vHector and HeisenVec: Scalable Vector Graphics Generation Through Large Language Models

Leonardo Zini[*] **Elia Frigieri**[*] **Sebastiano Aloscari** **Lorenzo Baraldi**

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

Abstract

We introduce HeisenVec, a large-scale dataset designed to advance research in vector graphics generation from natural language descriptions. Unlike conventional image generation datasets that focus on raster images, HeisenVec targets the structured and symbolic domain of Scalable Vector Graphics (SVG), where images are represented as sequences of drawing commands and style attributes. The dataset comprises 2.2 million SVGs collected from different online sources, each paired with four complementary textual descriptions generated by multi-modal models. To ensure structural consistency and efficiency for autoregressive modeling, all SVGs are standardized through a pre-processing pipeline that unifies geometric primitives as paths, applies affine transformations, and compresses syntax via custom tokens set. HeisenVec exhibits broad coverage among visual styles and sequence lengths, with a substantial portion of samples exceeding 8,000 tokens, making it particularly well-suited for benchmarking long-context language models. Our benchmark enables rigorous evaluation of text-conditioned SVG generation, encourages progress on sequence modeling with symbolic outputs, and bridges the gap between vision, graphics, and language. We release the dataset, tokenization tools, and evaluation pipeline to foster further research in this emerging domain. The dataset and the code for testing our parsing, standardization, and tokenization method are available at this link.

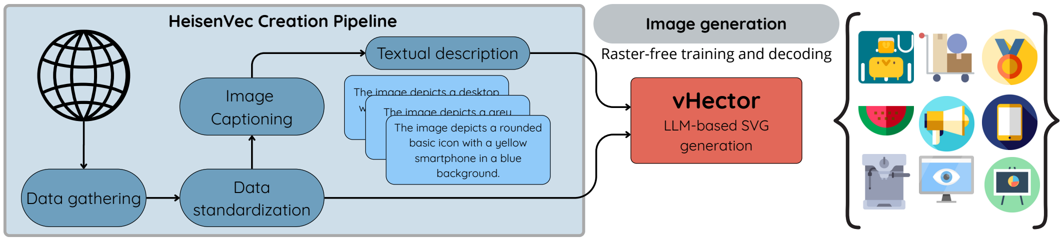

HeisenVec Creation Pipeline
Data gathering
Data standardization
Image Captioning
Textual description
The image depicts a desktop w
The image depicts a grey
The image depicts a rounded basic icon with a yellow smartphone in a blue background.
Image generation
Raster-free training and decoding
vHector
LLM-based SVG generation

Figure 1: Overview of the HeisenVec pipeline. Raw image data is gathered and standardized, then captioned using image-to-text models. Textual descriptions are then used during the training of vHector family model to generate corresponding SVG graphics without relying on raster representations.

1 Introduction

Imagine a graphic designer who, late on a Friday afternoon, needs a coherent set of clean, resizable icons for a pitch deck – "a playful purple rocket, a minimalist shopping cart with a colorful stroke, and a hand-drawn coffee mug." At present, its quickest option is to prompt a raster text-to-image model, manually sift through dozens of pixel outputs, and then run the best candidates through an

[*] Equal contribution.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

off-the-shelf vectorizer. The result is an SVG that looks acceptable at thumbnail scale, but whose underlying path structure is a tangled forest of thousands of tiny Bézier curves. Editing a single anchor point or recoloring an entire stroke invariably becomes a frustrating exercise. This disconnect – between the symbolic precision expected by vector-native workflows and the bitmap-centric lineage of most modern generative pipelines – highlights a fundamental gap in current research. Contemporary text-to-SVG systems either (i) generate a raster first and vectorize later, entangling two separate problems and inheriting the weaknesses of both, or (ii) remain confined to narrow domains, producing only short, template-style markup for simple icons. The field lacks a large-scale, vector-native benchmark that can drive progress toward controllable, semantically rich SVG generation.

Motivation. Latent Diffusion Models (LDMs) dominate creative image synthesis today, but their reliance on pixel-space supervision means that structure must be recovered after the fact – typically via differentiable renderers like DiffVG (17), or by gradient-guided path optimization as in VectorFusion (15). These approaches incur heavy computing, require brittle post-processing, and obscure the inherent human readability of SVGs. Large Language Models (LLMs), by contrast, excel at producing long, discrete sequences with strict syntax – precisely the characteristics of SVG markup. Early work such as IconShop (34) and LLM4SVG (38) hinted at this synergy, yet trained on relatively small corpora and were limited to less than 2k tokens. Without a truly large, diverse, and long-sequence dataset, it is impossible to assess whether LLMs can scale to real-world graphic complexity.

Overview of the dataset and benchmark. We address this deficiency with HeisenVec, a curated corpus of **2.2 million** SVG images paired with four complementary captions each obtained from BLIP2 (16), Florence 2 (36), and Idefics 3 (13). Every file has been standardized through a deterministic pipeline that: (i) converts all primitives to canonical `<path>` commands; (ii) collapses group hierarchies, resolves styles, inlines transforms, and limits geometry to `M/A/C/L`; (iii) rescales to a 512×512 viewbox, quantizes coordinates, and canonicalizes colors; and (iv) filters outputs with a DINOv2 (19) similarity check to guarantee fidelity. The result is an order-of-magnitude leap in both length (mean 24k characters, up to 52k) and semantic spread over prior collections (Figures 3 and 4). We release HeisenVec alongside a rigorous text-to-SVG benchmark: prompts, evaluation scripts, and reference metrics (CLIP-S, Geometric Consistency, and Edit Distance to valid markup).

Baseline models. To demonstrate how LLMs profit from this scale, we introduce the vHector model family: fully autoregressive sequence-to-sequence transformers built on LLaMA 3.1/3.2 (7), augmented with 17 SVG-specific tokens. vHector-8B serves as a strong full-fine-tuned baseline for up to 8k-token graphics, while lightweight LoRA adapters extend the 3B backbone to 16k (vHector-long) and 32k (vHector-exlong) contexts. Across different evaluations, vHector substantially outperforms previous LLM-based pipelines in structure validity and prompt alignment – yet still leaves ample headroom, validating HeisenVec difficulty.

Figure 1 summarizes our contributions:

- **HeisenVec**: the first long-context million-scale dataset of richly captioned SVGs, processed into a compact, learnable token space and spanning up to 32k tokens per sample.
- **vHector**: a suite of LLM-based baselines for direct text-to-SVG generation, complete with long-context LoRA (11) adapters and explicit domain tokens that promote syntactic correctness and geometric coherence.
- **Benchmark and code**: end-to-end tooling – dataset loader, tokenizer, evaluation metrics, and training recipes to catalyze future research in vector-native generative modeling.

