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

1 Tokens list and Commands details

To fully appreciate the process and the effectiveness of our proposed tokenization strategy, it's fundamental to start by analyzing the key components. Table 2 provides the complete list of tokens introduced to extend the base tokenizer, along with their corresponding semantic roles and string mappings. Specifically, we introduce six structural delimiter tokens (<|begin_of_svg|>, <|end_of_svg|>, <|begin_of_style|>, <|end_of_style|>, <|begin_of_path|>, <|end_of_path|>), six style-related tokens (<|fill|>, <|stroke|>, <|stroke-width|>, <|opacity|>, <|none|>, <|currentColor|>), four command-related tokens (<|M|>, <|A|>, <|C|>, <|L|>), and one general-purpose token (<|SEP|>). A key observation is that all SVGs in HeisenVec share an identical header structure due to the normalization of control points onto a fixed 512×512 canvas. Furthermore, this standardization allows for both token-level compression and structural consistency. Moreover, the format of each path is regularized such that style attributes always precede the command sequence, making the SVG representation particularly well-suited for compression in an autoregressive setting.

Table 1 shows the effectiveness of vHector tokenization method, compared to StarCoder (7) (as adopted by StarVector), which is tailored for coding. This demonstrates not only an improved compression ratio but also the ability to disambiguate overloaded terms shared between natural language and SVG syntax (e.g., distinguishing between the word style and the style attribute). In contrast to other approaches such as (12), which introduces a bigger token inventory, our design intentionally limits the number of additional tokens to minimize the amount of randomly initialized parameters, without sacrificing representation fidelity or generation quality.

To ensure that our standardization pipeline preserves the semantic content of the original SVGs, we conducted a similarity-based filtering step during dataset construction. While the standardization process introduces minor approximations, usually imperceptible, it can occasionally results in destructive transformations. To mitigate this, we computed the cosine similarity between the original and standardized images in the DinoV2 (6) embedding space, discarding any samples with a similarity score below 0.9.

Figure 1 illustrates the distribution of these similarity scores, confirming the reliability of our filtering strategy. This step effectively eliminates low-quality or corrupted samples, ensuring a clean and semantically faithful dataset for training. A visual example is also shown in Figure 2.

	Char len	**StarCoder len**	**vHector len**
Average	26,359	21,734	11,472

Table 1: Average tokenized length between StarVector and vHector. Looking at the values, from left to right, it is possible to see the compression ratio of our custom tokenizer.

Concerning the casting of coordinate points to integers, the induced approximation is mathematically negligible. For example, when discretizing to a 12×12 grid, the maximum displacement from a casted

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Track on Datasets and Benchmarks.

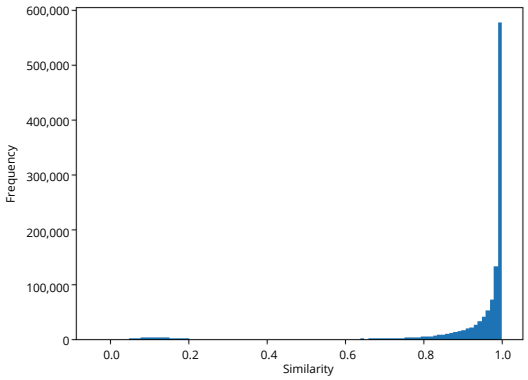


Figure 1: Distribution of similarity scores between standardized images and their corresponding raw versions. Cosine similarity is computed in the DinoV2-base (6) embedding space, comparing each standardized SVG to its original counterpart. A conservative minimum threshold of 0.9 is applied to retain only highly similar pairs.

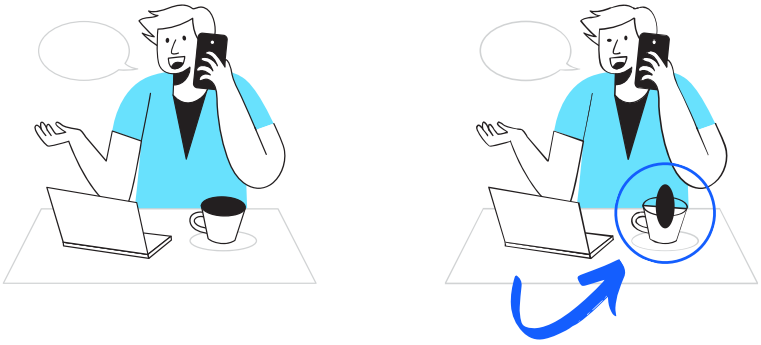


Figure 2: Raw SVG image (left) compared to the standardized one with our method (right). Even if the image reconstruction is not perfect (*e.g.*, the glass on the table), our proposed standardization algorithm keeps the semantics and the quality of the original SVG, while being able to better compress the context length during the training phase.

point to its original counterpart is 0.5 units per axis, which brings to a relative error of approximately 4.17%. However, given that our standardized canvas size is 512×512, this displacement becomes negligible, corresponding to a maximum axis error of just 0.097%. Such precision ensures that integer casting does not meaningfully distort the underlying geometry.

To facilitate semantic alignment between the textual input and the target SVG output, we deliberately structured each caption to start with the phrase "The image depicts". This consistent phrasing encourages the model to focus on the descriptive content of the prompt and improves its ability to ground the subsequent generation in the intended semantics. Additionally, we bypassed chat-based prompting schemes and instead employed a fixed instruction format, directly prompting the model with the following: "`### ImageDescription: {caption}.\n### SVG: <|begin_of_svg|><begin_of_style|>`". This explicit and minimal formatting helps the model to establish a clearer mapping between the natural language description and the structured SVG output, while also optimizing context length and tackling the distribution shift.

In conclusion, our standardization pipeline effectively normalizes the most frequent and semantically relevant attributes found in SVG images, as evidenced by the high similarity scores shown in Figure 1. We deliberately excluded infrequent or poorly supported attributes from processing to avoid introducing unnecessary complexity or noise into vHector training signal.

Token	Description	Replacement
<|begin_of_svg|>	Delimitates the start of an SVG image	'<XML ... />'
<|begin_of_style|>	Marks the begin of a style block	'<path style='
<|fill|>	Fill color attribute	'fill:'
<|stroke|>	Stroke attribute	';stroke:'
<|opacity|>	Opacity attribute	';opacity:'
<|stroke-width|>	Stroke-width attribute	';stroke-width:'
<|end_of_style|>	Indicates the end of the style block	';\t'
<|begin_of_path|>	Delimitates the begin of a draw path	'd='
<|M|>	Draw command replacement	'M'
<|A|>	"	'A'
<|C|>	"	'C'
<|L|>	"	'L'
<|SEP|>	Separator between two numbers	','
<|none|>	None representation	'none'
<|currentColor|>	Refers to "currentColor" SVG attribute	'currentColor'
<|end_of_path|>	Delimitates the end of path	'Z />'
<|end_of_svg|>	Indicates the end of SVG image	'</svg>'

Table 2: Proposed token set in vHector, along with their semantic meaning and substring replacement.

