Reliable Reasoning in SVG-LLMs via Multi-Task Multi-Reward Reinforcement Learning

Haomin Wang[1,2], Qi Wei[2,3], Qianli Ma[1,2], Shengyuan Ding[2,4], Jinhui Yin[2,3], Kai Chen[2], and Hongjie Zhang[2⋆]

[1]Shanghai Jiao Tong University [2]Shanghai Artificial Intelligence Laboratory
[3]Nanjing University [4]Fudan University
{kiyotakawang,mqlqianli}@sjtu.edu.cn, {yinjinhui,chenkai}@pjlab.org.cn,
{ziqiwu5777,syding1105,nju.zhanghongjie}@gmail.com

Abstract. With the rapid advancement of vision–language models, an increasing number of studies have explored their potential for SVG generation tasks. Although existing approaches improve performance by constructing large-scale SVG datasets and introducing SVG-specific tokens, they still suffer from limited generalization, redundant paths in code outputs, and a lack of explicit reasoning. In this work, we present **CTRL-S** (**C**hain-of-**T**hought **R**einforcement **L**earning for **S**VG), a unified framework that introduces a chain-of-thought mechanism to explicitly expose the model's reasoning process during SVG generation. To support this structured reasoning, we construct **SVG-Sophia**, a high-quality dataset containing 145K samples across SVG code refinement, Text-to-SVG, and Image-to-SVG tasks. By training the model to generate group-level structured SVG code, CTRL-S significantly improves structural coherence and visual fidelity. Furthermore, we adopt the GRPO algorithm and design a multi-reward optimization framework, incorporating DINO, image–text similarity, format, and code efficiency rewards. Through joint multi-reward optimization and multi-task training, our approach systematically enhances overall generation capabilities. Extensive experiments show that CTRL-S outperforms existing methods, achieving higher task success rates, superior SVG code quality, and exceptional visual fidelity. Our code is available at https://github.com/hmwang2002/CTRL-S.

Keywords: SVG Generation · VLMs · Reinforcement Learning

1 Introduction

Scalable Vector Graphics (SVG) is an XML-based vector format that represents 2D content using parameterized geometric primitives rather than pixel grids, offering compact storage, resolution independence, and fine-grained editability. Owing to its seamless integration with modern front-end systems and interactive frameworks, SVG has become a fundamental graphic medium in web design, user interface development, scientific visualization, and computer-aided design.

⋆ Corresponding author

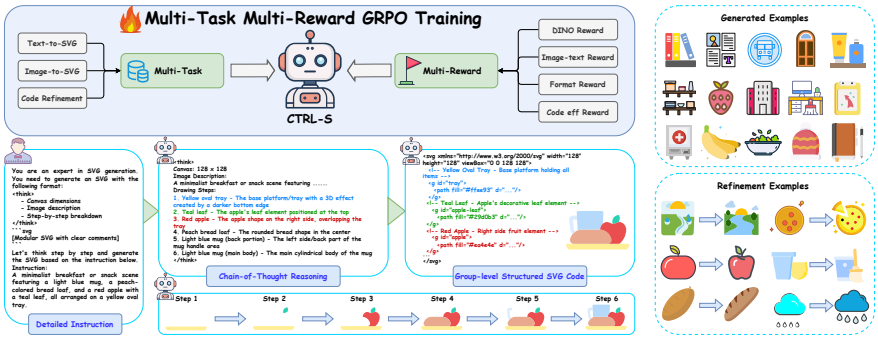


Fig. 1: Overview of CTRL-S. (Top Left) The Multi-Task Multi-Reward GRPO training framework integrates diverse generation tasks (Text-to-SVG, Image-to-SVG, and Code Refinement) guided by multiple rewards. **(Bottom Left)** During inference, CTRL-S leverages chain-of-thought reasoning to plan step-by-step drawing operations before generating the final group-level structured SVG code, ensuring a clear one-to-one correspondence between the reasoning steps and the generated code groups. **(Right)** Examples of high-quality generated SVGs and successful code refinement processes.

With the rapid development of vision-language models [2, 10, 17, 18, 35, 45], recent research has begun to explore their application to high-quality SVG code generation [25, 34, 38, 42]. By integrating vision encoders and SVG-specific tokens, these approaches significantly improve performance on Text-to-SVG and Image-to-SVG tasks. However, these approaches still suffer from limited generalization, frequently producing SVG programs with redundant paths. In addition, overly aggressive code compression during training degrades the readability and editability of the generated vector graphics. SVGen [33] and SVGThinker [4] introduce the chain-of-thought (CoT) reasoning into SVG generation by explicitly exposing intermediate reasoning steps to improve the quality of the generated SVG. However, they do not fully exploit the inherent grouping (`<g>`) structures in SVG code to organize components hierarchically, nor do they establish a clear alignment between reasoning steps and the corresponding grouped code segments, resulting in limited structural transparency and editability. While recent works like RLRF [26] and Reason-SVG [37] incorporate the GRPO algorithm [28] to leverage visual reward signals during post-training reinforcement learning, they primarily optimize individual tasks in isolation and lack a unified framework for jointly training Text-to-SVG and Image-to-SVG generation.

To address these limitations, we propose **CTRL-S**, a unified framework tailored for Text-to-SVG, Image-to-SVG, and SVG code refinement tasks. As illustrated in Figure 1, we integrate CoT reasoning into SVG generation to expose the model's planning processes. By leveraging the inherent grouping characteristics of SVG, we establish a step-wise alignment between the reasoning steps and the corresponding code groups. Furthermore, to break the isolation of prior works that exclusively focus on single-task optimization, we not only jointly train

the Text-to-SVG and Image-to-SVG tasks but also introduce an SVG code refinement task. By endowing the model with self-diagnostic and error-correction capabilities, these three tasks mutually reinforce each other within a single unified model.

To facilitate this unified paradigm, we first construct **SVG-Sophia**, a high-quality dataset that encompasses CoT question-answering pairs across the three tasks. Comprising 131K SFT samples and 14.4K RL samples, SVG-Sophia provides a solid foundation for CTRL-S to excel in these diverse domains. In the RL post-training phase, we address the limitations of conventional SFT, which relies solely on token-level supervision and lacks visual feedback. We introduce a multi-task, multi-reward optimization framework based on the GRPO algorithm. Specifically, we design four complementary rewards: (1) a format reward to ensure structural validity and renderability, (2) a DINO reward to encourage deep visual feature alignment between the rendered SVG and the reference image, (3) an image–text similarity reward to promote semantic consistency between the generated SVG and the input instruction, and (4) a code efficiency reward to penalize unnecessarily verbose SVG outputs and improve inference efficiency. This multi-reward optimization not only enhances visual fidelity but also mitigates the repetitive code generation commonly observed in prior SVG-LLM models, achieving a balanced trade-off between reasoning efficiency and generation quality. Extensive experiments show that our multi-task, multi-reward RL algorithm yields significant gains over SFT. Joint multi-task training further improves performance and generalization compared to single-task optimization. Moreover, the introduction of CoT enhances generation success and visual quality for complex geometries, while transforming the implicit generation process into explicit, structured code blocks, substantially improving the readability and editability of the resulting SVGs. In summary, our contributions are as follows:

1. We propose **CTRL-S**, a unified framework that integrates chain-of-thought reasoning and multi-task, multi-reward online RL for SVG code refinement, Text-to-SVG, and Image-to-SVG tasks.
2. We construct **SVG-Sophia**, a high-quality dataset providing explicit chain-of-thought supervision across three SVG tasks.
3. Extensive experiments show that our multi-task, multi-reward RL framework achieves substantial performance gains over SFT baselines. CTRL-S achieves state-of-the-art performance in SVG generation, delivering higher visual quality, faster inference, and highly readable and editable code.

