VA-SVG: IMPROVING SVG CODE PREDICTION VIA GENERATIVE VISUAL ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing SVG generation methods typically adopt multimodal large language models (MLLMs) as their backbone and formulate SVG synthesis primarily as code sequence prediction. However, SVGs constitute a distinctive image–code multimodal representation, requiring the model to generate code that is visually consistent with the target image. This process demands not only a precise understanding of fine-grained visual attributes, such as layout, structure, and spatial relationships, but also accurate alignment among images, textual semantics, and SVG code. Without explicit visual guidance, conventional cross-entropy supervision over code sequences struggles to capture complex visual semantics, often producing SVGs with limited visual fidelity or poor geometric consistency. To address these limitations, we propose **VA-SVG, a visual alignment post-training framework that couples SVG code prediction with visual generation.** Specifically, we train an auxiliary diffusion model to map continuous SVG code representations into the visual domain, and align its intermediate representations with visual features extracted from rendered SVGs. This differentiable pathway transfers visual generative knowledge back to the code generation model without requiring gradients through the SVG renderer. Extensive experiments and ablation studies demonstrate the effectiveness of the proposed training paradigm.

1 INTRODUCTION

Scalable Vector Graphics (SVG) is an XML-based format widely used in graphic design due to its editability and compact storage. Recent SVG generation models (Rodriguez et al., 2025; Yang et al., 2026; Wang et al., 2026c) predominantly adopt VLMs or MLLMs to generate SVG code conditioned on text or images. Compared with natural language, SVG code sequences are substantially longer and follow highly regular syntactic structures, which poses significant challenges to the conventional next-token prediction paradigm (Rodriguez et al., 2025; Wang et al., 2026c). More importantly, small token-level errors can accumulate and cause deviations in the rendered image (Zhang et al., 2026; Xing et al., 2025b). Unlike standard MLLM pretraining, which mainly encourages models to understand the semantic content of text and images, SVG code prediction requires a different set of capabilities. For Image-to-SVG, the model must attend to structured visual information, including graphic primitives and component layouts, and translate them into valid SVG code with precise coordinates. For Text-to-SVG, the model must infer a spatial layout and visual composition from the textual description, and then synthesize the corresponding drawing program. Therefore, accurate SVG prediction relies on both *fine-grained semantic perception* and *precise alignment between visual appearance and structured code*.

To align SVG code with its visual rendering, a natural solution is to render the predicted SVG and directly impose visual consistency (Fig. 1 (c)). However, standard SVG renderers are non-differentiable, preventing gradients from being propagated back to the code generator. Reinforcement learning (RL) can circumvent this issue through rendering-based rewards (Wang et al., 2026a;b; 2025a), but such indirect supervision can be unstable due to the high sensitivity of SVG rendering to discrete code changes (Fig. 1 (d)). As summarized in Fig. 1, existing SVG generation paradigms either sacrifice vector editability, lack explicit visual supervision, or rely on non-differentiable rendering and indirect visual rewards. This motivates us to seek a differentiable visual supervision mechanism that improves code modeling while preserving standard autoregressive SVG inference.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