2 Computational requirements and Training details

Dataset curation. To obtain textual descriptions for the millions of collected SVG images, which originally lacked associated captions, we employed an automatic captioning pipeline leveraging three pretrained vision-language models of varying scales: Idefics3, BLIP-2, and Florence2. Specifically, we first generated detailed, long-form captions using Idefics3, a large multimodal model, requiring approximately 250 GPU hours on 8 NVIDIA A100 GPUs. Subsequently, we produced COCO-style captions using BLIP-2, which was executed in parallel across 8 GPUs with at least 16 GB of VRAM, totaling around 960 GPU hours. Finally, we employed Florence2 under a similar configuration, with the captioning process completed in approximately 240 GPU hours.

Training vHector-8B. As detailed in Section 4 of the main paper, we begin by aligning the base language model (LLaMA 3.1 8B (2)) with the text-to-SVG generation task, incorporating our custom vocabulary extensions. This initial training stage was conducted on a filtered subset of HeisenVec containing approximately 1M SVG-caption pairs, hereafter referred to as S0. Specifically, S0 includes only those samples whose tokenized length (including text) falls below 2,048 tokens. To progressively extend the model context length, we constructed two additional subsets. The first, S1, includes samples between 2,048 and 4,096 tokens, augmented with a randomly selected set of images from S0 to balance training; this results in a corpus of 431k samples. The second subset, S2, spans lengths from 4,096 to 8,192 tokens, similarly extended with S0 and S1 samples, totaling 809k examples. We then introduced LoRA adapters and trained the model on S1 for four epochs and on S2 for two epochs. Once the model successfully operated at the target 8,192-token context, we merged the LoRA weights into the base model and conducted three additional epochs of full fine-tuning. This staged approach enabled a gradual and efficient adaptation to longer sequences. This strategy derives from the notice of catastrophic forgetting and the constraint of not being able to afford full-finetuning at each stage. The full training process of vHector-8B, from initial alignment to final fine-tuning, amounted to 1,552 node hours, where each node is composed by four NVIDIA A100 GPUs, allocated as follows: 576 node hours for initial alignment (S0), 456 node hours for context extension (S1 and S2 with LoRA), and 520 node hours for final full fine-tuning.

vHector-3B and Extended Variants. The training of vHector-3B followed a similar multi-stage procedure outlined for the larger 8B variant, including initial alignment and context extension via LoRA adapters. The complete training process required a total of 616 node hours.

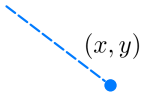
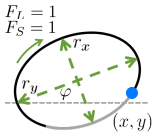
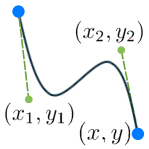
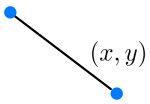
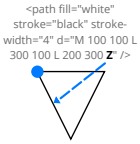
Command	Arguments	Description	Visualization
M (MoveTo)	(x, y)	Move the current point to the coordinate (x, y) without drawing anything	
A (Elliptical Arc Curve)	r_x , r_y , φ , F_L , F_S , (x, y)	Draw an arc curve from the current point to the coordinate (x, y). r_x and r_y are the two radii of the ellipse. φ is the angle representing a rotation (in degrees) of the ellipse relative to the x-axis. F_L is large-arc-flag, which selects the large arc (1) or small arc (0). F_S is sweep-flag, which chooses the clockwise turning arc (1) or counterclockwise turning arc (0)	
C (Cubic Bézier Curve)	(x_1, y_1), (x_2, y_2), (x, y)	Draw a cubic Bézier curve from the current point to the end point specified by (x, y). The start control point is specified by (x_1, y_1) and the end control point is specified by (x_2, y_2)	
L (LineTo)	(x, y)	Draw a line to the point (x, y)	
Z (ClosePath)	$\varnothing$	Close the current subpath by connecting the last point of the path with its initial point	

Table 3: <|M|>, <|A|>, <|C|>, <|L|> commands definition that have been used during the standardization phase.

To support longer context generation, we developed two extended variants of this model. vHector-3B-long was initialized from the vHector-3B checkpoint and trained on the S3 split, which consists of 351k samples with tokenized lengths between 8,192 and 16,384. This phase lasted two epochs and required 192 node hours. Building on this, vHector-3B-exlong was initialized from vHector-3B-long and trained on the S4 split, comprising 288k samples ranging from 16,384 to 32,768 tokens. This final extension phase also lasted two epochs and consumed 384 node hours.

This progressive approach enables scalable context adaptation while preserving model stability and training efficiency.

3 Additional HeisenVec analysis

3.1 Mixture of Gaussian

As shown in Figure 4 (of the main paper), HeisenVec demonstrates clear advantages in complexity, diversity, and semantic coverage measured using MoG and SpreadScore metrics. To further challenge the robustness of these findings and validate the consistency of our evaluation method, we also computed the SpreadScore using three Gaussian components per dataset. This configuration introduces a more fine-grained and demanding analysis than the standard two-component setting, requiring datasets to exhibit more nuanced semantic separability to score highly. Remarkably, even under this

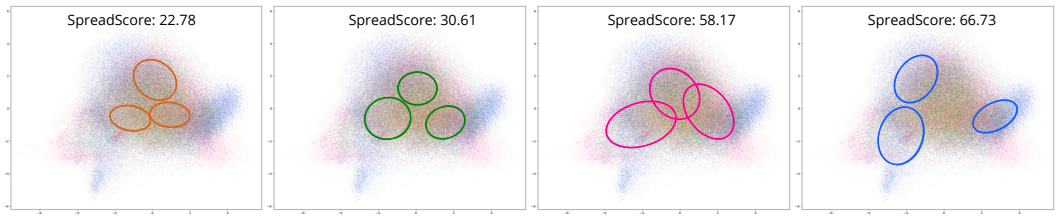


Figure 3: Two-dimensional projection of SigLIP2 embeddings for 25k samples per dataset. Each dataset is represented as a point cloud with overlaid Gaussian ellipses (from a Mixture of Gaussians with n = 3) fitted in the embedding space. ColorSVG (orange), SVGX-Core (green), StarVector (pink) and HeisenVec (blue).

stricter evaluation, HeisenVec maintains the highest SpreadScore among all datasets. As illustrated in Figure 3, the ellipses remain both larger and more distant, without overlaps, visually confirming the diversity and spread of semantic concepts. Values also quantitatively support this result, showing that our dataset retains its leading position even in a more complex and sensitive scenario. This outcome was far from guaranteed and highlights not only the inherent strength of HeisenVec but also the reliability and discriminative power of our framework. It reinforces the conclusion that HeisenVec provides a rich, well-structured foundation for research in symbolic visual generation and is well-suited to support future work requiring semantic coverage and structural diversity.