2 Related Work

Optimization-based SVG Modeling. Optimization-based methods formulate SVG modeling as a parameter optimization problem rather than training a dedicated generative model. Early works such as DiffVG [13] and LIVE [16] leverage differentiable rasterization to directly optimize Bézier control points and styling attributes by minimizing pixel-level reconstruction losses. To incorporate

semantic supervision, CLIP-based approaches [7, 27, 30–32] replace pixel losses with image-text similarity objectives, enabling text-conditioned SVG generation without training. More recently, Score Distillation Sampling (SDS) [21] has been adopted to transfer diffusion priors into the vector graphics domain [11, 39–41, 44]. These methods optimize rendered SVGs through gradients derived from pretrained diffusion models, with later variants such as VPSD introducing particle-based distributional optimization to improve diversity and stability. Despite their strong visual fidelity, optimization-based approaches remain computationally intensive, instance-specific, and lack explicit hierarchical modeling of SVG structure, limiting scalability and downstream editability.

Learning-based SVG Modeling. Early learning-based methods represent SVG as sequences of geometric primitives with task-specific generative architectures [3, 9, 15, 23, 24, 29], ranging from sequential pen trajectories and latent-variable modeling to hierarchical VAEs with Transformer decoders. With the emergence of LLMs and VLMs, recent research has shifted toward semantically grounded SVG generation [4–6, 12, 14, 25, 26, 33, 34, 36–38, 42, 46]. Methods like StarVector [25], LLM4SVG [38], OmniSVG [42], and InternSVG [34] incorporate vision encoders and SVG-specific tokens to support Text-to-SVG and Image-to-SVG generation. Moreover, recent works such as SVGen [33] and SVGThinker [4] aim to introduce chain-of-thought reasoning into SVG generation by explicitly exposing intermediate reasoning steps, thereby improving performance. However, they fail to fully exploit the inherent grouping characteristics of SVG code to establish a one-to-one alignment between the intermediate planning steps and the generated code blocks.

Reinforcement Learning for SVG Generation. Beyond standard supervised fine-tuning, applying reinforcement learning (RL) during the post-training stage has emerged as a promising frontier for SVG generation. Recent works such as RLRF [26] and Reason-SVG [37] adopt the GRPO algorithm [28], introducing visual reward signals to further enhance generative quality. However, these approaches remain confined to single-task optimization, failing to unify Text-to-SVG and Image-to-SVG generation under a shared paradigm. In contrast, our CTRL-S introduces a unified, multi-task RL optimization framework that jointly aligns Text-to-SVG, Image-to-SVG, and SVG code refinement within a single unified model.

3 SVG-Sophia

We collect the original SVG files from the ColorSVG-100K [6] dataset and leverage Claude-Sonnet-4.5 [1] to annotate them into high-quality samples with explicit chain-of-thought reasoning and group-level structured SVG code. For Text-to-SVG generation, we construct 50K SFT samples and 5.5K RL samples. For Image-to-SVG generation, we similarly build 50K SFT samples and 5.5K RL samples, sharing the same underlying SVG programs as Text-to-SVG but differing in input modality. For SVG code refinement, we curate 31K SFT samples and 3.4K RL samples, along with a test set of 934 samples.

3.1 Task Definition

Let $\mathcal{M}$ denote the MLLM and I_{text} represent the user-provided textual instruction. For the **Text-to-SVG** generation task, the model is tasked with autoregressively generating a CoT planning sequence O_{think}, followed by the corresponding executable SVG code O_{svg}. This process is defined as:

$$(O_{\text{think}}, O_{\text{svg}}) = \mathcal{M}(I_{\text{text}}) \tag{1}$$

Similarly, for the **Image-to-SVG** generation task, the model is additionally conditioned on a reference image I_{image}. The task is formulated as:

$$(O_{\text{think}}, O_{\text{svg}}) = \mathcal{M}(I_{\text{text}}, I_{\text{image}}) \tag{2}$$

To empower the model with self-correction and optimization capabilities, we introduce the **SVG code refinement** task. In this setting, the model is provided with a textual instruction I_{text}, a reference image I_{image}, and a flawed SVG code draft I_{draft} to be refined:

$$(O_{\text{think}}, O_{\text{refined}}) = \mathcal{M}(I_{\text{text}}, I_{\text{image}}, I_{\text{draft}}) \tag{3}$$

3.2 Data Annotation Pipeline

The raw SVG files are initially collected from the ColorSVG-100K [6] dataset and then normalized to a 128×128 viewBox. We employ Claude-Sonnet-4.5 [1] to annotate detailed image captions from the rendered vector graphics. Subsequently, we prompt Claude-Sonnet-4.5 with both the generated caption and the raw SVG code, instructing it to refactor the original code into a highly structured format, enriched with descriptive comments and semantic group-level hierarchies, while also producing a step-by-step reasoning process that outlines its planning procedure. To ensure strict visual fidelity and eliminate failed refactoring attempts, we filter the refactored SVGs by retaining only those achieving an SSIM ≥ 0.95 against their original renderings. To further ensure annotation quality, we engage 100 human annotators to review all annotated samples, manually correcting any captions that inaccurately describe the visual content or CoT reasoning steps that fail to correspond to the generated code groups. Finally, we use the generated image captions as user instructions and treat the CoT reasoning along with the reconstructed structured SVG code produced by Claude-Sonnet-4.5 as the ground-truth responses for Text-to-SVG and Image-to-SVG tasks.