Collectively, these resources lay the groundwork for vector graphics generation that is scalable, editable, and directly conditioned on natural language – closing the gap between designer intent and vector output, nevertheless setting a new agenda for multi-modal foundation models.

2 Related Works

Latent Diffusion Models. The introduction of DiffVG (17), a differentiable renderer, represented a significant advancement, enabling direct optimization of SVG parameters via backpropagation using image-based losses like Score Distillation Sampling (1). Generative models based on Latent Diffusion Models (LDMs) (26) have become a prevalent approach for SVG generation. These methods typically synthesize raster images from textual prompts and subsequently vectorize them adapting conventional vectorization tools (18; 21; 20) or proposing novel approaches. Expanding upon this, VectorFusion (15) employs latent-space SDS gradients to refine SVG paths initialized with a predefined structure iteratively. SVGDreamer (40; 39) introduces more sophisticated initialization leveraging text-image activation maps, yet overall approach complexity increases computational overhead. Chat2SVG (33) proposed an SVG generation pipeline combining prompt expansion, zero-shot inference through LLM, and diffusion-based detail enhancements. Most recently, SVGFusion (37) substantially advanced generative modeling for SVGs by proposing a unified architecture combining diffusion transformers and autoencoder models.

These methods combine raster image generation with vectorization, merging two distinct research areas and resulting in outputs that draw on the strengths and weaknesses of both approaches (*e.g.*, they create the background by default). We firmly believe that vector graphic generation should instead be based on the direct generation of paths, rather than relying on their optimization as a post-processing step, due to their human-readability characteristic.

Image-to-SVG Generation. Complementary to text-driven approaches, image-to-SVG generation converts raster images into vector representations. Supervised methods (23; 2; 29; 3) directly learn this mapping through neural architectures. Recent advances address reconstruction accuracy and semantic preservation: SuperSVG (12) employs superpixel-based decomposition with a two-stage coarse-to-refinement framework using dynamic path warping loss, while Layered Vectorization (31) progressively reconstructs images as semantically-aligned layer-wise vectors through SDS-based simplification. StarVector (24) integrates LLMs with image encoders for direct image-to-SVG translation. These image-based methods excel at preserving visual fidelity and offer complementary insights to text-driven generation for comprehensive SVG asset construction.

Large Language Models for SVG Generation. The structured, human-readable nature of SVGs makes them particularly well-suited for generation through Large Language Models (LLMs), which can naturally process and generate markup languages. This paradigm enables direct path generation without requiring rasterization as an intermediate step, preserving the semantic structure and editability that define vector graphics.

Early transformer-based approaches, such as IconShop (34), demonstrated the feasibility of directly generating simplified SVG icons from textual descriptions using decoder architectures. These initial efforts established that the sequential nature of SVG code aligns well with autoregressive language modeling. LLM4SVG (38) significantly advanced this direction by fine-tuning LLMs on a medium-scale dataset, demonstrating how large-scale pretraining combined with domain-specific fine-tuning could markedly enhance generative capabilities for structured vector graphics. Their work, along with OmniSVG (41), highlighted the importance of dataset scale and quality in unlocking LLMs' potential for SVG generation.

The connection between LLMs and visual generation extends beyond SVG: Sun et al. (27) demonstrated that LLMs can generate complex rasterized images through appropriate tokenization strategies, providing valuable insights on how to adapt language models for visual content creation. This work inspired approaches to treating visual generation as a sequence modeling problem.

Other methods, such as CLIPDraw (6), tackle text-to-image synthesis by iteratively generating images at evaluation time, conditioned on CLIP encoders, demonstrating alternative ways to bridge language and visual modalities.

The LLM-based paradigm offers several distinct advantages for SVG generation: (*i*) direct generation of human-readable, editable code; (*ii*) natural handling of the hierarchical structure inherent in SVG markup; (*iii*) ability to leverage massive pretraining on code and structured data; and (*iv*) straightforward integration of textual conditioning through the model's native language understanding capabilities. However, challenges remain in generating long, complex SVG sequences that require extended context windows and maintaining geometric coherence across multiple path elements. Our work builds upon this foundation by addressing these challenges through large-scale dataset construction and systematic investigation of long-context generation dynamics.

SVG Datasets and Benchmarks. Recent efforts have introduced various text-to-SVG benchmarks and datasets. Chen et al. (5) released ColorSVG, consisting of 100k SVG images with categorical labels. Xing et al. (38) contributed SVGX-Core, comprising 250k simple SVG graphics paired with generated textual descriptions, alongside their LLM-based approach leveraging auxiliary vision-

Table 1: Comparison of SVG datasets in terms of scale, captioning, and sequence characteristics.

Dataset	# Images	# captioners	CLIP-S ↑	Avg. SVG len	SpreadScore ↑	Long seq.
ColorSVG	100k	1	21.13	11,251	16.30	✗
SVGX-Core	250k	3	26.54	1,614	23.87	✗
StarVector	2.2M	3	24.62	2,524	42.80	✗
HeisenVec	2.2M	3 (+1)	**28.11**	**24,075**	**68.54**	✓

language tasks. StarVector (24) scaled this work with approximately 2.2M images, though with narrower iconographic diversity. While these resources represent important steps forward, none offers the combination of scale, sequence length, and generality required to robustly support text-to-SVG generation for complex illustrative graphics without relying on rasterized intermediates, a gap our work aims to address.

3 HeisenVec dataset and text-to-SVG generation LLM-based

3.1 Task definition

We define the task of SVG generation from text as a conditional sequence modeling problem. The goal is to generate structured vector graphics in the form of SVG markup, directly from natural language prompts. Formally, given an input sequence of text tokens $\mathbf{x} = (x_1, \ldots, x_T)$, the model must generate a target sequence of SVG tokens $\mathbf{y} = (y_1, \ldots, y_S)$, where usually $S \gg T$ and each y_t belongs to a hybrid vocabulary that includes both natural language subwords and domain-specific tokens for vector operations. This setting follows an autoregressive, sequence-to-sequence paradigm:

$$P(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{S} p(y_t \mid \mathbf{x}, y_{<t}; \theta),$$

where θ are the model parameters. Unlike traditional sequence generation tasks, such as translation or summarization, the output domain S in our case is highly structured and semantically dense: each token y_t may correspond to a drawing command (`<|M|>`, `<|A|>`, `<|C|>`, `<|L|>`), a visual attribute (*e.g.*, `<|opacity|>`, `<|stroke|>`), or a numerical coordinate. All of which must compose a syntactically and visually valid SVG.