3.2 Color Distribution & Diversity

Measuring Colourfulness. To quantify perceptual colorfulness, we employed the widely used method proposed by Hasler and Süsstrunk (4). This model reflects human visual perception by computing variation and bias in two opponent color channels: red-green (RG) and yellow-blue (YB). Given an RGB image converted from SVG using CairoSVG and PIL, channels are defined as:

$$\text{RG} = |R - G| \quad \text{and} \quad \text{YB} = \left| \frac{R + G}{2} - B \right|$$

From these, we compute:

$$\sigma_{\text{RG}} = \text{std}(\text{RG}), \quad \mu_{\text{RG}} = \text{mean}(\text{RG}), \quad \sigma_{\text{YB}} = \text{std}(\text{YB}), \quad \mu_{\text{YB}} = \text{mean}(\text{YB})$$

The final colorfulness score C for each SVG is calculated as:

$$C = \sqrt{\sigma_{\text{RG}}^2 + \sigma_{\text{YB}}^2} + 0.3 \cdot \sqrt{\mu_{\text{RG}}^2 + \mu_{\text{YB}}^2} \tag{1}$$

Sparsity Estimation. To analyze the spread of colorfulness scores within each dataset, we used an inter-percentile range between the 85th and 15th percentiles:

$$\text{IQR}_{85-15} = P_{85} - P_{15} \tag{2}$$

This percentile-based range provides a robust view of internal variability, avoiding sensitivity to outliers while capturing the diversity of color usage across samples.

Results. Despite comprising 2.2M SVGs, HeisenVec demonstrates strong color diversity, as shown by its high IQR_{85-15} value of 71.22. This indicates a broad range of perceptual colorfulness across its samples, which is a key strength compared to other datasets. While ColorSVG has a slightly higher average colorfulness, HeisenVec stands out for its internal variation, making it particularly suitable for training models that benefit from stylistic and chromatic diversity.

Metric	SVGX-Core	ColorSVG	StarVector	HeisenVec
Min	0.00	32.34	0.00	0.00
Max	197.32	206.73	208.56	213.18
Average	96.10	119.38	95.44	103.52
IQR_{85-15}	60.29	70.00	70.77	**71.22**

Table 4: Values comparison of Colorfulness across datasets.

4 Dataset sources and licenses

The HeisenVec dataset was curated by aggregating SVG images from multiple publicly accessible sources. These include the Wikipedia dataset (CC BY-NC 4.0), FIGR8-SVG and svg-stack-labeled (MIT License), SVGrepo (a collection under varying permissive licenses), Open Doodles (CC0 1.0), and the works of Lukasz Adam (CC0). In addition, a subset of images was collected via web crawling, for which the precise licensing of individual assets could not be reliably determined. To ensure compliance with ethical research standards while acknowledging these limitations, the entire HeisenVec dataset is released under a custom license, available at this link.

5 Safeguards

In recognition of the importance of responsible data curation, we applied careful and deliberate safeguards to minimize the presence of inappropriate or toxic content in the dataset. Our filtering process combined both keyword-based exclusion of explicit NSFW terms and a toxicity score threshold of 0.1, computed using the Detoxify model (3) developed by Unitary AI. This conservative threshold reflects our sensitivity toward even mild toxicity to ensure a respectful and inclusive dataset. Given the large scale of 2.2M samples, we employed a dual filtering approach to maximize effectiveness; however, we acknowledge that, despite our best efforts, some residual instances may persist. In addition, a small portion of images and captions that might be considered "borderline" could have been not removed to preserve the dataset's heterogeneity and represent a broad spectrum of visual and cultural concepts, always with careful attention to respect and inclusivity. We view these measures as a thoughtful balance between thorough moderation and the need for diverse, meaningful data to support robust research. Going forward, we recommend complementing automated safeguards with human oversight and culturally aware review processes to further enhance dataset quality and ethical standards.

6 Social Impact

HeisenVec represents a significant step forward for the research community interested in structured visual generation, offering the first large-scale benchmark for text-to-SVG synthesis at scale. By providing over 2M vector images paired with rich natural language descriptions, this dataset enables rigorous study of models that map between symbolic, interpretable visual forms and language, a challenge that lies at the intersection of vision, graphics, and NLP. The standardized tokenization, long-context sequences, and stylistic diversity make HeisenVec a valuable resource for developing and benchmarking auto-regressive models, particularly those aimed at handling long, structured outputs. Beyond its direct utility for training and evaluation, HeisenVec lays the groundwork for a new class of generative systems that support editable, resolution-independent, and semantically grounded visual content creation. We anticipate that this work will catalyze research in text-conditioned illustration tools, interactive design assistants, and creative AI systems tailored to the needs of graphic designers. Furthermore, it can inspire advances in compositional generation, hierarchical modeling, and multimodal reasoning across academic and industry settings. While the dataset relies on synthetic captions, which may introduce biases or noise, we believe its scale and structure provide a strong foundation for studying these challenges explicitly. By releasing the dataset, tokenizer, and evaluation suite, we aim to lower the barrier to entry and encourage a diverse range of contributions from the community in structured visual generation and design-oriented AI.

7 Diffusion Models

7.1 SVGDreamer Vanilla Run

Regarding the footnote * in Table 2 (of the main paper), we adapted the original SVGDreamer pipeline to balance computational cost, energy efficiency, and response time. Starting from the publicly available *iconography.yaml* configuration on the official GitHub repository, we reduced *VPSD num_iter* from 2000 to 250 and set *VPSD n_particle* to 1. These adjustments significantly lowered the generation time from approximately 7 hours to just 15 minutes per image, making the method more practical for benchmarking in realistic use cases, such as interactive design workflows.