For the SVG code refinement task, we first train a Qwen3-VL-8B model [2] on the annotated Text-to-SVG and Image-to-SVG data, and use it to generate draft SVG programs on the training set. We then retain only moderately flawed samples ($0.30 \leq \text{SSIM} \leq 0.95$) against the ground truth. Claude-Sonnet-4.5 is then prompted with the defective and ground-truth images to produce discrepancy analysis and correction-oriented CoT reasoning. Rule-based filtering is further applied to remove invalid annotations, such as cases claiming complete consistency or providing irrelevant analysis. To mitigate potential annotator bias, 100

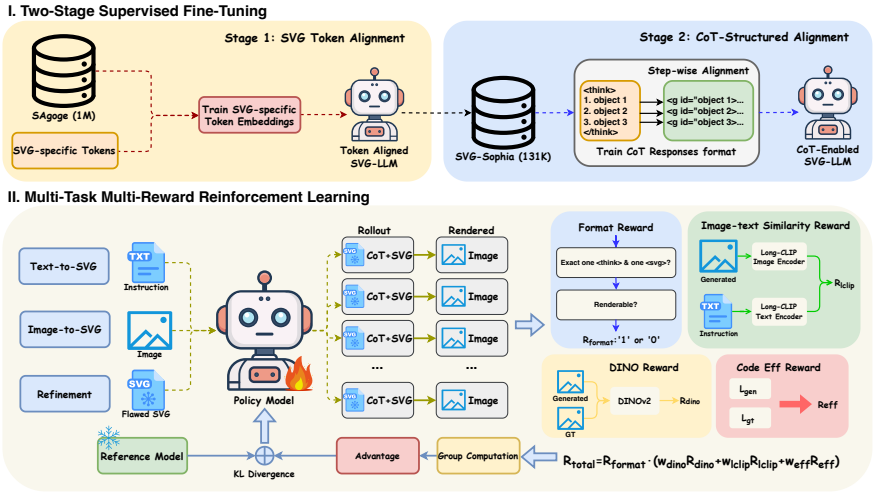


Fig. 2: The overall pipeline of CTRL-S. (1) Two-Stage SFT: The model is first trained on 1M SAgoge samples to align SVG-specific tokens, and then fine-tuned on SVG-Sophia to learn CoT-structured responses with explicit step-wise planning. **(2) Multi-Task Multi-Reward RL:** We jointly optimize Text-to-SVG, Image-to-SVG, and SVG refinement tasks via a multi-reward mechanism, including Format Reward, DINO Reward, Image-text Similarity Reward, and Code Efficiency Reward, to improve structural validity, visual fidelity, semantic alignment, and concise code generation.

human annotators further review all refinement annotations, manually correcting cases where the identified defects or correction reasoning are inaccurate or task-irrelevant. For the test set, we select non-overlapping SVG programs from ColorSVG-100K and apply the same annotation pipeline. Defective drafts are generated using the SFT-trained Qwen3-VL-8B, as well as Claude-Sonnet-4.5, Gemini-3-Pro [8], GPT-5.2 [19], and Qwen3-VL-235B-A22B [2], to ensure a fair and unbiased evaluation.

4 CTRL-S

Figure 2 illustrates the overall pipeline of CTRL-S. Our framework begins with a two-stage supervised fine-tuning to align SVG-specific tokens and establish step-wise chain-of-thought reasoning. Subsequently, a multi-task, multi-reward reinforcement learning phase jointly optimizes Text-to-SVG, Image-to-SVG, and code refinement tasks via comprehensive feedback signals.

4.1 Preliminary

Notation and Problem Formulation. We formulate SVG generation as a unified multi-task sequence-to-sequence autoregressive generation problem. Let

$\mathcal{M}$ (defined in Sec. 3.1) parameterized by θ denote our MLLM. Depending on the specific task, the model is conditioned on a varying set of inputs to generate a target sequence $y = (y_1, y_2, \ldots, y_T)$, which consists of a chain-of-thought reasoning sequence O_{think} followed by the executable SVG code (O_{svg} or O_{refined}). To unify our three core tasks, c encapsulates varying inputs: $c = I_{\text{text}}$ for *Text-to-SVG*, $c = (I_{\text{text}}, I_{\text{image}})$ for *Image-to-SVG*, and $c = (I_{\text{text}}, I_{\text{image}}, I_{\text{draft}})$ for *SVG Code Refinement*. Given the task-specific context c, the generation probability of the output sequence y is factorized as:

$$P(y|c) = \prod_{t=1}^{T} \pi_\theta(y_t|c, y_{<t}), \tag{4}$$

where $y_{<t}$ represents the sequence of tokens generated prior to step t. The model, typically initialized after multi-task SFT, serves as our reference policy π_{ref} for the reinforcement learning phase.

Group Relative Policy Optimization (GRPO). To efficiently optimize the MLLM across diverse tasks without the memory overhead of a parameterized value model, we employ GRPO [28]. For a given context c, the current policy $\pi_{\theta_{old}}$ samples a group of G diverse output trajectories $\mathcal{G} = \{y^{(1)}, \ldots, y^{(G)}\}$. Each trajectory $y^{(i)}$ is evaluated by our multi-reward function to yield a score r_i. GRPO computes the relative advantage by normalizing these rewards within the group: $\hat{A}_i = (r_i - \mu_\mathcal{G})/(\sigma_\mathcal{G} + \epsilon)$. The policy π_θ is then optimized by maximizing a clipped surrogate objective, augmented with a Kullback-Leibler (KL) divergence penalty to mitigate excessive deviation from π_{ref}:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{c\sim\mathcal{D},\mathcal{G}\sim\pi_{\theta_{old}}} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|y^{(i)}|} \sum_{t=1}^{|y^{(i)}|} \mathcal{L}_{clip}^{i,t}(\theta) - \beta \mathbb{D}_{KL}(\pi_\theta \| \pi_{ref}) \right], \tag{5}$$

where the clipped likelihood ratio is defined as

$$\mathcal{L}_{clip}^{i,t}(\theta) = \min \left(\rho_{i,t}(\theta)\hat{A}_i, \text{clip}(\rho_{i,t}(\theta), 1-\gamma, 1+\gamma)\hat{A}_i \right), \tag{6}$$

and $\rho_{i,t}(\theta) = \frac{\pi_\theta(y_t^{(i)}|c, y_{<t}^{(i)})}{\pi_{\theta_{old}}(y_t^{(i)}|c, y_{<t}^{(i)})}$ is the probability ratio of generating the t-th token under the current versus the old policy.

4.2 Two-Stage Supervised Fine-Tuning

To establish a robust initialization for the subsequent reinforcement learning phase, CTRL-S adopts the SVG-specific token design introduced in InternSVG [34] (detailed in the Appendix) and undergoes a two-stage SFT process. In the first stage, we stabilize the embeddings of the SVG-specific tokens by sampling 1M training instances from the SAgoge dataset [34]. Following this modality alignment, the second stage utilizes the SFT split of the SVG-Sophia dataset to train the model. This phase introduces a strict step-wise alignment, where each intermediate reasoning step in the CoT explicitly corresponds to a hierarchically organized, group-level (`<g>`) structural block in the resulting SVG, ensuring that the SVG generation process is both interpretable and logically transparent.

4.3 Multi-Reward Design for Reinforcement Learning in CTRL-S

Following the SFT phase, we employ reinforcement learning to further align the model's generation with visual, semantic, and structural objectives. To provide comprehensive guidance without relying on costly human annotations, we design a multi-reward framework comprising four complementary components.