The resulting token sequence $\mathbf{y}$ corresponds to an SVG file that encodes a 2D vector image through a series of drawing primitives and style specifications. This representation is not only resolution-independent and human-readable but also highly concept-related. This aspect makes the SVG generation task particularly suited to language models with long-context capabilities.

In contrast to raster-based text-to-image generation, which often relies on a pixel-level output space or diffusion-based latent sampling, our approach retains the symbolic structure of the SVG and delegates to the model the responsibility of learning a valid and semantically aligned image composition in a pure vectorial output space.

3.2 Dataset standardization method and Analysis

Data collection. We sourced raw SVGs artwork from a variety of public repositories (*e.g.*, SVGrepo, Kaggle SVG, SVG Stack Labeled, Wikipedia's SVG collection and additional web archives). Then we subjected each file to a rigorous, fully deterministic transformation so as to yield a uniform, learnable token representation. Firstly, every visual shape tag (*e.g.*, `<circle>`, `<rect>`, `<ellipse>`, and so forth) was algorithmically converted into its exact <path> equivalent. In parallel, we collapsed all nested `<g>` groups into a single flat sequence of `<path>` entries, preserving the original draw order. We also resolved any definitions held in `<defs>`, by in-lining their computed styles directly onto each path before discarding the `<defs>` section entirely. For both gradients, linear and radial, only the final `<stop>` color was retained while discarding all the previous ones.

Next, we constrained the expressive power of the path data to just four primitive commands, `M/A/C/L`, which are sufficient to reproduce arbitrary elliptic arcs, cubic Béziers curves, and linear segments, obtaining a dramatic reduction in vocabulary size. All affine transformations (*e.g.*, translations, rotations, scalings, and skews) that might have been attached to individual elements or to group

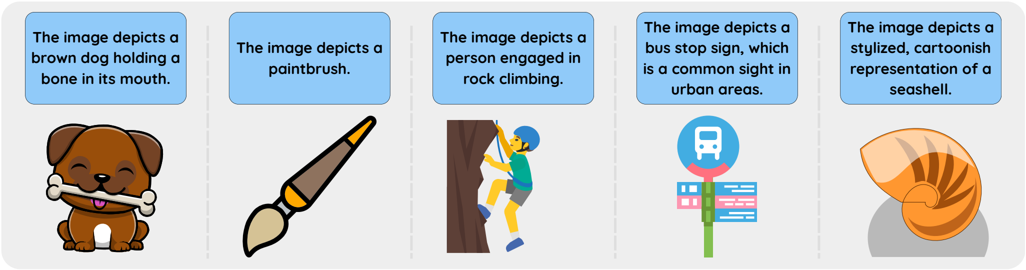

The image depicts a brown dog holding a bone in its mouth.
The image depicts a paintbrush.
The image depicts a person engaged in rock climbing.
The image depicts a bus stop sign, which is a common sight in urban areas.
The image depicts a stylized, cartoonish representation of a seashell.

Figure 2: Qualitative examples from HeisenVec. Each column shows a different SVG image, with the corresponding Idefics3 short caption on the top. The images illustrate the diversity of styles and structures present in the dataset.

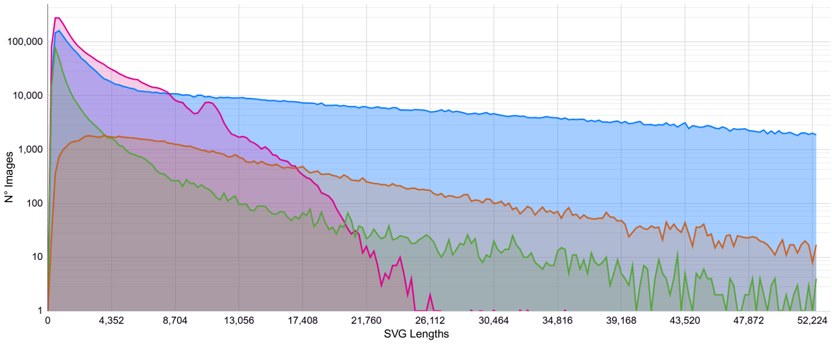


Figure 3: Distribution of SVG sequence lengths across datasets. The x-axis denotes the character length of SVG files, truncated at 52k characters to correspond to the maximum context of 32k tokens under our tokenizer. The y-axis (logarithmic scale) indicates the number of samples per length bin. HeisenVec (blue) exhibits the widest and most balanced coverage across length ranges while other datasets (ColorSVG in orange, SVGX-Core in green, and StarVector in pink) are concentrated in shorter sequences.

containers were algebraically fused to the coordinate values themselves, eliminating the need for any transform logic during decoding. We then uniformly rescaled each graphic so that its original viewbox fills a canonical 512 × 512, preserving the aspect ratio. To further simplify the downstream task, we rounded every path coordinate to the nearest integer (to associate a single token to each number) and pruned away redundant commands, such as consecutive zero-length line segments, so as to minimize token entropy.

The two color attributes (fill and stroke) were converted from any hexadecimal or named references into three base-ten numbers, representing RGB triplets, thereby removing any representational ambiguity. Finally, we performed a quality check in DINOv2 embedding space: each processed SVG was rasterized and its embedding was compared against the original. Items whose cosine similarity fell below 0.9 were removed to guard against pathological distortions. Finally, for each image, we generated three natural-language textual descriptions using: (i) BLIP2 (16), (ii) Florence2 (36) and (iii) Idefics3 (13). We extended HeisenVec with an additional caption obtained by extracting the first sentence of the Idefics3 output. These inputs were used as textual conditioning during models training phase. The full process of dataset standardization is represented in Figure 5. Using this pipeline, we produced a corpus of 2.2M fully self-contained images of which about 6k were set aside as a held-out test set, each of them ready for direct input to sequence-based language models. In conclusion, we filtered samples whose generated captions contained NSFW words or toxicity score computed by (8) greater than 0.1, which is a low threshold that means very sensitive. In Figure 2 we show some samples taken from HeisenVec test set.