Importantly, we did not compromise on fairness or qualitative rigor. To validate that these simplifications still preserved a reasonable level of visual fidelity, we carefully assessed the outputs and found that the differences between the full and reduced settings were often irrelevant in practice. For transparency and respect toward both the readers and the original authors of SVGDreamer, we include a small-scale comparison using the unmodified "vanilla" configuration, shown in Figure 4. This evaluation confirms that our choice reflects a thoughtful trade-off, grounded in practical constraints, while maintaining the spirit and capabilities of the original method.

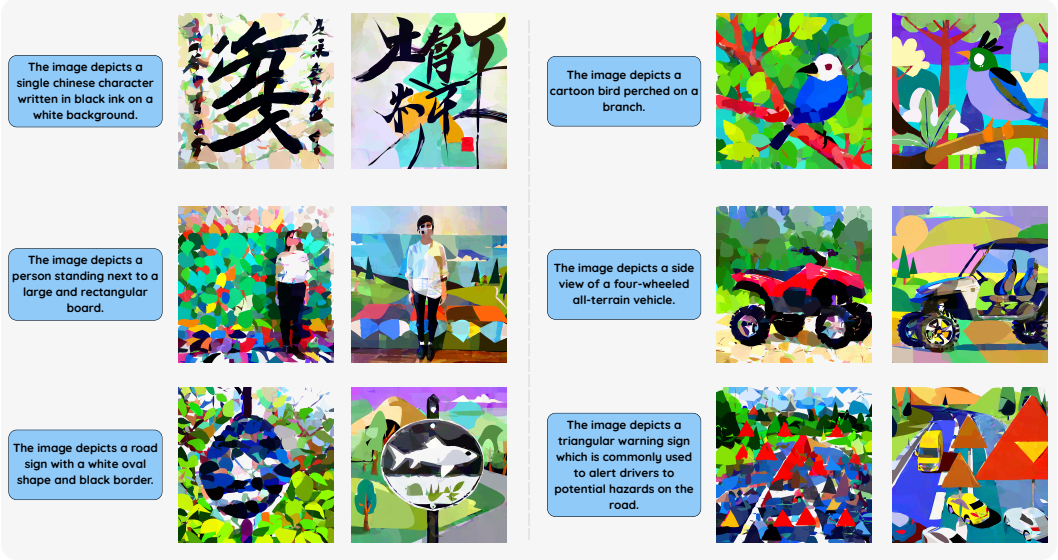


Figure 4: Visual comparison between 2 different generations by SVGDreamer starting from the same prompt. For each example: left image (15 minutes) and right image (7 hours).

7.2 SVG Generation Time

A critical factor in evaluating SVG generation models is their response time to a given prompt, especially in real-world scenarios such as interactive design workflows. As outlined in our introduction, for graphic designers, a model that takes several minutes or even hours to return a result is impractical, regardless of output quality. We argue that latency between one and two minutes represents a reasonable tradeoff between responsiveness, quality, and personalization. Standard LLMs typically respond quickly, but their SVG outputs tend to be short and structurally simplistic. In contrast, diffusion-based models can generate more complex visuals but often suffer from slower inference, making them less suitable for interactive applications.

Our method vHector occupies a compelling middle ground. It is a pure language model, meaning it benefits from the inference efficiency and scalability of transformer-based LLMs. At the same time, thanks to its ability to model long context sequences and handle symbolic outputs in a structured manner, it achieves a level of visual and semantic complexity that approaches diffusion models. We believe this architecture strikes the right balance for the graphics domain, delivering rich, high-fidelity

	Foundation LLMs	vHector	VectorFusion	SVGDreamer
Gen. Time	$\approx 1min$	$\approx 3mins$	$\approx 15mins$	$\approx 7hours$

Table 5: text-to-SVG task generation time comparison across different types of architectures.

Datasets	CLIP-S ↑	FID ↓	HPSv2 ↑	SSIM ↑	LPIPS ↓	DINO-S ↑
ColorSVG	66.82	252.83	14.74	**50.74**	67.48	39.72
SVGX-Core	**70.20**	258.67	16.49	35.46	**63.34**	**47.06**
StarVector	68.30	**199.08**	**17.90**	29.10	66.66	44.99

Table 6: Performance of vHector-8B on other existing text-to-SVG test set.

vector content while preserving the responsiveness essential for near real-time design tools. Moreover, it opens up a promising direction for future work in fast, structure-aware generative systems tailored for creative professionals.

Table 5 shows a numerical time usage comparison, per single image, between all the models we used.

8 vHector-8B on others test set

Table 6 assesses the generalization capabilities of vHector-8B by evaluating its performance on the test splits of the other three existing text-to-SVG datasets: ColorSVG (1), SVGX-Core (12), and StarVector (7). Results highlight the competitive robustness of our model across multiple evaluation metrics. Notably, vHector-8B achieves the highest CLIP-Score (5) on SVGX-Core and DINO-Score (6) on the same set, indicating strong semantic alignment and visual features coherence. On StarVector, the model attains the best FID (9) and HPSv2 (11), underscoring its ability to preserve stylistic fidelity and human preference alignment. SSIM (10) peaks on ColorSVG, reflecting structural consistency, while LPIPS (13) remains low across all benchmarks. These results demonstrate that vHector-8B maintains high-quality generation even outside the domain it was trained on.

9 Ablation studies

In Table 7 we show the effectiveness of introducing the <|SEP|> token during the initial alignment stage (S0), where new tokens are integrated into the model vocabulary. Across all metrics, the variant using <|SEP|> consistently outperforms the baseline without it, achieving superior CLIP-Score, FID, SSIM, LPIPS, and DINO-Score. Notably, we placed particular emphasis on DINO-Score at this stage, as it offers a more reliable assessment of perceptual alignment with the ground-truth SVG domain compared to purely pixel-based metrics. The consistent improvement in DINO-Score suggests that the inclusion of <|SEP|> significantly helps the model in maintaining domain fidelity and visual coherence from the earliest phase of fine-tuning.

To evaluate the HeisenVec's benefits, we trained our vHector 3B on the alternative datasets, using our standard preprocessing approach. Training was limited to the first alignment stage (6 epochs of fine-tuning from the pretrained LLaMa 3.2 3B) due to computational constraints. As shown in Table 8, models trained on HeisenVec consistently achieve higher scores across different met-

Tokenization	CLIP-S ↑	FID ↓	HPSv2 ↑	SSIM ↑	LPIPS ↓	DINO-S ↑
without <|SEP|>	69.34	122.60	15.52	65.39	56.66	51.14
with <|SEP|>	**69.85**	**95.37**	**15.53**	**65.76**	**55.98**	**53.24**

Table 7: Ablation study about the usage of <|SEP|> token computed at the end of the first alignment (S0). It is evident that using <|SEP|> token helps the model to better learn the images distribution.