Format Reward (R_{format}) To guarantee both structural compliance and execution validity, we introduce a binary format reward R_{format}. The reward yields 1 if the model's output strictly contains exactly one `<think>...</think>` reasoning block followed by a single SVG code block that can be rendered by CairoSVG successfully, and 0 otherwise.

DINO Reward (R_{dino}) A primary limitation of standard SFT is its inherent reliance on token-level textual supervision, which lacks the capacity to penalize *global visual discrepancies*. For SVG-related tasks, explicit pixel-level feedback is crucial to enhance the overall visual fidelity of the generated graphics. To address this, we introduce R_{dino}. Specifically, the generated SVG code is first rasterized into an image V_{gen}. We then compute the feature similarity between this rendering and the ground-truth image V_{gt} using a pre-trained DINOv2 [20] model, capturing deep, structural visual alignments. Formally, let $\mathcal{E}_{\text{DINO}}$ denote the DINOv2 feature extractor; the reward is formulated as the normalized cosine similarity between the two image embeddings:

$$R_{\text{dino}} = \frac{1}{2}(\cos(\mathcal{E}_{\text{DINO}}(V_{\text{gen}}), \mathcal{E}_{\text{DINO}}(V_{\text{gt}})) + 1). \tag{7}$$

Image-text Similarity Reward (R_{lclip}) Beyond low-level visual fidelity (Eq. 7), the generated SVG must semantically align with the user's high-level textual instruction I_{text}. Considering that the instructions in SVG-Sophia typically consist of several detailed descriptive sentences, the standard CLIP model [22], bounded by its strict 77-token input limit, often truncates crucial structural details and fails to adequately capture fine-grained semantics in long contexts. To overcome this, we adopt Long-CLIP [43] to compute the semantic alignment reward R_{lclip}. By leveraging the Long-CLIP image encoder $\mathcal{E}_{\text{L-img}}$ and text encoder $\mathcal{E}_{\text{L-text}}$, we project both the rendered image V_{gen} and the instruction I_{text} into a shared embedding space. The reward is computed as follows:

$$R_{\text{lclip}} = \frac{\mathcal{E}_{\text{L-img}}(V_{\text{gen}})}{\|\mathcal{E}_{\text{L-img}}(V_{\text{gen}})\|_2} \cdot \frac{\mathcal{E}_{\text{L-text}}(I_{\text{text}})}{\|\mathcal{E}_{\text{L-text}}(I_{\text{text}})\|_2}. \tag{8}$$

Code Efficiency Reward (R_{eff}) During the generation of SVG code, SFT models frequently suffer from a repetition problem, producing excessively long, redundant, and invalid code that significantly degrades inference speed. To mitigate this issue, we adapt a length-based penalty inspired by RLRF [26]. Specifically, let L_{gt} and L_{gen} denote ground-truth and generated SVG code lengths, the code efficiency reward R_{eff} is formulated as follows:

$$R_{\text{eff}} = 1 - (\frac{1}{L_{\text{gt}}} \max(0, L_{\text{gen}} - \frac{L_{\text{gt}}}{2}))^2. \tag{9}$$

Total Reward (R_{total}) Finally, we aggregate the visual (Eq. 7), semantic (Eq. 8), and efficiency objectives (Eq. 9) into a unified multi-reward formulation. Crucially, the binary format reward R_{format} acts as a multiplicative gating factor, ensuring that unrenderable or structurally malformed outputs receive a total reward of zero, preventing degenerate policy updates. The final reward is defined as:

$$R_{\text{total}} = R_{\text{format}} \cdot (w_{\text{dino}} R_{\text{dino}} + w_{\text{lclip}} R_{\text{lclip}} + w_{\text{eff}} R_{\text{eff}}) . \tag{10}$$

Empirically, we set the trade-off weights as $w_{\text{dino}} : w_{\text{lclip}} : w_{\text{eff}} = 2 : 1 : 1$.

5 Experiments

5.1 Experimental Setup

Building upon Qwen3-VL-8B-Instruct, CTRL-S initially undergoes a two-stage SFT process, as detailed in Sec. 4.2. We set the learning rate to 1e-4 in the first stage and decrease it to 5e-5 in the second stage. The training is performed on 48 H200 GPUs with a global batch size of 96. In the RL stage, we optimize the model using the GRPO algorithm implemented via the `verl` framework. The RL training is performed on 32 GPUs with a global batch size of 128 and a learning rate of 1e-5. During the rollout phase, we sample 16 responses per prompt. The model is trained for 2 epochs, and the entire RL training process takes approximately 12 hours.

5.2 Quantitative Evaluations

As shown in Table 1, CTRL-S achieves leading performance on both the Text-to-SVG and Image-to-SVG tasks on the SArena-Icon [34] benchmark. For Text-to-SVG, CTRL-S attains the highest CLIP-T2I score of 25.944, demonstrating the outstanding semantic understanding and text-to-image alignment capabilities of the model. Notably, compared with the CoT-based methods SVGen [33] and SVGThinker [4], CTRL-S consistently achieves better results across FID, CLIP-T2I, CLIP-I2I, and SR, while generating substantially fewer tokens. For Image-to-SVG, our model obtains the best results across the DINO, SSIM, and LPIPS metrics compared to mainstream general VLMs and SVG-LLMs. This proves that CTRL-S exhibits extremely high visual fidelity in accurately reconstructing geometric shapes and colors. Furthermore, compared to our SFT baseline, CTRL-S (SFT + RL) not only gains significant enhancements across all visual metrics but also substantially increases the success rate and greatly reduces the number of generated tokens. This validates the effectiveness of our proposed multi-task, multi-reward RL training framework. The framework successfully enhances visual fidelity and semantic alignment while simultaneously empowering the model with stronger generalization capabilities, robustness, and superior code generation efficiency.

Table 1: SVG generation results on SArena-Icon. SR denotes Success Rate. CTRL-S (SFT) denotes the model after two-stage SFT, CTRL-S (SFT+RL) denotes the full model after RL post-training.

