$ ↓	METEOR ↓
CLIP [21]	✓	98.67	11.66	89.42	65.88
DINOv2 [19]	✓	98.86	14.68	89.23	65.45
DeepSVG [5]	✗	81.45	32.06	87.11	56.73
SPE	✗	**89.95**	**25.27**	**79.09**	**52.22**

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 the highest cosine similarity to the query) for evaluation. Since DeepSVG and SPE can decode embeddings back into SVG paths, while raster-based encoders such as DINOv2 *base* [19] and CLIP *vit-base-patch32* [21] cannot, we used the unique indices of the retrieved embeddings to recover the corresponding original SVG paths in all cases.

Retrieval quality was assessed through both visual and syntactic metrics. Visual similarity was measured using MSE-Sim, computed using mean squared error between rasterized query and retrieved paths, and LAB distance, which captures color differences in the fill and stroke attributes in CIE-LAB space. Syntactic fidelity was evaluated using $BLEU_5$ and METEOR, which quantify token-level overlap between the SVG command sequences of the query and retrieved paths.

Results. Table 2 reports the path retrieval results, in comparison with DeepSVG [5], a CLIP encoder [18] and a DINOv2 [19] encoder. The proposed SPE encoder achieves the best overall retrieval performance among the SVG-based approaches, surpassing DeepSVG by a large margin in both visual and syntactic metrics. In particular, SPE yields a higher MSE-Sim and lower LAB distance, indicating improved visual consistency between retrieved and query paths. Similarly, as distance metrics where lower values indicate better performance computed as $1 - BLEU_5$ and $1 - METEOR$, its scores confirm stronger preservation of structural and token-level fidelity.

While raster-based encoders such as CLIP and DINOv2 achieve higher visual similarity due to their image-level training objectives, they exhibit limited sensitivity to the underlying vector structure. In contrast, both SVG-based models generalize better across geometric and syntactic variations, capturing path-level semantics beyond purely visual appearance. As shown qualitatively in Figure 4, SPE consistently retrieves more structurally similar paths compared to DeepSVG. Overall, these results demonstrate that SPE provides an optimal trade-off between visual and structural similarity, establishing it as the most effective encoder for vector path retrieval.

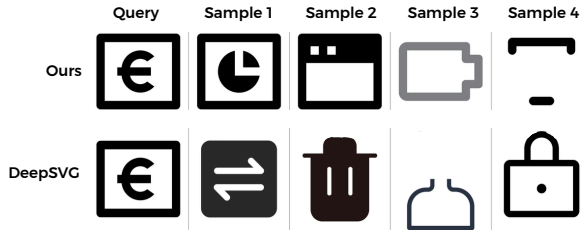

Figure 4. Qualitative path retrieval results. Query paths (left) and top-ranked retrievals ordered by cosine similarity for SPE (Ours) and DeepSVG [5]

4.4. SVG Image Captioning

Experimental Setup. We then move to evaluating the quality of SPE embeddings through SVG-to-text generation. Following standard vision-language practices, we train lightweight projection layers that map SPE 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 [2] for efficient inference, Llama 3.2 1B [1] for balanced performance, and Gemma 2 2B [27] for enhanced caption quality.

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. BLEU [28] measures n-gram overlap with reference captions, while ROUGE [16] emphasizes recall for longer texts. METEOR [3] accounts for synonyms and paraphrasing. CLIP-Score [10] quantifies the semantic alignment between the generated caption and the rasterized SVG image using CLIP embeddings. To mitigate the influence of imperfect automatically generated captions (see Section 4.1), we also report a Scaled CLIP-Score, which normalizes the CLIP-Score of predicted captions by that of the reference caption, as follows:

$$\text{ScaledCLIP} = \frac{\text{CLIP-Score}_{\text{pred}}}{\text{CLIP-Score}_{\text{ref}}}. \quad (10)$$

CVPR #10880

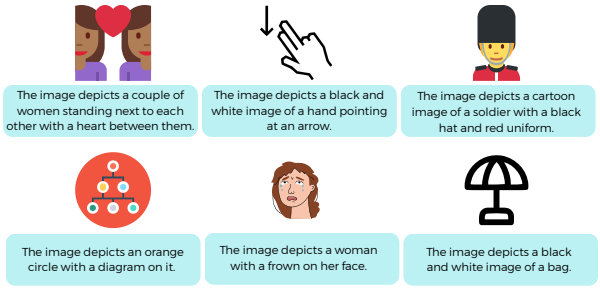

Figure 5. Qualitative captioning samples of Llama 3.2 1B [1] as Large Language Model using SPE as SVG Image encoder.