Models	CLIP-S ↑	HPSv2 ↑	DINO-S ↑
FT_SVGX-Core	62.98	13.98	35.36
FT_ColorSVG	58.86	12.83	31.46
vHector 3B	**67.83**	**14.84**	**48.52**

Table 8: Performance comparison of vHector 3B trained on alternative datasets (FT means Fine Tuning).

Preprocessing	CLIP-S ↑	HPSv2 ↑	DINO-S ↑
without std	64.37	13.91	38.45
with std	**67.83**	**14.84**	**48.52**

Table 9: Impact of the SVG standardization pipeline on vHector 3B performance.

rics, demonstrating that the scale and diversity of our dataset effectively support improved model generalization.

To assess the contribution of our standardization pipeline on model performance, we trained a version of our model excluding both path command reduction and coordinate quantization from the preprocessing pipeline. Table 9 shows that the proposed standardization steps provide substantial performance benefits.

The performance improvements from standardization operate through two complementary mechanisms. First, removing these preprocessing steps leads to significantly longer token sequences for representing the same visual content. As demonstrated, this increased sequence length directly degrades model performance. Second, our standardization approach reduces the entropy of the training dataset by constraining the set of possible SVG commands while preserving all essential visual information from the user's perspective. This produces a more uniform and predictable distribution that's easier for the model to learn: the model can focus on learning meaningful visual patterns rather than navigating redundant representational variations that do not contribute to the final rendered output.

Our standardization choices serve as crucial components for enabling effective SVG generation through both computational efficiency and improved learning dynamics.

9.1 User Study

Table 10 reports the additional results from our user study, showing the average ratings provided by humans. The user study assessed both overall image quality and text-image semantic alignment across six different SVG generators methods. As shown in the table, vHector 8B achieves the highest scores in image quality, indicating superior visual fidelity and rendering coherence compared to other approaches. In terms of text alignment, vHector 8B ranks second, with VectorFusion (a

Models	Image-Text Correlation Score	Image Quality Score
CodeLLaMa	1.41	1.52
IconShop	2.27	2.30
SVGDreamer*	1.35	2.38
OmniSVG	2.73	2.70
VectorFusion	**3.99**	3.00
vHector 8B	3.88	**3.80**

Table 10: Human evaluation results from the user study comparing models on Image-Text Correlation and Image Quality. * denotes that the SVGDreamer generation pipeline was simplified due to computational time constraints.

diffusion-based method) performing slightly better. However, this advantage comes at the cost of substantially higher computational overhead.

10 Limitations

Alignment Challenges from Sequence Imbalance. In text-to-SVG generation, the output sequences are significantly longer than the input prompts, often by more than an order of magnitude. This substantial length disparity poses a challenge for the model in establishing fine-grained correspondences between elements of the SVG and the input text. Ideally, each generated path should reflects a semantically coherent element derived, explicitly or implicitly, from the prompt, and the tokens composing the path should maintains internal consistency. While analogous alignment challenges exist in traditional language modeling, where one might compare paths with sentences, the unbalance between input and output lengths is typically less severe. In this setting, the model must attends over much longer output sequences which increase the difficulty of preserving local and global semantic fidelity. As a result, certain textual details may be diluted or omitted, and portions of the generated SVG may lack a clear or coherent connection to the original prompt.

Dataset Size. Although our proposed dataset comprises 2.2M samples with a focus on long-context generation, a fundamental limitation of text-to-SVG task emerges when compared to the text-to-image domain in terms of data availability. State-of-the-art text-to-image models benefit from massive-scale datasets such as LAION-5B (8), whereas, in comparison, text-to-SVG suffers from a scarcity of training data. Bridging the gap in generalization performance will likely require access to datasets several orders of magnitude larger. However, collecting such data poses a substantial challenge: unlike raster images, high-quality SVG content, particularly with associated captions, is rarely available at scale, even in unlabeled form, making large-scale dataset construction for text-to-SVG task inherently more difficult.

11 Progressive Generation

vHector creates SVG images autoregressively generating pieces of SVG paths. Each time vHector concludes to create a complete path, it will correspond to a simple shape, which is then added to the canvas. Over time, these individual paths are aggregated to form the complete image.

This strategy reflects the inherent structure of SVGs, which are composed of a series of vector paths layered in sequence. By progressively adding new paths, the model builds up the visual content of the image, starting from the overall composition and gradually refining it with more specific and intricate details.

Figure 5 illustrates how the image evolves step by step. The early paths typically define the global layout and main visual elements, while subsequent ones introduce finer features and contextual elements, ultimately leading to a coherent and visually rich SVG representation, enlightening the need for long context modeling for accurate text-to-SVG.

12 Additional Dataset Samples

12.1 Image Captioning Examples

To provide a clearer visual overview of our dataset composition, in Figure 6 we present additional SVG examples from training and test sets, along with their associated captions. The image highlights that each SVG is accompanied by not just one but three distinct captions. Recalling that the short caption was obtained by extracting the first sentence from the long version originally generated by Idefics3. We opted for this choice to offer a concise summary of the main visual characteristics of the image, as the full caption produced by Idefics3 tends to be verbose. Including multiple captions enhances the versatility of the dataset: depending on the intended application, one can choose to use all, a single, or a subset of the available captions.

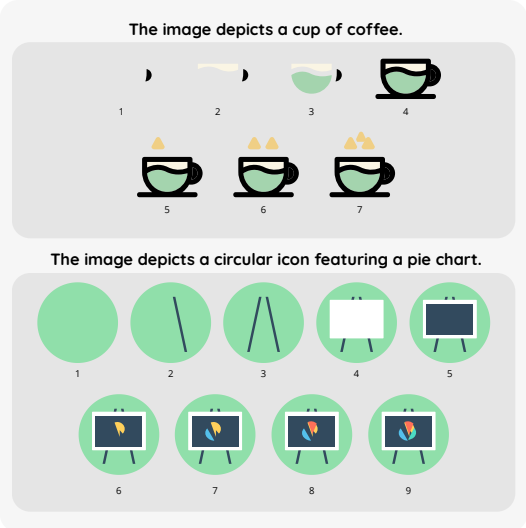


Figure 5: Progressive generation of an SVG image by vHector-8B.