Model	Data	Text-to-SVG					Image-to-SVG				
		FID ↓	CLIP-T2I ↑	CLIP-I2I ↑	SR	Tokens	DINO ↑	SSIM ↑	LPIPS ↓	SR	Tokens
Traditional SVG Methods											
IconShop	–	32.288	20.894	70.922	86.51%	1.6k	–	–	–	–	–
VectorFusion	–	16.594	22.992	70.308	100.0%	33k	–	–	–	–	–
SVGDreamer	–	26.612	20.329	69.975	100.0%	132k	–	–	–	–	–
DiffVG	–	–	–	–	–	–	0.869	0.927	0.097	100.0%	17k
LIVE	–	–	–	–	–	–	0.973	0.986	0.024	100.0%	18k
VTracer	–	–	–	–	–	–	0.966	0.875	0.054	100.0%	4.4k
Vision-Language Models											
Llama 4 Scout	–	17.908	22.849	73.563	96.03%	256	0.844	0.582	0.346	97.61%	246
Llama 4 Maverick	–	14.931	23.570	75.816	98.12%	265	0.863	0.596	0.329	97.75%	255
Qwen3-VL-235B-A22B	–	18.404	24.024	76.289	94.96%	326	0.927	0.718	0.210	95.28%	679
GPT-5.2	–	14.965	25.457	82.701	97.02%	623	0.936	0.653	0.269	98.87%	497
Claude-Sonnet-4.5	–	16.451	25.322	81.038	99.63%	320	0.910	0.677	0.253	97.95%	705
Gemini-3-Pro	–	13.203	25.731	84.157	96.91%	397	0.940	0.723	0.198	97.14%	769
SVG-LLMs											
StarVector-8B	2.2M	–	–	–	–	–	0.871	0.623	0.206	72.51%	951
LLM4SVG-7B	580K	21.939	19.458	70.726	90.95%	705	0.748	0.472	0.409	95.53%	485
OmniSVG-3B	2M	28.292	21.679	74.831	99.68%	1.8k	0.894	0.756	0.186	99.97%	2.4k
InternSVG-8B	16M	8.715	23.916	80.911	97.24%	1.0k	0.949	0.811	0.127	94.45%	1.3k
SVGen	1M	23.019	21.715	75.341	85.10%	1.5k	–	–	–	–	–
SVGThinker	270K	12.276	23.573	78.076	98.87%	794	–	–	–	–	–
Ours											
Qwen3-VL-8B	–	16.875	22.578	73.187	99.55%	374	0.796	0.500	0.354	74.02%	474
CTRL-S (SFT)	131K	18.154	23.655	77.959	92.02%	1.1k	0.903	0.717	0.187	90.12%	1.4k
CTRL-S (SFT+RL)	145K	**11.584**	**25.944**	**82.291**	**99.85%**	346	**0.980**	**0.835**	**0.098**	**99.93%**	512
Δ Improvement		*6.570*	*2.289*	*4.332*			*0.077*	*0.118*	*0.089*		

As shown in Table 2, on the SVG-Sophia Code Refinement Benchmark, CTRL-S achieves the best performance across all evaluation metrics compared with state-of-the-art proprietary models, including GPT-5.2, Claude-Sonnet-4.5, and Gemini-3-Pro. These results highlight the superiority of CTRL-S in structural understanding and fine-grained code editing. Specifically, starting from imperfect SVG programs, our method accurately locates defective components and applies targeted corrections, achieving more precise structural refinement while preserving semantic consistency. This indicates that the model not only possesses a strong understanding of the visual information encoded in SVG programs but also demonstrates stable, iterative code-optimization capability. Furthermore, compared with the SFT baseline, the RL-enhanced model achieves a substantial improvement in task success rate while significantly reducing the number of generated tokens. This suggests better generalization and a reduced tendency to produce redundant code, leading to more efficient and controllable refinement.

5.3 Qualitative Evaluations

As shown in Figure 3, we present qualitative comparisons between CTRL-S and representative baselines across three tasks. For Text-to-SVG and Image-to-SVG tasks (Figures 3a and 3b), CTRL-S consistently generates SVG with accurate structural layouts and fine-grained visual details. For Text-to-SVG, CTRL-S ac-

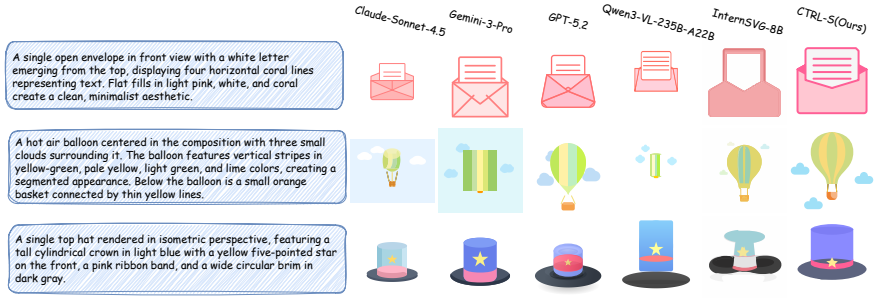
Table 2: SVG refinement results on SVG-Sophia Code Refinement Benchmark.

Model	DINO ↑	SSIM ↑	LPIPS ↓	SR	Tokens
Vision-Language Models					
Llama 4 Scout	0.840	0.634	0.383	97.00%	788
Llama 4 Maverick	0.845	0.615	0.377	94.75%	616
Qwen3-VL-235B-A22B	0.809	0.515	0.403	77.84%	840
GPT-5.2	0.911	0.640	0.342	99.26%	975
Claude-Sonnet-4.5	0.796	0.579	0.401	84.58%	790
Gemini-3-Pro	0.883	0.593	0.284	81.37%	992
Ours					
Qwen3-VL-8B	0.796	0.501	0.410	76.77%	980
CTRL-S (SFT)	0.888	0.665	0.236	84.37%	2.9k
CTRL-S (SFT+RL)	**0.951**	**0.765**	**0.180**	**99.79%**	866
Δ Improvement	*0.067*	*0.126*	*0.104*		

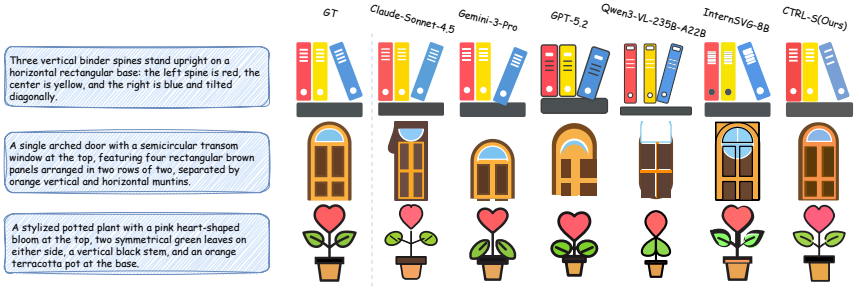
curately renders complex multi-element scenes, such as a striped hot air balloon with surrounding clouds and a suspended basket. In contrast, competing methods frequently distort the global structure, omit essential components, or fail to reproduce the correct color patterns. For Image-to-SVG, CTRL-S faithfully reconstructs reference images while preserving color attributes and spatial arrangements. In contrast, general VLMs often produce geometries that are structurally inconsistent and physically implausible, while SVG-LLMs such as InternSVG-8B exhibit noticeable deviations in stroke thickness and shape boundaries. For SVG code refinement (Figure 3c), CTRL-S demonstrates strong self-correction capability by accurately identifying structural defects in the draft SVG and applying targeted modifications. In contrast, general VLMs often fail to precisely localize defective components, resulting in incomplete or globally inconsistent corrections. Starting from imperfect drafts, CTRL-S successfully recovers missing components and corrects spatial misalignments, producing refined outputs that closely match the ground truth. In Figure 4, we further illustrate the progressive improvement of CTRL-S throughout RL training. As the number of training steps increases, the generated SVGs exhibit increasingly accurate structural layouts, more faithful color reproduction, and richer fine-grained details, demonstrating the effectiveness of our multi-reward optimization in iteratively refining the model's generation capability.