As can be seen, this normalization makes results more comparable across samples with varying reference quality.

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 SPE to DeepSVG [5], the plain XML code, and CLIP [21]. Here, CLIP [21] 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 (often more than 1000 tokens per path) make learning prohibitive. SPE approach achieves significantly better performance (+4.8 CLIP-Score, +29.4 BLEU_5) using dense embeddings, validating the effectiveness of our learned compression for downstream tasks. Further, SPE consistently outperforms DeepSVG across all metrics and LLMs, with substantial average gains in CLIP-Score (+3.02), BLEU_5 (+19.6), and METEOR (+24.95), demonstrating superior semantic richness while maintaining invertibility. Qualitative results for SVG captioning are presented in Figure 5, where SPE serves as the visual encoder paired with Llama 3.2 1B as the language model, demonstrating strong caption quality.

4.5. Embedding Space Analysis

Robustness to Noise. 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

$$\tilde{\mathbf{z}} = \mathbf{z} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}), \tag{11}$$

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:

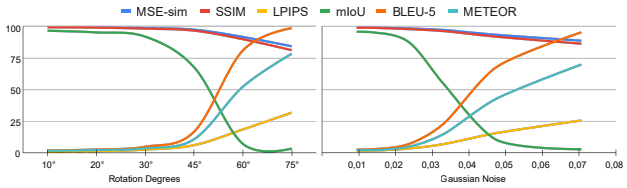

Figure 6. Reconstruction performance of SPE under varying noise levels. Left: rotational perturbations applied to input vectors. Right: additive Gaussian noise.

$$\tilde{\mathbf{z}} = \|\mathbf{z}\| \left[\cos(\Delta\theta)\, \hat{\mathbf{z}} + \sin(\Delta\theta)\, \hat{\mathbf{y}}\right],$$

$$\hat{\mathbf{z}} = \frac{\mathbf{z}}{\|\mathbf{z}\|}, \quad \hat{\mathbf{y}} = \frac{\mathbf{r} - (\mathbf{r}^\top \hat{\mathbf{z}})\hat{\mathbf{z}}}{\|\mathbf{r} - (\mathbf{r}^\top \hat{\mathbf{z}})\hat{\mathbf{z}}\|}, \quad \mathbf{r} \sim \mathcal{N}(0, I_d);$$

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 SPE decoder and compared it to the original using both visual and structural metrics. Specifically, we computed MSE, SSIM, and LPIPS over rasterized renderings, and IoU, BLEU_5, and METEOR over SVG path commands. 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 SPE 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.

Clustering Analysis. To evaluate the structural quality and representational geometry of different embedding models, we perform K-Means clustering in the embedding space and compute cluster quality measures. In particular, we employ a set of different metrics that capture a distinct geometric property of the embedding space. Cohesion measures the average intra-cluster similarity, while Density assesses local compactness around samples. Centroid Separation quantifies inter-cluster distinctiveness, with higher values indicating more separated clusters. Participation Ratio and Isotropy describe the intrinsic dimensionality and directional uniformity of the feature space, respectively, both desirable when high. Cosine Diversity evaluates the angular spread among feature vectors, reflecting representational richness. Finally, the Silhouette [24] and Calinski–Harabasz [4] assess the global clustering quality and separation-to-compactness ratio.