3.3 Comparisons with Other Datasets

As shown in Figure 3 and Table 1, our dataset exhibits markedly broader and more uniform coverage of token lengths against competing collections. In the figure, it is evident that StarVector, SVGX-Core, and ColorSVG concentrate predominantly at shorter sequences and decline sharply as length

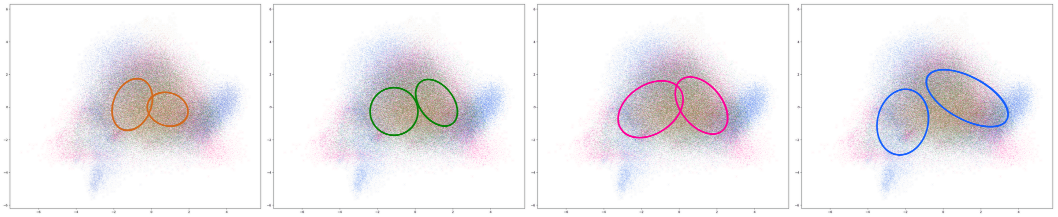


Figure 4: Two-dimensional projection of SigLIP2 embeddings for 25k samples per dataset. Each dataset is represented as a point cloud with overlaid Gaussian ellipses (from a Mixture of Gaussians with $n = 2$) fitted in the embedding space. HeisenVec (blue) exhibits larger ellipses and greater inter-cluster separation, indicating higher semantic diversity and broader coverage. ColorSVG (orange), SVGX-Core (green), and StarVector (pink) show more compact and overlapping distributions.

increases, whereas HeisenVec maintains high density across the entire range. Thereby better capturing the structural richness and complexity found in real-world vector graphics. Table 1 reports mean CLIP-Score, average sequence length, and total sample count; here, CLIP-Score is defined as the cosine similarity in CLIP (22) embedding space between each processed SVG, rasterized against a white background, and its BLIP2-generated caption. HeisenVec reaches the highest mean CLIP-Score, indicating superior alignment between image and text, and also the greatest average sequence length, underscoring its capacity for long-context SVG generation. Overall, these results establish HeisenVec as the most comprehensive and scalable benchmark for text-to-SVG research.

To further quantify the semantic diversity of each dataset from a quantitative point of view, we introduced the SpreadScore, a metric computed from a Mixture of Gaussians (MoG) fitted on SigLIP2 (30) image embeddings as per Figure 4. Specifically, it measures the weighted sum of pairwise distances between Gaussian components. SpreadScore rewards datasets that are both semantically rich and well-distributed across the embedding space.

This yields a single aggregated score, which is formally computed as

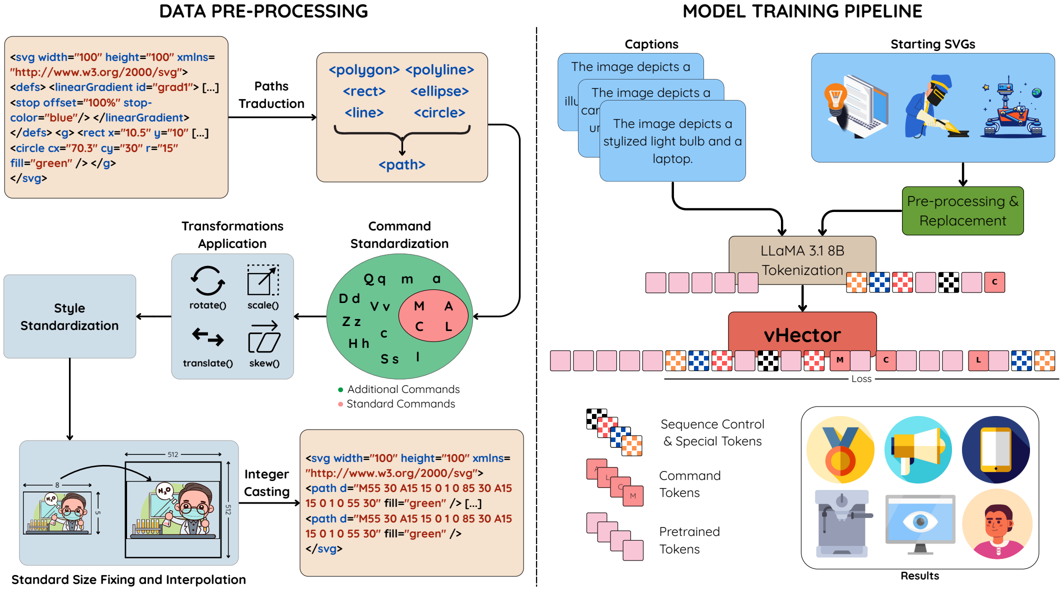
$$\text{SpreadScore} = \left(\sum_{i=1}^{n} A_i \right) \cdot \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n} \|\mu_i - \mu_j\|$$

where A_i is the area of the ellipse from the covariance matrix, μ_i is the mean (centroid) of the *i*-th Gaussian, n is the number of gaussian components and $\|\mu_i - \mu_j\|$ is the euclidean distance between centroids i and j.

Moreover, SpreadScore further affirms HeisenVec superiority: it demonstrates that our dataset not only covers a wider semantic spectrum but does so with a balanced and uniform distribution of concepts. Taken together, the length distribution analysis, elevated CLIP-Scores, and high SpreadScore paint a consistent picture: HeisenVec faithfully mirrors the complexity, diversity, and nuance inherent in real-world SVGs. This makes it an indispensable resource for developing and evaluating text-to-SVG systems capable of generating long-context and semantically coherent vector graphics at scale.

4 Baselines

In this section, we present the vHector model family, a suite of models developed to tackle SVG generation from text using a purely autoregressive, sequence-to-sequence approach. All models trained and evaluated on our proposed HeisenVec dataset, are based on two foundational LLM backbones: LLaMA 3.1 8B Instruct and LLaMA 3.2 3B Instruct. To bridge the gap between natural language modeling and structured graphic generation, we introduce a set of 17 custom tokens on top of the original tokenizer, which comprises more than 128k text-only aligned tokens. These tokens explicitly capture SVG-specific constructs, such as drawing commands, visual attributes, and sequence control delimiters, thereby extending the model's vocabulary to natively support vector image syntax. We categorize our models into two subgroups depending on the length of the SVG sequences they target: (i) short-context models, optimized for SVGs up to 8,192 tokens, and (ii) long-context models, capable of generating images up to 32k tokens in length.



```
<svg width="100" height="100" xmlns=
"http://www.w3.org/2000/svg">
<defs> <linearGradient id="grad1"> [...]
<stop offset="100%" stop-
color="blue"/> </linearGradient>
</defs> <g> <rect x="10.5" y="10" [...]
<circle cx="70.3" cy="30" r="15"
fill="green" /> </g>
</svg>
```

```
<svg width="100" height="100" xmlns=
"http://www.w3.org/2000/svg">
<path d="M55 30 A15 15 0 1 0 85 30 A15
15 0 1 0 55 30" fill="green" /> [...]
<path d="M55 30 A15 15 0 1 0 85 30 A15
15 0 1 0 55 30" fill="green" />
</svg>
```