12.2 Categorize SVG Content by Length

Our dataset includes SVG contents of varying lengths, which correspond to different levels of image complexity. In Figure 7, we organize them into seven categories to provide a comprehensive overview of the visual diversity represented in the dataset. It is evident that, as the length of the SVG increases, the level of detail within the corresponding image also increases, allowing it to align with the context introduced by the caption more accurately.

13 Additional Generated Images

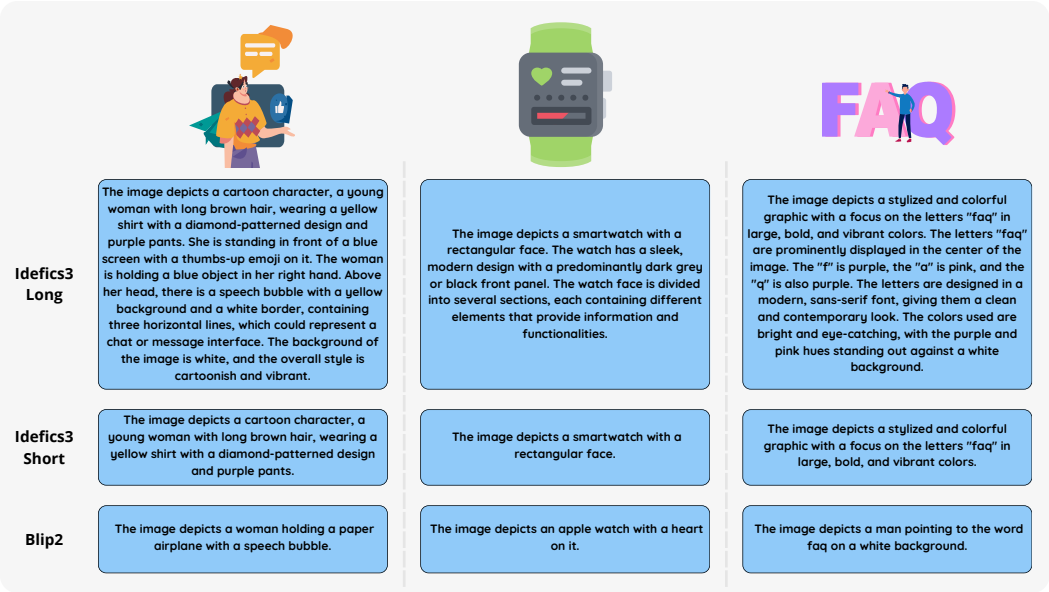
We inferred vHector models using short captions from our test set, as input. Figure 8 presents a sample of SVGs generated by vHector-8B, providing a visual representation of the quality it can achieve. The results demonstrate both high image quality and effective context preservation.

14 Additional Qualitative Comparisons

We provided the short captions from our test set to competitor models, asking them to generate SVG images, and then performed a qualitative comparison. We included many models in the evaluation, also considering baselines that tackle text-to-SVG task using diffusion model (e.g., SVGDreamer and VectorFusion) but can still be used for the same final purpose. Figures 9 and 10 demonstrate that both the 3B and 8B versions of our model produce high-quality generations, especially compared to LLM-based competitors. However, the 3B variant sometimes fails to generate clean SVGs: as shown in Figure 11, it can produce noisy or malformed outputs.

15 Failure cases

We collected examples where even vHector 8B failed to generate correct SVGs, some of which are shown in Figure 12. While these outputs are incorrect, they often partially reflect the reference caption, capturing aspects like shape type or color despite errors in composition or layout. Reporting such failures highlights the model's limitations and the types of prompts that remain challenging. A systematic error analysis on the full HeisenVec test set (5,837 captions) showed that 5,552 generations (95.06%) were syntactically correct and visually meaningful; among these, 51 (0.87%) were correct but empty, 38 had invisible paths, 12 had null styles, only one had stroke width exceeding the canvas, lastly 237 (4.07%) were syntactically incorrect.



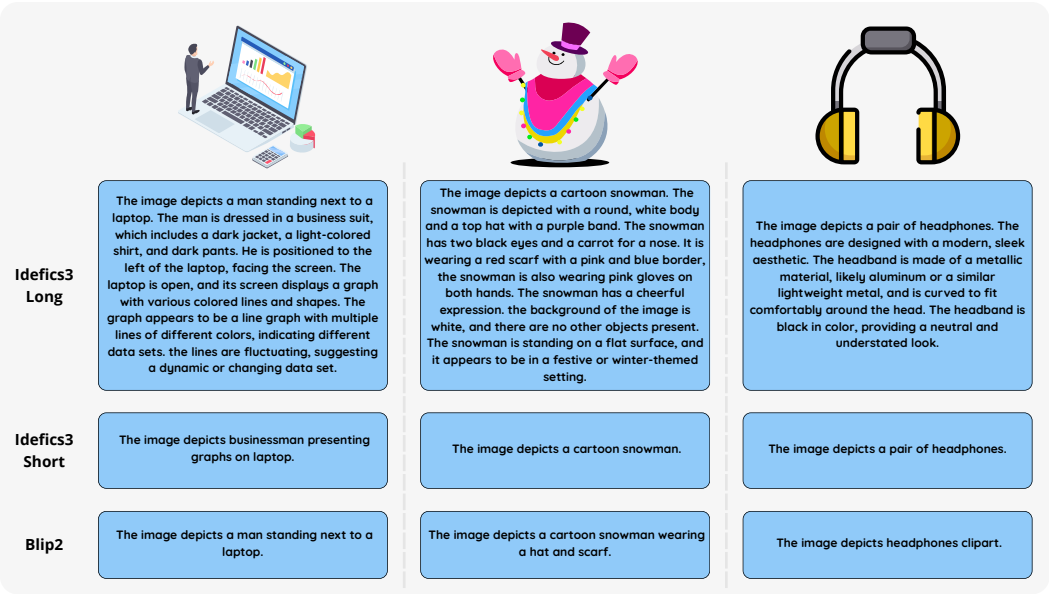


Figure 6: SVG examples from our training and test set, each one followed by the corresponding captions.

Figure 7: Examples combining short caption and the corresponding SVG image from our dataset, ordered by SVG data length. Greater content length introduces more complexity, resulting in more detailed figures.

Figure 8: SVG images generated with vHector-8B followed by their corresponding short captions.