For simplicity, we also report an aggregated normalized score. Let $M = \{m_1, m_2, \ldots, m_n\}$ be the set of metrics computed for each model, and let $m_i^{\max}$ denote the maximum value of metric m_i across all compared models. The

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

Image Encoder	SVG Native	LLM	CLIP-Score ↑	CLIP-Score scaled ↑	$BLEU_5$ ↑	METEOR ↑	$ROUGE_1$ ↑	$ROUGE_2$ ↑
CLIP [21]	✗	Qwen3 0.6B [2]	29.48	1.00	49.44	69.12	75.23	63.48
		Gemma2 2B [27]	29.03	0.98	24.78	53.72	55.70	41.40
		Llama 3.2 1B [1]	29.57	1.00	54.67	71.87	77.39	67.18
XML	✓	Qwen3 0.6B [2]	21.95	0.75	10.38	32.14	38.84	23.49
		Gemma2 2B [27]	22.73	0.78	12.62	34.48	44.31	26.67
		Llama 3.2 1B [1]	23.99	0.83	12.31	32.45	43.25	28.13
DeepSVG [5]	✓	Qwen3 0.6B [2]	25.42	0.87	27.30	40.55	48.96	37.02
		Gemma2 2B [27]	25.39	0.87	18.71	37.07	44.95	27.49
		Llama 3.2 1B [1]	23.25	0.79	18.77	35.82	44.77	27.22
SPE (Ours)	✓	Qwen3 0.6B [2]	27.86	**0.95**	46.17	65.29	70.76	59.53
		Gemma2 2B [27]	27.33	0.94	30.03	56.89	59.44	45.78
		Llama 3.2 1B [1]	**27.92**	**0.95**	**47.39**	**66.12**	**71.59**	**60.49**

Table 4. Quantitative evaluation of embedding space quality. Comparison across clustering metrics (Silhouette, Calinski-Harabasz) and embedding distribution properties (isotropy, diversity, cohesion). SPE attains the highest aggregate score, indicating well-structured latent representations.

Model	Best k ↑	Cohesion ↓	Density ↓	Centr. Sep. ↑	Part. Ratio ↑	Isotropy ↑	Cosine Diversity ↑	Silhouette ↑	Calinski-Harabasz ↑	Score ↑
CLIP	**8**	0.960	0.197	0.014	17.802	0.136	0.105	0.074	708	1.959
DINOv2	2	0.790	0.342	0.211	16.324	0.119	0.440	0.262	1861	3.138
DeepSVG	2	0.946	**0.079**	**0.321**	2.409	0.018	0.256	**0.713**	**13424**	4.353
SPE	**8**	**0.525**	0.909	0.114	**189.147**	**0.380**	**0.861**	0.004	147	**4.659**

aggregated normalized score S is defined as:

$$S = \sum_{i=1}^{N} \frac{m_i}{m_i^{\max}}, \tag{12}$$

where m_i denotes the value of the i-th metric for a given model, $m_i^{\max}$ is the maximum value of that metric across all models , and N the number of metrics considered.

As can be seen in Table 4, SPE achieves the highest overall score (4.66), followed by DeepSVG (4.35), DINOv2 (3.14), and CLIP (1.96). While SPE and DeepSVG each excel in four metrics (highlighted in bold), SPE achieves a stronger balance across dimensions (embedding distribution is shown in Figure 7). In particular, its high participation ratio, isotropy, and cosine diversity indicate a high-rank, isotropic, and semantically diverse representation. Conversely, DeepSVG shows excellent silhouette and centroid separation, suggesting clear but possibly over-partitioned clusters. Overall, the results suggest that SPE produces the most expressive and geometrically well-balanced embedding space among all compared models. An RGB-based image encoder may struggle in representing SVG images due to a domain shift with respect to their training distribution.

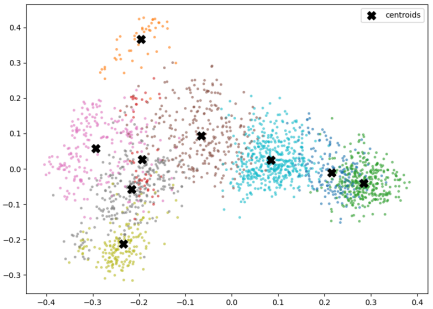

Figure 7. 2D PCA projection of the cluster centroids obtained from the SPE embeddings.

5. Conclusion

In this work, we introduced SPE (SVG Path Encoder), the first Transformer-based autoencoder designed to learn compact and expressive path-level representations of SVGs. By shifting the modeling focus from entire SVG files to individual paths, SPE embeddings achieve semantic interpretability and computational scalability, addressing a critical gap in vector graphics understanding. Through extensive experiments, we demonstrated that SPE learns a well-structured and isotropic latent space, supporting diverse downstream applications such as path retrieval, SVG captioning, and geometric manipulation. The learned latent space also exhibits strong robustness to noise, semantic coherence, and cluster regularity, confirming the emergence of meaningful manifolds for vector content. Future directions include extending this framework to hierarchical SVG generation and exploring its integration with large-scale vision-language models for creative design tasks. To foster further research in this emerging domain, we will release model weights, training and inference codebase.