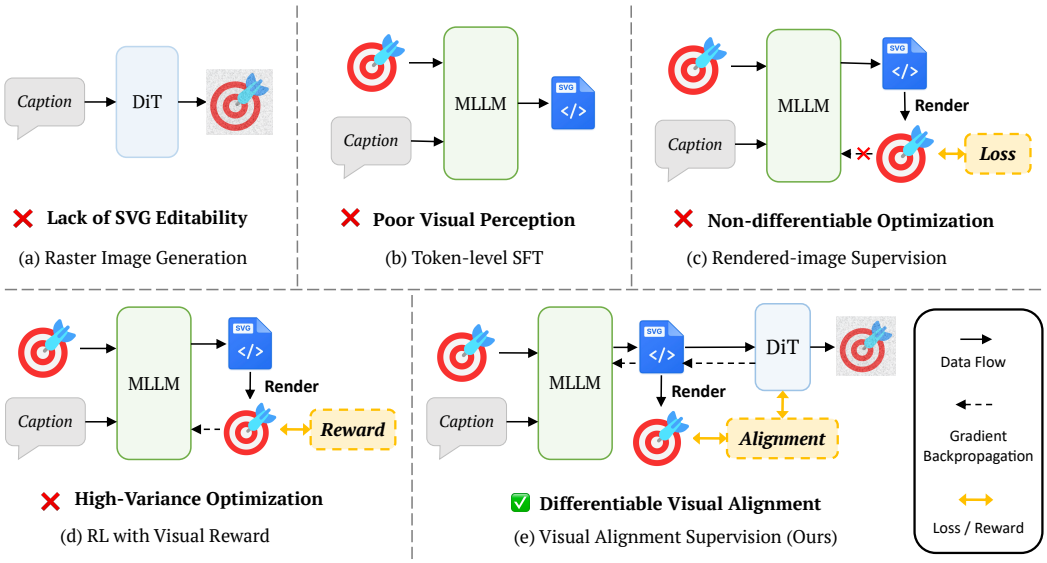


Figure 1: Comparison of SVG generation paradigms. (a) Raster image generation sacrifices vector editability. (b) Token-level SFT lacks explicit visual supervision. (c)&(d) Rendering-aware optimization is blocked by either non-differentiable rendering or unstable training. (e) Unlike the above methods, VA-SVG introduces a differentiable generative visual alignment pathway while preserving autoregressive SVG inference.

SVG naturally provides two complementary representations of the same content: structured code and its rendered image. This dual representation provides an opportunity to use visual generation to supplement token-level supervision for SVG code prediction. Motivated by this property, we propose **VA-SVG**, a visual-alignment post-training framework that couples an MLLM for SVG code prediction with a DiT-based visual generation branch, as illustrated in Fig. 1(e). Rather than using the DiT merely as an auxiliary image generator, VA-SVG turns its learned visual prior into differentiable supervision for the code generator. To achieve this, VA-SVG adopts a two-stage visual post-training strategy. In the first stage, building upon a pretrained SVG code generator, we train the DiT to reconstruct rasterized SVG images conditioned on MLLM code representations, establishing a differentiable mapping from code representations to visual appearance. In the second stage, we freeze the trained DiT and align its intermediate representations with visual features extracted from renderings of MLLM-predicted SVGs. The alignment objective then propagates gradients through the frozen DiT into the MLLM, allowing visual supervision to improve SVG code prediction without differentiating through the renderer.

Overall, the main contributions of VA-SVG can be summarized in three aspects. First, we introduce a two-stage post-training framework that first learns a mapping from SVG code representations to rasterized images and then transfers the resulting visual knowledge back to SVG code prediction. Second, we develop a visual alignment objective that propagates supervision through a frozen DiT to the MLLM, providing differentiable visual guidance without backpropagating through the SVG renderer. Third, experiments on MMSVG-Bench (Yang et al., 2026) demonstrate consistent gains over matched code-only training across both Text-to-SVG and Image-to-SVG tasks.

2 Related Work

2.1 SVG Generation

Existing SVG generation methods can be broadly categorized into optimization-based and learning-based approaches. Early methods optimize vector primitives with differentiable rasterizers or pretrained diffusion priors (Li et al., 2020; Ma et al., 2022; Jain et al., 2023; Xing et al., 2024), achieving high visual fidelity but requiring costly per-instance optimization. Recent works (Rodriguez et al., 2025; Xing et al., 2025b; Yang et al., 2026; Wu et al., 2023a; Wang et al., 2026c) instead formulate