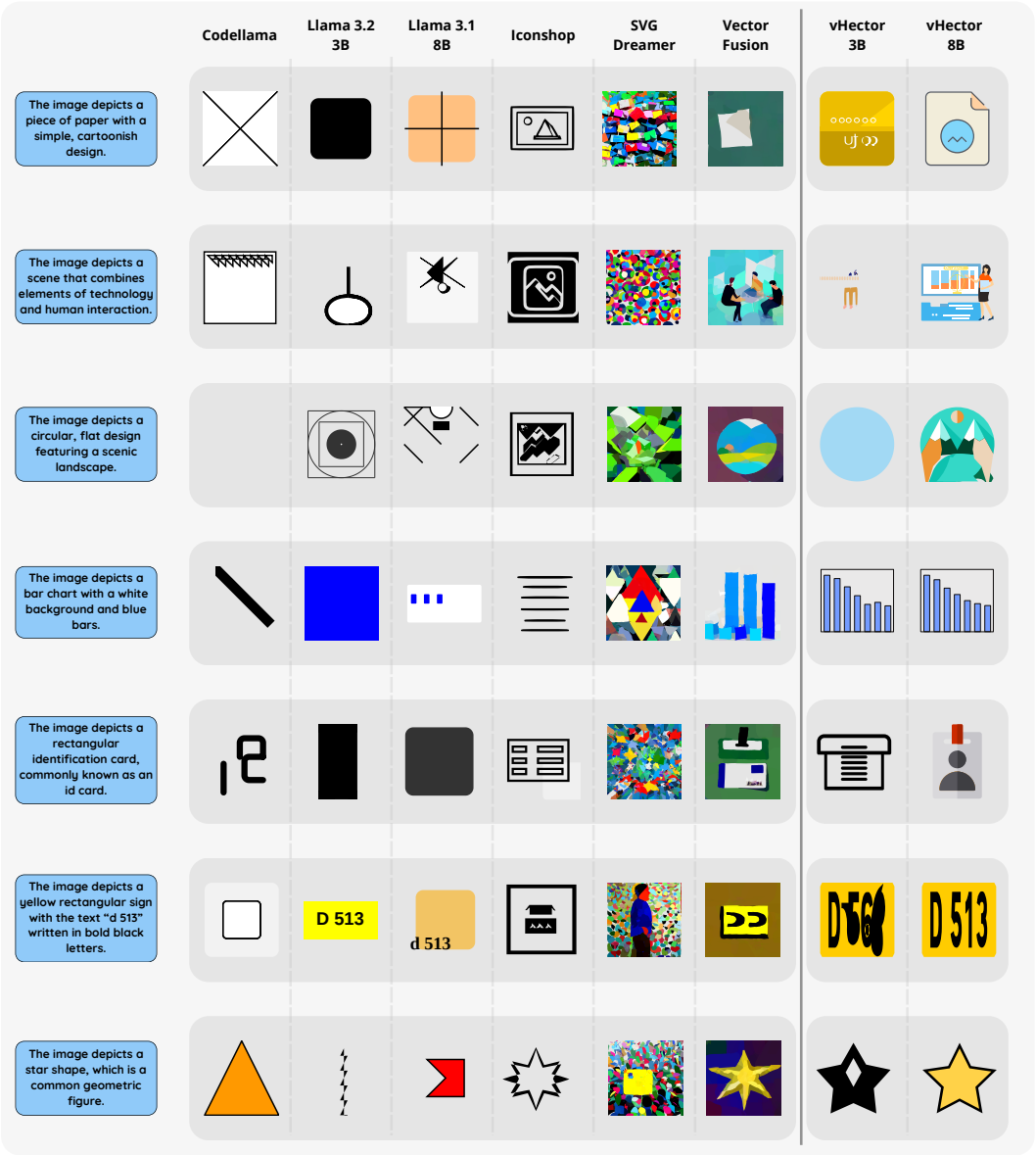


Figure 9: Short captions followed by generations from different competitor models (center columns), and both vHector-3B / vHector-8B models (last two columns).