Figure 5: **Overview of our text-to-SVG framework. (left) HeisenVec Standardization Pipeline.** Input SVGs are first traduced into a sequence of primitive path commands, retaining only the M/A/C/L instructions and eventually applying its affine transformation(s). All style attributes (including definitions in <defs>) are parsed and applied inline. We then interpolate points to normalize coordinates into a 512×512 viewbox and cast all point values to integers. **(right) vHector Training Pipeline.** We fine-tuned a large language model on tokenized pairs of captions and standardized SVG sequences. Custom sequence-control and SVG command tokens are introduced to capture both syntax and geometric semantics.

Tokenization. To accommodate the syntactic structure and semantic specificity of SVG markup, we extend the pretrained tokenizer with a compact set of domain-specific tokens tailored for vector graphic generation. These additions serve two complementary purposes: (i) they reduce the effective sequence length by collapsing frequent command patterns, and (ii) they disambiguate overloaded terms that would otherwise conflict with natural language usage. For example, tokens such as `<|M|>` or `<|L|>` explicitly encode SVG path commands, while tokens like `<|stroke|>` and `<|opacity|>` clarify attribute labels that may appear in both code and natural descriptions. The extended vocabulary also includes structural delimiters (e.g., `<|begin_of_style|>`, `<|end_of_path|>`) and symbolic constants (e.g., `<|none|>`, `<|currentColor|>`).

In designing this tokenization scheme, we intentionally preserved the pretrained model's original semantic grounding for natural language and numeric tokens, especially within textual descriptions and coordinate values. SVG control points, originally defined as (p_x, p_y) are encoded as $(p_x$`<|SEP|>`$p_y)$, leveraging the model's pre-existent knowledge on numerical values and learning separation patterns. This hybrid strategy enables the model to retain general-purpose language understanding while learning the statistical regularities of structured SVG syntax. The introduced tokens are exclusively dedicated to vector graphics generation and their associated distributions are learned from scratch during fine-tuning. In the replacement stage, we substitute the substring of the standardized SVG with the character representation of added tokens (*e.g.*, substring style "fill:" is replaced with `<|color|>`). The full list of added tokens, along with their semantic meaning and examples, is presented in the supplemental materials.

vHector 8B and vHector 3B. We introduce two fully fine-tuned variants, vHector-8B and vHector-3B, trained on dataset segments up to 8,192 tokens (using our tokenizer). During this phase, we perform end-to-end fine-tuning to tightly align the newly added token embeddings with the model's internal representations. A crucial step given the structured nature of SVGs compared to conventional text. We started training the model from the subset 0-2k tokens, then gradually extended the context window to reach 8,192 tokens, adjusting the batch size accordingly. Training leverages the short-form captions from Idefics3, ensuring that each model learns to generate SVG code conditioned on concise, globally descriptive prompts. Empirically, the 8B variant delivers superior visual fidelity, while the

Methods	Type	Token pred.	CLIP-S ↑	FID ↓	HPSv2 ↑	SSIM ↑	LPIPS ↓	DINO-S ↑
ChatGPT-4V	Prop. VLM	-	73.66	262.42	16.68	65.78	61.53	47.44
Claude3	Prop. VLM	-	76.38	219.15	17.33	67.27	59.66	41.18
VectorFusion	Diffusion	-	77.29	431.61	18.44	49.07	78.06	39.76
SVGDreamer*	Diffusion	-	55.63	848.61	14.27	32.87	86.21	20.20
LLaMa 3.2 3B	LLM	8,192	60.06	701.75	14.17	65.32	63.63	27.33
LLaMa 3.1 8B	LLM	8,192	61.54	578.86	14.17	64.43	64.8	30.93
CodeLLaMA	LLM	8,192	60.82	793.97	13.83	59.51	67.79	25.68
IconShop	Dec. Only	512	66.09	316.86	12.94	46.78	68.29	39.23
LLM4SVG	LLM	2,048	62.08	830.70	14.01	**69.62**	61.85	35.92
OmniSVG	VLM	8,192	63.82	309.20	14.46	61.05	65.93	37.63
vHector 8B	LLM	8,192	**67.84**	**105.22**	**15.32**	65.91	**57.54**	**49.33**
vHector 3B	LLM	8,192	64.30	343.90	14.34	63.62	61.40	37.76

Table 2: Quantitative comparison of text-to-SVG captioning performance. Top entries correspond to diffusion-based methods. Our vHector models (highlighted) consistently outperform prior LLM-based approaches across CLIP-S, FID, HPSv2, SSIM, LPIPS, and DINO-S metrics. Underlined values indicate the absolute best scores across all methods, while bold values denote the best performance within the autoregressive/LLM setting.
* denotes that SVGDreamer generation pipeline was simplified due to computational time constraints.

3B model achieves comparable quality with substantially lower computational overhead. These checkpoints serve as robust baselines for subsequent adaptation stages. Training details are discussed in the supplementary materials, while Figure 5 shows, on the right, the fine-tuning pipeline.

vHector 3B-long and vHector 3B-exlong. Extending generation for longer sequences, we build upon vHector-3B by training lightweight LoRA adapters (with rank $r = 16$ and $\alpha = 32$) for progressively larger context windows, taking a cue from previous work (4; 14). Specifically, vHector-3B-long handles up to 16k tokens by fine-tuning vHector-3B on the 8k-16k split, and vHector-3B-exlong reaches 32k tokens via adapters trained on the 16k-32k split. We freezed the base model and updated only adapter parameters; this strategy scales efficiently, isolates each context interval in its own specialized adapter, and minimizes resource overhead while enabling robust long-sequence generation from compact prompts.

5 Experiments

Evaluation Metrics. We evaluate our text-to-SVG models using a combination of distributional, semantic, and perceptual measures. First, we report Fréchet Inception Distance (FID) (10), which quantifies the distance between the Inception v3 (28) embedding distributions of generated samples and those of the ground truth. Lower FID indicates closer alignment with the true data manifold. Next, we use CLIPScore (CLIP-S) (9), the cosine similarity between CLIP (22) image and text embeddings, to assess how well each generated image matches its source caption. Finally, we include the Human Preference Score (HPS v2) (35), a CLIP-based surrogate fine-tuned to approximate human judgments: given a caption-image pair, HPS v2 predicts the likelihood that a human would prefer the generated image.