5.4 Ablation Studies

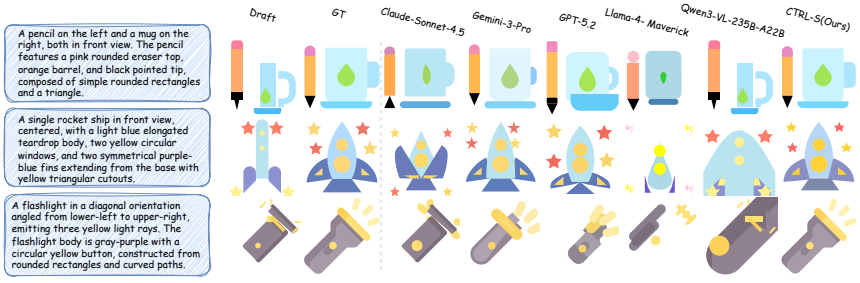
Benefits of Chain-of-Thought Reasoning. To evaluate the impact of CoT reasoning on model performance, we construct a non-CoT dataset by directly using the original SVG codes from ColorSVG without reconstruction by Claude-Sonnet-4.5. Based on this dataset, we train baseline SFT and RL models without



(a) Qualitative comparison on Text-to-SVG generation.



(b) Qualitative comparison on Image-to-SVG generation.



(c) Qualitative comparison on SVG code refinement.

Fig. 3: Qualitative comparisons of SVG generation and code refinement between baselines and CTRL-S.

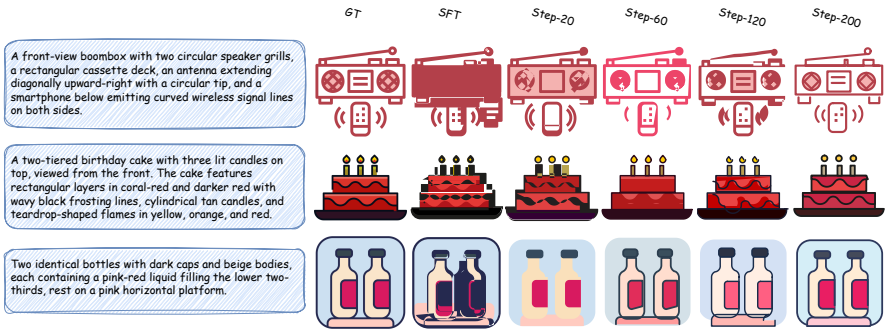


Fig. 4: Qualitative visualization of SVG generation quality across RL training steps.

CoT capabilities. As shown in Table 3a, for the SFT models, introducing CoT reasoning substantially improves the task success rate. This indicates that incorporating explicit reasoning enhances generation robustness, enabling the model to produce more renderable and syntactically valid SVG programs. Moreover, for both SFT and RL settings, the CoT-enabled models consistently outperform their non-CoT counterparts across all visual quality metrics. These results suggest that CoT reasoning plays a critical role not only in improving generation stability but also in enhancing visual fidelity and semantic alignment.

Ablation on Multi-Reward Functions. To systematically investigate the specific contributions of individual reward functions, we conduct an ablation study on the reward mechanism, sequentially comparing the performance of models trained with: (1) only the Format and DINO Rewards, (2) the addition of the Image-Text Similarity Reward, and (3) the full composite reward. As demonstrated in Table 3b, for the Text-to-SVG task, incorporating the Image-Text Similarity Reward significantly enhances the model's semantic comprehension and text-image alignment capabilities, improving the CLIP-T2I score from 24.573 to 25.444 and the CLIP-I2I score from 80.897 to 82.070. More importantly, with the further integration of the Code Efficiency Reward, the model effectively mitigates the issue of code redundancy. It not only maintains superior image generation quality but also significantly accelerates the per-sample inference speed, reducing the generated tokens from 701 to 346 and the inference time from 7.121 to 4.439 seconds per SVG. Ultimately, this achieves an optimal balance between visual fidelity and inference efficiency.

Ablation on Reward Ratio. As demonstrated in Table 3c, we empirically evaluate different weight ratios among w_{dino}, w_{lclip}, and w_{eff}. Our findings indicate that the 2:1:1 configuration achieves the most effective balance among visual quality, semantic alignment, and code efficiency. Specifically, increasing w_{dino} favors structural consistency but leads to longer outputs, while reducing it harms geometric fidelity. We therefore adopt 2:1:1 in all experiments.

Effects of Multi-task Training. We analyze the impact of joint training across three tasks. As shown in Table 3d, compared with single-task training,

Table 3: Ablation studies on CoT, reward signals, reward ratios (w_{dino}:w_{lclip}:w_{eff}), and multi-task training.

(a) Ablation on Chain-of-Thought (CoT).

Model	Text-to-SVG					Image-to-SVG				
	FID ↓	CLIP-T2I ↑	CLIP-I2I ↑	SR	Tokens	DINO ↑	SSIM ↑	LPIPS ↓	SR	Tokens
w/o CoT (SFT)	22.960	23.002	76.567	85.75%	1.0k	0.887	0.663	0.204	81.96%	1.3k
w/ CoT (SFT)	18.154	23.655	77.959	92.02%	1.1k	0.903	0.717	0.187	90.12%	1.4k
w/o CoT (SFT+RL)	12.161	24.278	80.604	99.47%	274	0.961	0.813	0.125	99.53%	466
w/ CoT (SFT+RL)	**11.584**	**25.944**	**82.291**	**99.85%**	346	**0.980**	**0.835**	**0.098**	**99.93%**	512

(b) Ablation on multi-reward functions.

Reward	FID ↓	CLIP-T2I ↑	CLIP-I2I ↑	Tokens	Time (s/svg)
$R_{\text{format}} \cdot R_{\text{dino}}$	12.443	24.573	80.897	606	6.151
$R_{\text{format}} \cdot (R_{\text{dino}} + R_{\text{lclip}})$	11.831	25.444	82.070	701	7.121
Full R_{total} (Ours)	**11.584**	**25.944**	**82.291**	**346**	**4.439**

(c) Ablation on reward ratio.

Ratio	Text-to-SVG				Image-to-SVG			
	FID ↓	CLIP-T2I ↑	CLIP-I2I ↑	Tokens	DINO ↑	SSIM ↑	LPIPS ↓	Tokens
1:1:1	12.575	24.623	80.849	333	0.962	0.818	0.120	532
5:3:1	11.642	24.653	81.336	428	0.967	0.830	0.109	649
2:1:1 (Ours)	**11.584**	**25.944**	**82.291**	346	**0.980**	**0.835**	**0.098**	**512**

(d) Ablation on multi-task training across Text-to-SVG, Image-to-SVG, and Refinement tasks. T2S denotes Text-to-SVG, I2S denotes Image-to-SVG, and Refine denotes SVG code refinement.