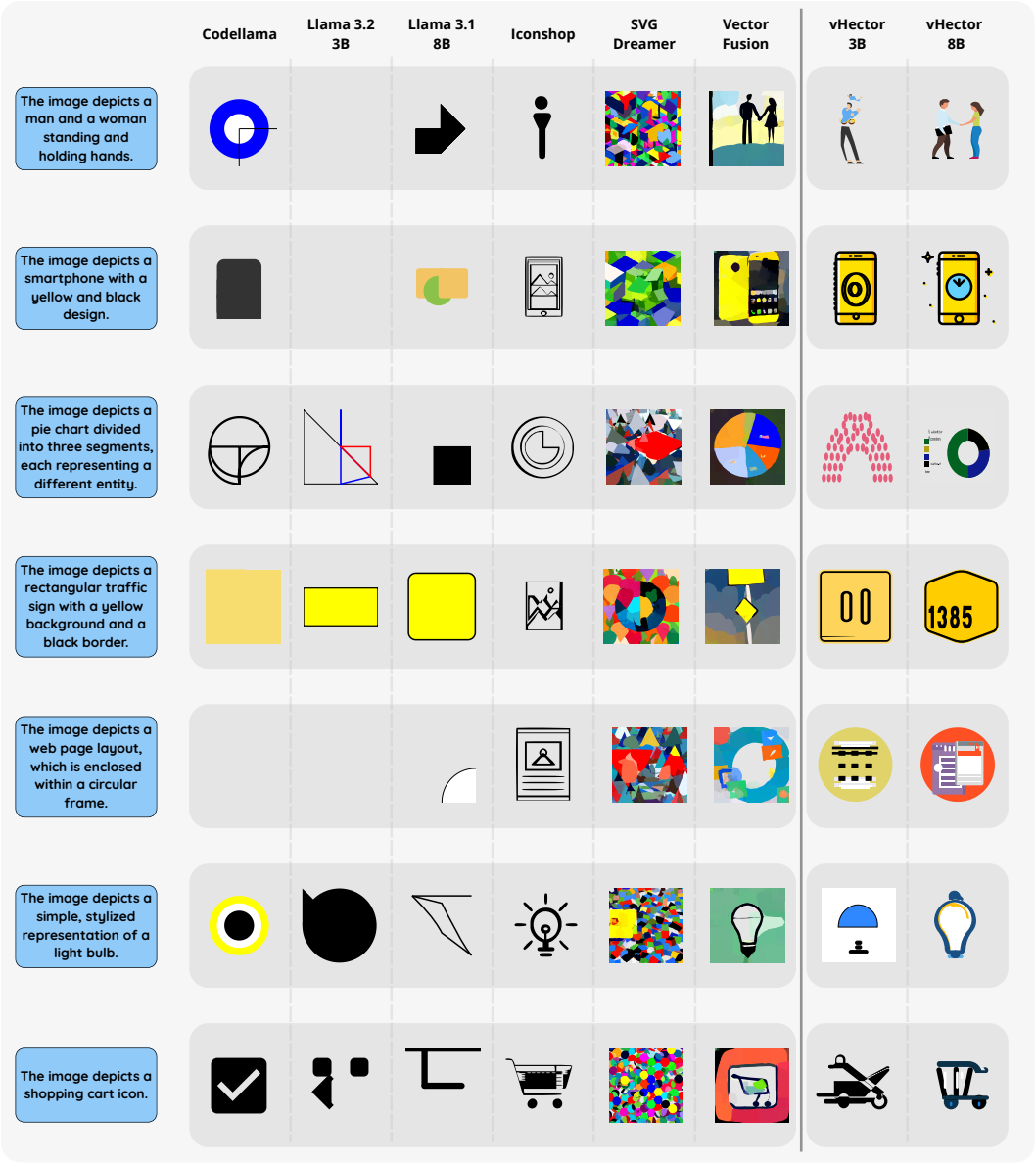


Figure 10: Short captions followed by generations from different competitor models (center columns), and both vHector-3B / vHector-8B models (last two columns).

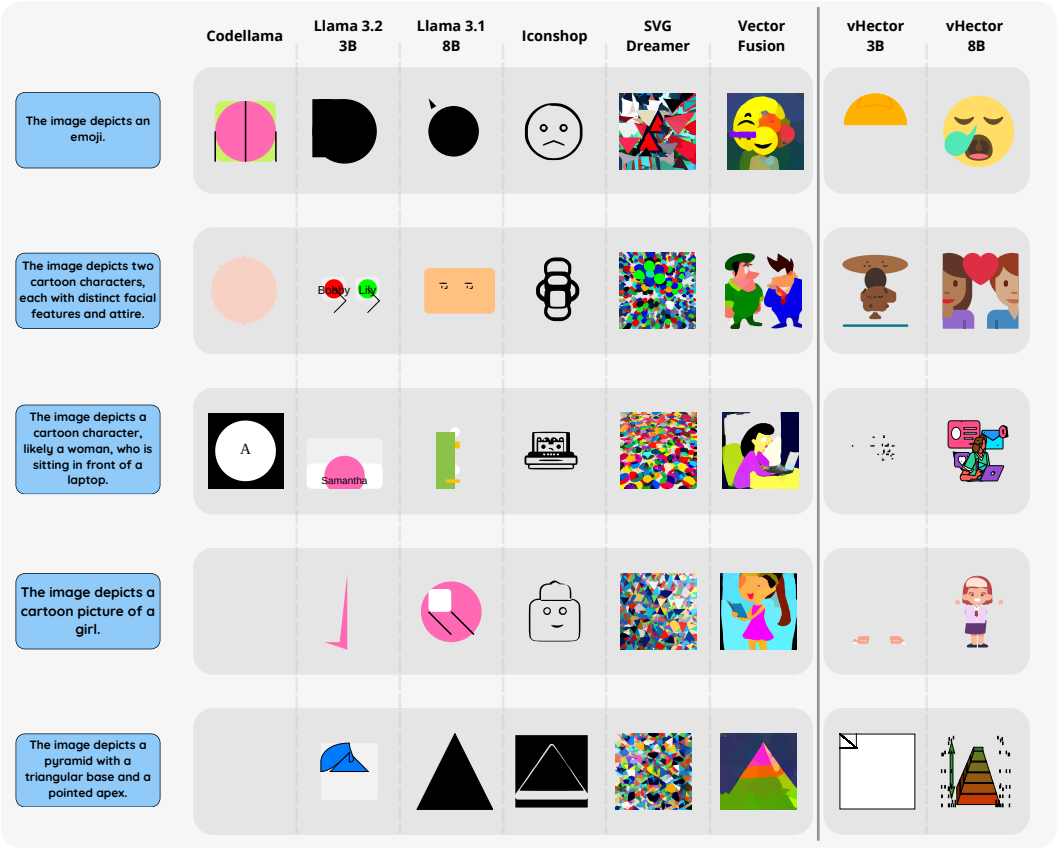


Figure 11: Short captions followed by generations from different competitor models (center columns), and both vHector-3B / vHector-8B models (last two columns). In these examples, the 3B version of our model struggled to generate well-structured SVG images.

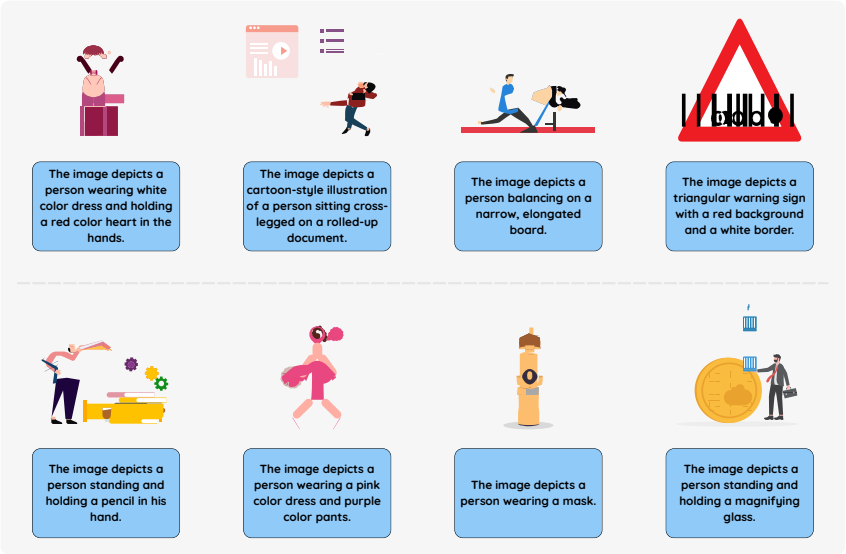


Figure 12: Failure cases on text-to-SVG task by vHector-8B followed by the corresponding short captions.

References

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

[2] 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)

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

[4] Hasler, D., Suesstrunk, S.: Measuring colourfulness in natural images. Proceedings of SPIE - The Inter. Soc. for Opt. Engin. (2003)

[5] 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)

[6] 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)

[7] 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)

[8] 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)

[9] 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)

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

[11] 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)

[12] 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)

[13] 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)