In addition to these generation-oriented metrics, we also report instance-level reconstruction scores, namely Learned Perceptual Image Patch Similarity (LPIPS) (42), Structural Similarity Index Measure (SSIM) (32), and DINO Similarity (DINO-S) (19). Although these metrics were originally designed for image reconstruction and may not fully capture the diversity inherent in generative tasks, we include them to provide complementary evidence of fidelity at the pixel and feature levels.

5.1 Experimental Results

Quantitative Results. Table 2 summarizes our results on the HeisenVec text-to-SVG benchmark, considering only those large language model-based methods whose pretrained weights were publicly available at the time of writing. Our vHector-8B model achieves the best performance among autoregressive approaches across all three generation-oriented metrics: it attains the highest CLIP-S, the lowest FID, and the strongest HPS v2 scores compared to other LLM-based competitors. Moreover, vHector-8B leads on both DINO-S and LPIPS, demonstrating superior feature-level agreement and perceptual generated Scalable Vector Graphics (SVG) images demand evaluation criteria tuned to their symbolic and vectorial nature: criteria that existing metrics such as FID, LPIPS, or CLIPScore fail to satisfy quality preservation compared to alternative methods such as LLaMa 3.1 8B, CodeLLaMA, IconShop, LLM4SVG, and OmniSVG (41). While LLM4SVG achieves the best SSIM score overall among LLM-based approaches, our method ranks competitively on this metric while maintaining stronger performance across other dimensions. These results collectively highlight vHector ability to generate semantically accurate and visually faithful SVGs. Although reconstruction metrics are not ideally suited to capture the diversity inherent in generative tasks, they provide useful complementary evidence of per-sample fidelity.

When compared to diffusion-based approaches, a substantial performance gap remains, underscoring the challenge of directly generating high-quality SVG content. We view our publicly released dataset and benchmarks as a foundation stone for narrowing this gap while acknowledging that the scale and diversity of existing large-scale raster-captioned datasets (*e.g.*, LAION5B (25)) still outclass current text-to-SVG resources.

Context Length Analysis. To investigate how model capacity interacts with context length requirements, we conducted targeted experiments (results shown in Table 3) comparing vHector-3B and vHector-8B across different context windows (2k and 8k tokens).

Metrics	vHector 3B (2k)	vHector 8B (2k)	vHector 3B (8k)	vHector 8B (8k)
CLIP-S	67.9	69.6 (+1.7)	64.3	67.8 (+3.5)
DINO-S	56.4	60.5 (+4.1)	37.8	49.3 (+11.5)
AVG Gain	–	**+2.9**	–	**+7.5**

Table 3: Comparison of vHector models on different metrics and context lengths.

These findings suggest that larger models are better equipped to maintain generation quality across varying sequence lengths and handle the complexity of extended SVG structures without sacrificing performance on shorter sequences.

Table 4 reports the results of our ablation studies in which we vary the model's context window up to 32k tokens. As context length increases, the task becomes progressively more challenging: the model must capture long-range dependencies while generating accurate SVG markup from a relatively brief prompt, per our task definition in Section 3.1. This represents an inherently divergent problem where short text inputs must condition the generation of much longer, structured output sequences that are a fundamental challenge that differs from typical language modeling scenarios.

The observed performance drops for longer sequences reflect the broader difficulty of maintaining coherent generation over extended outputs: a challenge amplified by the input-to-output expansion inherent in text-to-SVG generation. To mitigate these challenges, we developed specialized preprocessing techniques that reduce the required context window while preserving essential structural information, and introduced domain-specific tokens to create more efficient representations of SVG elements. However, we acknowledge that our current work represents an initial baseline rather than a complete solution. More sophisticated mitigation strategies, such as hierarchical generation approaches or adaptive context management, remain important directions for future work. To our knowledge, this work constitutes the first systematic investigation of long-context text-to-SVG generation and model scaling dynamics in this setting, establishing both a baseline and a research foundation through the HeisenVec dataset for future advances in handling complex, extended vector graphics generation.

Qualitative Results. Figure 6 presents representative SVG outputs from vHector alongside those produced by competing methods. Our approach consistently yields semantically faithful vector artwork that adheres to the typical structure of SVG graphics. In contrast, diffusion-based models tend to drift toward raster-style rendering, introducing background noise and pixelated artifacts that fall outside the typical SVG domain. These examples underscore vHector ability to maintain vector fidelity while accurately reflecting the input prompt.

6 Limitations

Our study is constrained by available computational resources, which prevent a thorough exploration of long-context modeling. As noted in Section 3.1, generating complex illustrations with large

Methods	Type	Token pred.	CLIP-S ↑	FID ↓	HPSv2 ↑	SSIM ↑	LPIPS ↓	DINO-S ↑
vHector 8B	LLM	8,192	**67.84**	**105.22**	**15.32**	**65.91**	**57.54**	**49.33**
vHector 3B	LLM	8,192	64.30	343.90	14.34	63.62	61.40	37.76
vHector 3B long	LLM	16,384	61.43	658.49	13.62	58.93	74.17	18.67
vHector 3B exlong	LLM	32,768	58.13	1308.79	13.30	55.17	70.60	20.69

Table 4: Ablation study on how vHector models and training pipeline behave when scaling up the total amount of predicted tokens.