Training	Text-to-SVG			Image-to-SVG			Refinement		
	FID ↓	CLIP-T2I ↑	CLIP-I2I ↑	DINO ↑	SSIM ↑	LPIPS ↓	DINO ↑	SSIM ↑	LPIPS ↓
T2S Only	12.321	24.431	80.651	–	–	–	–	–	–
I2S Only	–	–	–	0.959	0.824	0.119	–	–	–
Refine Only	–	–	–	–	–	–	0.940	0.753	0.187
T2S + I2S	11.637	24.804	81.433	0.966	0.824	0.114	–	–	–
Multi-Task (Ours)	**11.584**	**25.944**	**82.291**	**0.980**	**0.835**	**0.098**	**0.951**	**0.765**	**0.180**

multi-task learning significantly improves the overall quality of generated SVG. Jointly training Text-to-SVG and Image-to-SVG enhances cross-modal semantic alignment, leading to better consistency between textual instructions, input images, and generated SVG. Furthermore, incorporating the SVG code refinement task brings substantial gains to the Image-to-SVG setting. This suggests that learning to analyze rendered images from imperfect SVG and perform targeted SVG correction strengthens the model's ability to reconstruct vector graphics from images. Overall, these results indicate that the three tasks provide complementary supervision signals, and their joint optimization leads to more robust and generalized SVG modeling.

Generalization to Complex Vector Graphics. We further evaluate CTRL-S on SArena-Illustration [34]. Compared with icons, illustrations are considerably more complex, exhibiting richer compositional structures and thus requiring much longer SVG sequences. As shown in Table 4, without training on any illustration data, CTRL-S surpasses the previous best result on every metric. On Text-to-SVG, it achieves the best FID (21.832 vs 22.397), CLIP-T2I (25.138 vs 24.800), and CLIP-I2I (75.083 vs 74.796). On Image-to-SVG, it leads across DINO (0.952 vs 0.924), SSIM (0.746 vs 0.716), and LPIPS (0.175 vs 0.188). Notably, CTRL-S achieves these gains while producing far more compact code, using only 1.4K tokens on average, compared with the roughly 8K tokens required by InternSVG-8B.

Table 4: SVG generation results on SArena-Illustration. SR denotes Success Rate.

Model	Text-to-SVG					Image-to-SVG				
	FID ↓	CLIP-T2I ↑	CLIP-I2I ↑	SR	Tokens	DINO ↑	SSIM ↑	LPIPS ↓	SR	Tokens
Vision-Language Models										
Llama 4 Scout	35.489	20.299	64.182	91.80%	524	0.807	0.599	0.360	92.80%	574
Llama 4 Maverick	30.835	21.872	67.366	97.30%	551	0.839	0.644	0.340	98.20%	608
Qwen3-VL-235B-A22B	32.444	21.410	69.140	94.00%	845	0.853	0.611	0.310	87.46%	1.5k
GPT-4o	28.124	23.637	70.696	99.75%	473	0.850	0.663	0.327	99.35%	484
Claude-Sonnet-4	27.294	23.094	74.525	98.85%	1.0k	0.901	0.670	0.305	99.25%	1.3k
Gemini-2.5-Flash	28.865	24.800	74.796	96.50%	1.2k	0.829	0.516	0.359	77.41%	1.8k
SVG-LLMs										
StarVector-8B	–	–	–	–	–	0.650	0.070	0.447	8.05%	2.6k
LLM4SVG-7B	48.704	15.468	62.933	79.36%	1.2k	0.713	0.494	0.413	96.60%	476
OmniSVG-3B	42.756	16.861	64.815	99.75%	4.5k	0.797	0.656	0.330	100.0%	6.7k
InternSVG-8B	22.397	21.116	74.662	97.95%	8.1k	0.924	0.716	0.188	90.01%	7.7k
CTRL-S (Ours)	**21.832**	**25.138**	**75.083**	99.45%	952	**0.952**	**0.746**	**0.175**	99.93%	1.4k

6 Conclusion

In this work, we present CTRL-S, a unified framework that introduces chain-of-thought (CoT) reasoning into Text-to-SVG, Image-to-SVG, and SVG code refinement tasks. By leveraging the hierarchical grouping structure inherent in SVG, we explicitly align planning steps with code groups, thereby improving the structural consistency, readability, and editability of generated SVG. Building upon this design, we propose a multi-task, multi-reward GRPO training paradigm that jointly optimizes complementary tasks under diverse reward signals. Through cross-task collaboration and multi-perspective reward guidance, our approach significantly enhances visual fidelity, semantic alignment, and generation stability. In addition, we construct the SVG-Sophia dataset, a high-quality SVG corpus augmented with explicit CoT question–answer pairs, providing systematic training resources for structured SVG generation and code refinement. Extensive experimental results demonstrate that CTRL-S consistently achieves state-of-the-art performance across multiple tasks, validating the effectiveness of structured reasoning and multi-task, multi-reward optimization for vector graphic modeling. Overall, this work establishes a new training paradigm for reliable reasoning in SVG-LLMs and lays the foundation for future research in complex vector graphic generation and editing.

Acknowledgements

This work was supported by the Shanghai Artificial Intelligence Laboratory and the Shanghai Committee of Science and Technology (No. 22YF1461500).

References

1. Anthropic: Introducing claude sonnet 4.5. `https://www.anthropic.com/news/claude-sonnet-4-5` (2025)

2. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)
3. Carlier, A., Danelljan, M., Alahi, A., Timofte, R.: Deepsvg: A hierarchical generative network for vector graphics animation. Advances in Neural Information Processing Systems **33**, 16351–16361 (2020)
4. Chen, H., Zhao, Z., Chen, Y., Liang, Z., Ni, B.: Svgthinker: Instruction-aligned and reasoning-driven text-to-svg generation. In: Proceedings of the 33rd ACM International Conference on Multimedia. pp. 11004–11012 (2025)
5. Chen, S., Dong, X., Xu, H., Wu, X., Tang, F., Zhang, H., Yan, Y., Wu, L., Zhang, W., Hou, G., et al.: Svgenius: Benchmarking llms in svg understanding, editing and generation. In: Proceedings of the 33rd ACM International Conference on Multimedia. pp. 13289–13296 (2025)
6. Chen, Z., Pan, R.: Svgbuilder: Component-based colored svg generation with text-guided autoregressive transformers. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 2358–2366 (2025)
7. Frans, K., Soros, L., Witkowski, O.: Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. Advances in Neural Information Processing Systems **35**, 5207–5218 (2022)
8. Google DeepMind: Gemini 3 pro model card. `https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf` (2025)
9. Ha, D., Eck, D.: A neural representation of sketch drawings. arXiv preprint arXiv:1704.03477 (2017)
10. Hurst, A., Lerer, A., Goucher, A.P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al.: Gpt-4o system card. arXiv preprint arXiv:2410.21276 (2024)
11. Jain, A., Xie, A., Abbeel, P.: Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1911–1920 (2023)
12. Li, J., Yu, J., Wei, C., Dong, H., Lin, Q., Yang, L., Wang, Z., Hao, Y.: Unisvg: A unified dataset for vector graphic understanding and generation with multimodal large language models. In: Proceedings of the 33rd ACM International Conference on Multimedia. pp. 13156–13163 (2025)
13. Li, T.M., Lukáč, M., Gharbi, M., Ragan-Kelley, J.: Differentiable vector graphics rasterization for editing and learning. ACM Transactions on Graphics (TOG) **39**(6), 1–15 (2020)
14. Liang, G., Wang, Z., Hu, J., Zhou, H., Xue, Z., Zhang, J., Xu, D., Yu, Q.: Render-in-the-loop: Vector graphics generation via visual self-feedback. arXiv preprint arXiv:2604.20730 (2026)
15. Lopes, R.G., Ha, D., Eck, D., Shlens, J.: A learned representation for scalable vector graphics. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7930–7939 (2019)
16. Ma, X., Zhou, Y., Xu, X., Sun, B., Filev, V., Orlov, N., Fu, Y., Shi, H.: Towards layer-wise image vectorization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16314–16323 (2022)
17. Meta AI: Llama 4 maverick 17b-128e (model card). `https://huggingface.co/meta-llama/Llama-4-Maverick-17B-128E` (2025)
18. Meta AI: Llama 4 scout 17b-16e (model card). `https://huggingface.co/meta-llama/Llama-4-Scout-17B-16E` (2025)
19. OpenAI: Introducing gpt 5.2. `https://openai.com/index/introducing-gpt-5-2/` (2025)