References

[1] Aaron Grattafiori et al. The Llama 3 Herd of Models. In *arXiv*, 2024. 6, 7, 8

[2] An Yang et al. Qwen3 Technical Report, 2025. 6, 8

[3] Satanjeev Banerjee and Alon Lavie. 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. 5, 6

[4] T. Caliński and J Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 1974. 7

[5] Carlier, Alexandre and Danelljan, Martin and Alahi, Alexandre and Timofte, Radu. Deepsvg: A hierarchical generative network for vector graphics animation. *NeurIPS*, 2020. 1, 2, 3, 4, 5, 6, 7, 8

[6] Chen, Zehao and Pan, Rong. SVGBuilder: Component-Based Colored SVG Generation with Text-Guided Autoregressive Transformers. 2024. 4

[7] Cipriano, Marco and Feuerpfeil, Moritz and De Melo, Gerard. Vector Grimoire: Codebook-based Shape Generation under Raster Image Supervision. 2025. 2

[8] Fan, Angela and Lewis, Mike and Dauphin, Yann. Hierarchical neural story generation. *arXiv preprint*, 2018. 4

[9] Ha, David and Eck, Douglas. A neural representation of sketch drawings. *arXiv preprint*, 2017. 2

[10] 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. 6

[11] Holtzman, Ari and Buys, Jan and Du, Li and Forbes, Maxwell and Choi, Yejin. The curious case of neural text degeneration. *ICLR*, 2020. 4

[12] 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. 3

[13] "Kudo, Taku and Richardson, John". "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing". 2018. 5

[14] 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. 3

[15] Leonardo Zini and Elia Frigieri and Sebastiano Aloscari and Lorenzo Baraldi. vHector and HeisenVec: Scalable Vector Graphics Generation Through Large Language Models. In *NeurIPS*, 2025. 1, 2, 3, 4, 5

[16] Lin, Chin-Yew and Och, Franz Josef. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. 2004. 6

[17] 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. 2

[18] 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*, pages 7930–7939, 2019. 2, 6

[19] 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. 5, 6

[20] Paul-Ambroise Duquenne and Holger Schwenk and Benoit Sagot. SONAR: Sentence-Level Multimodal and Language-Agnostic Representations. In *arXiv*, 2023. 3

[21] 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. 6, 7, 8

[22] Reddy, Pradyumna and Gharbi, Michaël and Lukac, Michal and Mitra, Niloy J. Im2Vec: Synthesizing Vector Graphics without Vector Supervision. In *CVPR*, 2021. 2

[23] 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. 2, 4

[24] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. 1987. 7

[25] 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. 3

[26] NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. No Language Left Behind: Scaling Human-Centered Machine Translation. In *arXiv*, 2022. 3

[27] 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. 6, 8

[28] vPapineni, Kishore and Roukos, Salim and Ward, Todd and Zhu, Wei-Jing. Bleu: a method for automatic evaluation of machine translation. 2002. 5, 6

[29] 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. 2

[30] 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

[31] Wu, Ronghuan and Su, Wanchao and Ma, Kede and Liao, Jing. IconShop: Text-Guided Vector Icon Synthesis with Autoregressive Transformers. *ACM TOG*, 2023. 1, 2

[32] 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, 5

[33] 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. 3

[34] Yang, Yiying and Cheng, Wei and Chen, Sijin and Zeng, Xianfang and Zhang, Jiaxu and Wang, Liao and Yu, Gang and Ma, Xingjun and Jiang, Yu-Gang. OmniSVG: A Unified Scalable Vector Graphics Generation Model. *NeurIPS*, 2025. 2

[35] Zhang, Peiying and Zhao, Nanxuan and Liao, Jing. Text-to-Vector Generation with Neural Path Representation. *ACM TOG*, 2024. 3

[36] 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. 5

[37] 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. 5