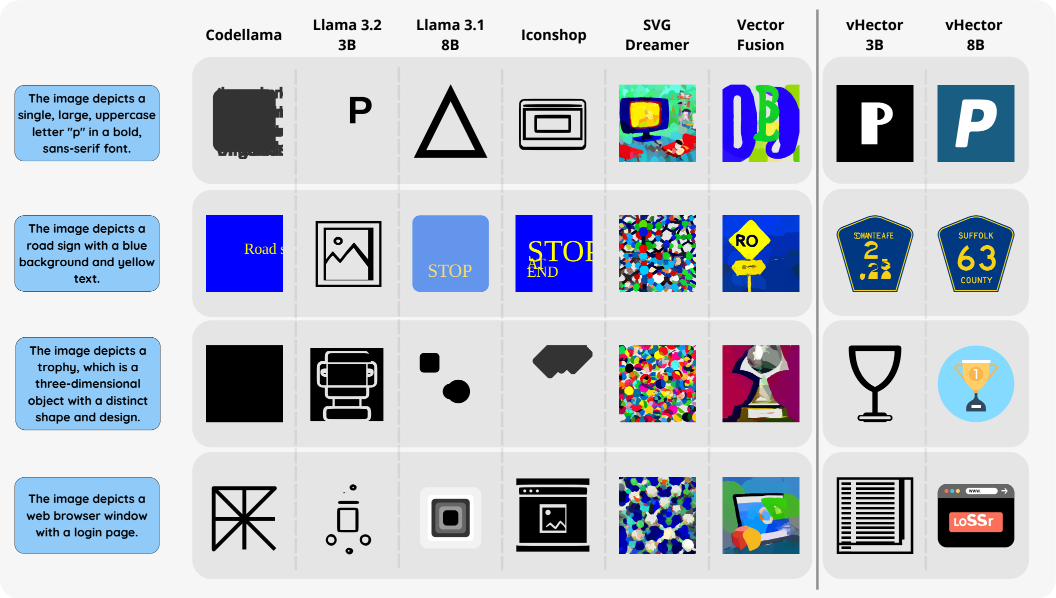

Codellama
Llama 3.2 3B
Llama 3.1 8B
Iconshop
SVG Dreamer
Vector Fusion
vHector 3B
vHector 8B
The image depicts a single, large, uppercase letter "p" in a bold, sans-serif font.
The image depicts a road sign with a blue background and yellow text.
The image depicts a trophy, which is a three-dimensional object with a distinct shape and design.
The image depicts a web browser window with a login page.
P
Road s
STOP
STOP
END
RO
SUFFOLK
63
COUNTY
LoSSr

Figure 6: Qualitative results on text-to-SVG generation on HeisenVec test set. The first column shows the input text prompts, while the followings show the corresponding generated SVG.

language models inherently depends on accurately estimates the output distribution over extended contexts starting from a brief textual prompt. Without a full investigation of this aspect, our evaluation does not capture the complete challenge of the text-to-SVG task requiring long-context integration. Furthermore, when analyzing the diversity of large-scale image-captions datasets (25) in the raster domain, text-to-SVG generation inevitably suffers from data scarcity. We believe that this work sets the first stone toward narrowing that gap, despite its uniqueness is not sufficient to fully overcome those limitations.

7 Conclusion

In this work, we introduced HeisenVec, a large-scale high-diversity dataset for text-to-SVG generation, built through a principled standardization pipeline and enriched with multi-source natural language captions. In comparison with previous proposals, HeisenVec stands out for its token-level complexity, broad stylistic coverage, and semantic diversity, offering a scalable and structurally rich benchmark for text-to-SVG generation.

To complement the dataset, we presented vHector, a family of LLM-based autoregressive models trained directly on SVG sequences, equipped with an hybrid tokenizer tailored for vector graphic constructs. Our model family supports generation up to 32k tokens, demonstrating both scalability and structural consistency across short and long SVG sequences. Experimental analysis underlines the challenges of long-context SVG generation and highlights the utility of domain-specific tokenization and curriculum fine-tuning for improving generation fidelity.

Together, HeisenVec and vHector establish a unified framework for training and evaluating SVG generation models in a pure vector domain, setting a new benchmark for future works in scalable, language-driven vector graphics synthesis.

Acknowledgments

This work has been conducted under a research grant co-funded by Doxee S.p.A. and supported by the PRIN 2022-PNRR project “MUSMA - Multimedia Understanding meets Social Media Analysis” (CUP E53D23008310001), and the PRIN 2022-PNRR project “MUCES - MUltimedia platform for Content Enrichment and Search in audiovisual archives” (CUP E53D23016290001). We further acknowledge the CINECA award, under the ISCRA initiative, for the availability of high-performance computing resources.

References

[1] Alldieck, T., Kolotouros, N., Sminchisescu, C.: Score distillation sampling with learned manifold corrective. In: Eur. Conf. Comput. Vis. (2024)

[2] Bing, Q., Zhang, C., Cai, W.: Deepicon: A hierarchical network for layer-wise icon vectorization. In: 2024 International Conference on Digital Image Computing: Techniques and Applications (DICTA) (2024)

[3] Cao, D., Wang, Z., Echevarria, J., Liu, Y.: Svgformer: Representation learning for continuous vector graphics using transformers. In: IEEE Conf. Comput. Vis. Pattern Recog. (2023)

[4] Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., Jia, J.: Longlora: Efficient fine-tuning of long-context large language models. In: Int. Conf. Learn. Represent. (2024)

[5] Chen, Z., Pan, R.: SVGBuilder: Component-Based Colored SVG Generation with Text-Guided Autoregressive Transformers. In: AAAI (2025)

[6] Frans, K., Soros, L., Witkowski, O.: Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. In: IEEE Conf. Comput. Vis. Pattern Recog. (2022)

[7] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al.: The llama 3 herd of models. In: arXiv preprint (2024)

[8] Hanu, L., Unitary team: Detoxify. Github. https://github.com/unitaryai/detoxify (2020)

[9] Hessel, J., Holtzman, A., Forbes, M., Bras, R.L., Choi, Y.: CLIPScore: a reference-free evaluation metric for image captioning. In: Conf. on Empir. Meth. in Nat. Lang. Proc. (2021)

[10] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: IEEE Conf. Comput. Vis. Pattern Recog. (2017)

[11] Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. In: Int. Conf. Learn. Represent. (2022)

[12] Hu, T., Yi, R., Qian, B., Zhang, J., Rosin, P.L., Lai, Y.K.: Supersvg: Superpixel-based scalable vector graphics synthesis. In: IEEE Conf. Comput. Vis. Pattern Recog. (2024)

[13] Hugo Laurençon and Andrés Marafioti and Victor Sanh and Léo Tronchon: Building and better understanding vision-language models: insights and future directions. In: arXiv preprint (2024)

[14] Jacobs, S.A., Tanaka, M., Zhang, C., Zhang, M., Song, S.L., Rajbhandari, S., He, Y.: Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. In: arXiv preprint (2023)

[15] Jain, A., Xie, A., Abbeel, P.: Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In: IEEE Conf. Comput. Vis. Pattern Recog. (2023)

[16] Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In: Int. Conf. Mach. Learn. (2023)

[17] Li, T.M., Lukáč, M., Gharbi, M., Ragan-Kelley, J.: Differentiable vector graphics rasterization for editing and learning. TOG (2020)