20. 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. arXiv preprint arXiv:2304.07193 (2023)
21. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988 (2022)
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: International conference on machine learning. pp. 8748–8763. PmLR (2021)
23. Reddy, P., Gharbi, M., Lukac, M., Mitra, N.J.: Im2vec: Synthesizing vector graphics without vector supervision. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 7342–7351 (2021)
24. Ribeiro, L.S.F., Bui, T., Collomosse, J., Ponti, M.: Sketchformer: Transformer-based representation for sketched structure. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 14153–14162 (2020)
25. 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: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 16175–16186 (2025)
26. Rodriguez, J.A., Zhang, H., Puri, A., Pramanik, R., Feizi, A., Wichmann, P., Mondal, A.K., Samsami, M.R., Awal, R., Taslakian, P., et al.: Rendering-aware reinforcement learning for vector graphics generation. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems (2025)
27. Schaldenbrand, P., Liu, Z., Oh, J.: Styleclipdraw: Coupling content and style in text-to-drawing translation. arXiv preprint arXiv:2202.12362 (2022)
28. Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al.: Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300 (2024)
29. Shen, I.C., Chen, B.Y.: Clipgen: A deep generative model for clipart vectorization and synthesis. IEEE Transactions on Visualization and Computer Graphics **28**(12), 4211–4224 (2021)
30. Song, Y., Shao, X., Chen, K., Zhang, W., Jing, Z., Li, M.: Clipvg: Text-guided image manipulation using differentiable vector graphics. In: Proceedings of the AAAI conference on artificial intelligence. vol. 37, pp. 2312–2320 (2023)
31. Vinker, Y., Alaluf, Y., Cohen-Or, D., Shamir, A.: Clipascene: Scene sketching with different types and levels of abstraction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4146–4156 (2023)
32. Vinker, Y., Pajouheshgar, E., Bo, J.Y., Bachmann, R.C., Bermano, A.H., Cohen-Or, D., Zamir, A., Shamir, A.: Clipasso: Semantically-aware object sketching. ACM Transactions on Graphics (TOG) **41**(4), 1–11 (2022)
33. Wang, F., Zhao, Z., Liu, Y., Zhang, D., Gao, J., Sun, H., Li, X.: Svgen: Interpretable vector graphics generation with large language models. In: Proceedings of the 33rd ACM International Conference on Multimedia. pp. 9608–9617 (2025)
34. Wang, H., Yin, J., Wei, Q., Zeng, W., Gu, L., Ye, S., Gao, Z., Wang, Y., Zhang, Y., Li, Y., Guo, Y., Wang, W., Chen, K., Qiao, Y., Zhang, H.: Internsvg: Towards unified svg tasks with multimodal large language models. In: The Fourteenth International Conference on Learning Representations (2026), https://openreview.net/forum?id=YxqnNNs3sf
35. Wang, W., Gao, Z., Gu, L., Pu, H., Cui, L., Wei, X., Liu, Z., Jing, L., Ye, S., Shao, J., et al.: Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. arXiv preprint arXiv:2508.18265 (2025)

36. Wu, R., Su, W., Ma, K., Liao, J.: Iconshop: Text-guided vector icon synthesis with autoregressive transformers. ACM Transactions on Graphics (TOG) **42**(6), 1–14 (2023)
37. Xing, X., Guan, Y., Zhang, J., Xu, D., Yu, Q.: Reason-svg: Hybrid reward rl for aha-moments in vector graphics generation. arXiv preprint arXiv:2505.24499 (2025)
38. Xing, X., Hu, J., Liang, G., Zhang, J., Xu, D., Yu, Q.: Empowering llms to understand and generate complex vector graphics. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 19487–19497 (2025)
39. Xing, X., Wang, C., Zhou, H., Zhang, J., Yu, Q., Xu, D.: Diffsketcher: Text guided vector sketch synthesis through latent diffusion models. Advances in Neural Information Processing Systems **36**, 15869–15889 (2023)
40. Xing, X., Yu, Q., Wang, C., Zhou, H., Zhang, J., Xu, D.: Svgdreamer++: Advancing editability and diversity in text-guided svg generation. IEEE transactions on pattern analysis and machine intelligence (2025)
41. Xing, X., Zhou, H., Wang, C., Zhang, J., Xu, D., Yu, Q.: Svgdreamer: Text guided svg generation with diffusion model. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4546–4555 (2024)
42. Yang, Y., Cheng, W., Chen, S., Zeng, X., Yin, F., Zhang, J., Wang, L., YU, G., Ma, X., Jiang, Y.G.: Omnisvg: A unified scalable vector graphics generation model. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems (2025)
43. Zhang, B., Zhang, P., Dong, X., Zang, Y., Wang, J.: Long-clip: Unlocking the long-text capability of clip. In: European conference on computer vision. pp. 310–325. Springer (2024)
44. Zhang, P., Zhao, N., Liao, J.: Text-to-vector generation with neural path representation. ACM Transactions on Graphics (TOG) **43**(4), 1–13 (2024)
45. Zhu, J., Wang, W., Chen, Z., Liu, Z., Ye, S., Gu, L., Tian, H., Duan, Y., Su, W., Shao, J., et al.: Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. arXiv preprint arXiv:2504.10479 (2025)
46. Zou, B., Cai, M., Zhang, J., Lee, Y.J.: Vgbench: A comprehensive benchmark of vector graphics understanding and generation for large language models. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. pp. 3647–3659 (2024)