[18] Ma, X., Zhou, Y., Xu, X., Sun, B., Filev, V., Orlov, N., Fu, Y., Shi, H.: Towards layer-wise image vectorization. In: IEEE Conf. Comput. Vis. Pattern Recog. (2022)

[19] Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. Trans. Mach. Learn Res. (2024)

[20] Peter Selinger: Potrace (2019), `https://potrace.sourceforge.net/`

[21] Pun, S., Tsang, C.: VTracer - Vision Cortex (2024), `https://www.visioncortex.org/vtracer/`

[22] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: Int. Conf. Mach. Learn. (2021)

[23] Reddy, P., Gharbi, M., Lukac, M., Mitra, N.J.: Im2vec: Synthesizing vector graphics without vector supervision. In: IEEE Conf. Comput. Vis. Pattern Recog. (2021)

[24] Rodriguez, J.A., Puri, A., Agarwal, S., Laradji, I.H., Rodriguez, P., Rajeswar, S., Vazquez, D., Pal, C., Pedersoli, M.: StarVector: Generating Scalable Vector Graphics Code from Images and Text. In: IEEE Conf. Comput. Vis. Pattern Recog. (2024)

[25] Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al.: Laion-5b: An open large-scale dataset for training next generation image-text models. In: IEEE Conf. Comput. Vis. Pattern Recog. (2022)

[26] Song, Y., Chen, D., Shou, M.Z.: Layertracer: Cognitive-aligned layered svg synthesis via diffusion transformer. In: arXiv preprint (2025)

[27] Sun, P., Jiang, Y., Chen, S., Zhang, S., Peng, B., Luo, P., Yuan, Z.: Autoregressive model beats diffusion: Llama for scalable image generation. In: arXiv preprint (2024)

[28] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: IEEE Conf. Comput. Vis. Pattern Recog. (2016)

[29] Tang, Z., Wu, C., Zhang, Z., Ni, M., Yin, S., Liu, Y., Yang, Z., Wang, L., Liu, Z., Li, J., et al.: Strokenuwa: Tokenizing strokes for vector graphic synthesis. In: arXiv preprint (2024)

[30] Tschannen, Michael and Gritsenko, Alexey and Wang, Xiao and Naeem, Muhammad Ferjad and Alabdulmohsin, Ibrahim and Parthasarathy, Nikhil and Evans, Talfan and Beyer, Lucas and Xia, Ye and Mustafa, Basil and others: Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. In: arXiv preprint (2025)

[31] Wang, Z., Huang, J., Sun, Z., Gong, Y., Cohen-Or, D., Lu, M.: Layered image vectorization via semantic simplification. In: IEEE Conf. Comput. Vis. Pattern Recog. (2025)

[32] Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. (2004)

[33] Wu, R., Su, W., Liao, J.: Chat2SVG: Vector Graphics Generation with Large Language Models and Image Diffusion Models. In: IEEE Conf. Comput. Vis. Pattern Recog. (2024)

[34] Wu, R., Su, W., Ma, K., Liao, J.: IconShop: Text-Guided Vector Icon Synthesis with Autoregressive Transformers. ACM Trans. Graph. (2023)

[35] Wu, X., Hao, Y., Sun, K., Chen, Y., Zhu, F., Zhao, R., Li, H.: Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. In: arXiv preprint (2023)

[36] Xiao, Bin and Wu, Haiping and Xu, Weijian and Dai, Xiyang and Hu, Houdong and Lu, Yumao and Zeng, Michael and Liu, Ce and Yuan, Lu: Florence-2: Advancing a unified representation for a variety of vision tasks. In: arXiv preprint (2023)

[37] Xing, X., Hu, J., Zhang, J., Xu, D., Yu, Q.: Svgfusion: Scalable text-to-svg generation via vector space diffusion. In: IEEE Conf. Comput. Vis. Pattern Recog. (2025)

[38] Xing, X., Hu, j., Zhang, L., Guotao, J., Xu, D., Yu, Q.: Empowering llms to understand and generate complex vector graphics. In: IEEE Conf. Comput. Vis. Pattern Recog. (2024)

[39] Xing, X., Yu, Q., Wang, C., Zhou, H., Zhang, J., Xu, D.: Svgdreamer++: Advancing editability and diversity in text-guided svg generation. IEEE Trans. Pattern Anal. Mach. Intell. (2025)

[40] Xing, X., Zhou, H., Wang, C., Zhang, J., Xu, D., Yu, Q.: Svgdreamer: Text guided svg generation with diffusion model. In: IEEE Conf. Comput. Vis. Pattern Recog. (2024)

[41] 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. arXiv preprint (2025)

[42] Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: IEEE Conf. Comput. Vis. Pattern Recog. (2018)

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our claims are (i) the introduction of the novel vHector text-to-SVG generation task, (ii) the build and release of HeisenVec dataset, and (iii) the evaluation of different training approaches on this task. Refer to Sec. 3.1 for tasks definition, to Sec. 3.2 for dataset presentation, to Sec. 4 for the introduction of the baselines and to Sec. 5 for experimental analysis.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Detailed limitations of the current dataset and the relative method used to perform the experiments on the task are discussed in the supplemental material.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The dataset proposes a dataset and benchmarks inherit the text-to-SVG task, and doesn't include any theoretical assumptions.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Each steps about dataset collection, its pre-processing and models trainings details are deeply discussed both in the main paper and in the supplemental materials.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The proposed dataset is openly available, the code base for testing the models and reproducing the results is linked in the supplemental materials.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The details provided in Sec. 3 and Sec. 4 are sufficient to understand the experimental results and the dataset contribution. Additionally, more implementation details are available in the supplemental material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The main experiments that support the claims of this paper do not require reporting error bars. Furthermore, calculating error bars has an unsustainable cost.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Computational resources used for curating the dataset and training the baselines are deeply discussed in the supplemental materials.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research is compliant with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The potential societal impacts of ours work are discussed in the supplemental material.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We automatically filtered the curated dataset, excluding images with NSFW captions. Details are presented both in the main and supplemental materials.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The original owners of the data used in the paper are properly credited. Additional details on the licenses and terms of use are mentioned in the supplemental material.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The proposed new assets are deeply discussed in the Section 3 and in the supplemental materials.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The paper deeply discusses the usage of LLMs, since one of the contributions is a family of models obtained starting from a pretrained LLMs, specifically LLaMA 3.1 8B Instruct, and LLaMA 3.2 3B Instruct. Further details are deeply discussed in Section 4 and supplemental materials.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.