SVG synthesis as autoregressive code generation with LLMs or MLLMs, directly predict structured SVG code from text or images. To improve complex SVG generation, SVGen (Wang et al., 2025a) and SVGThinker (Chen et al., 2025) introduce explicit chain-of-thought reasoning, whereas rendering-aware RL methods (Rodriguez et al., 2025; Xing et al., 2025a; Wang et al., 2026b) further incorporate sequence-level rendering reward or multi-task reasoning reward. More recently, visual guidance has been incorporated into SVG generation: DuetSVG (Zhang et al., 2026) jointly models image and SVG representations to provide internal visual guidance, while Render-in-the-Loop (Liang et al., 2026) feeds intermediate renderings back to the MLLM for iterative visual feedback. Nevertheless, existing approaches that explicitly incorporate visual feedback typically rely on costly optimization, indirect rewards, or additional visual feedback during generation. In contrast, VA-SVG transfers visual generative supervision during post-training while preserving standard autoregressive inference.

2.2 Generative Supervision for Multimodal Learning

Recent unified multimodal models (Pan et al., 2025; Wu et al., 2025; AI et al., 2025; Deng et al., 2025) have increasingly explored the synergy between visual understanding and generation. UniCorn (Fang et al., 2026) and SRUM (Jin et al., 2025) further employ the model's own understanding capability to evaluate or reward generated results, enabling self-improvement of the generation branch. These approaches mainly transfer knowledge from understanding to generation, whereas the reverse direction remains comparatively underexplored. A growing line of work has begun to investigate whether generative objectives can in turn improve multimodal representations. ROSS (Wang et al., 2025b) introduces image reconstruction as vision-centric supervision to preserve fine-grained visual details for multimodal comprehension, and GenHancer (Ma et al., 2025) exploits generative denoising objectives to enrich discriminative visual representations. RecA (Xie et al., 2026) further connects understanding and generation through self-supervised reconstruction alignment. More recently, UniMRG (Su et al., 2026b) explicitly improves multimodal learning by jointly generating complementary pixel, depth, and segmentation representations, while UniGame (Su et al., 2026a) turns the generation branch into an adversary that exposes fragile understanding representations during post-training. In contrast, VA-SVG focuses on structured SVG code prediction and uses a pretrained visual generative model as a differentiable supervisor, providing fine-grained geometric supervision and improve image-code alignment for SVG code modeling.

3 Method

3.1 Generative Visual Alignment for SVG Prediction

The goal of VA-SVG is to improve SVG code prediction by introducing explicit visual supervision into training. Since most recent learning-based SVG generators adopts MLLMs as their backbone, we utilize a modular MLLM+DiT architecture that decouples symbolic SVG prediction from visual generation, making the proposed post-training paradigm readily applicable to existing models.

Given a multimodal condition $\mathbf{x}$, which can be either a textual description or a reference image, the MLLM predicts an SVG program $\mathbf{s}$. The predicted program can be executed by a standard CairoSVG renderer $\mathcal{R}$ to obtain its rasterized visual output. In parallel, we introduce a Diffusion Transformer (DiT) (Peebles & Xie, 2023) as an auxiliary visual generation branch. Instead of directly consuming discrete SVG tokens, the DiT is conditioned on the continuous code representations produced by the MLLM and learns to reconstruct the corresponding rasterized SVG image. In this way, the two branches describe the same SVG instance from complementary perspectives: the MLLM models its symbolic structure and semantics, whereas the DiT models its visual appearance. This hierarchical design preserves the original autoregressive SVG generation pipeline while establishing a differentiable semantic-to-visual pathway for subsequent visual supervision.

SVG Code Generation Pretraining Following OmniSVG (Yang et al., 2026), in order to represent structured SVG elements, we augment the original MLLM tokenizer with an SVG-specific vocabulary and corresponding learnable embeddings. As shown in the green box of Fig. 2, given multimodal condition $\mathbf{x}$ and target SVG sequence $\mathbf{s}$, the MLLM predicts SVG tokens autoregressively through the LLM head and the MLLM is optimized using the standard next-token objective:

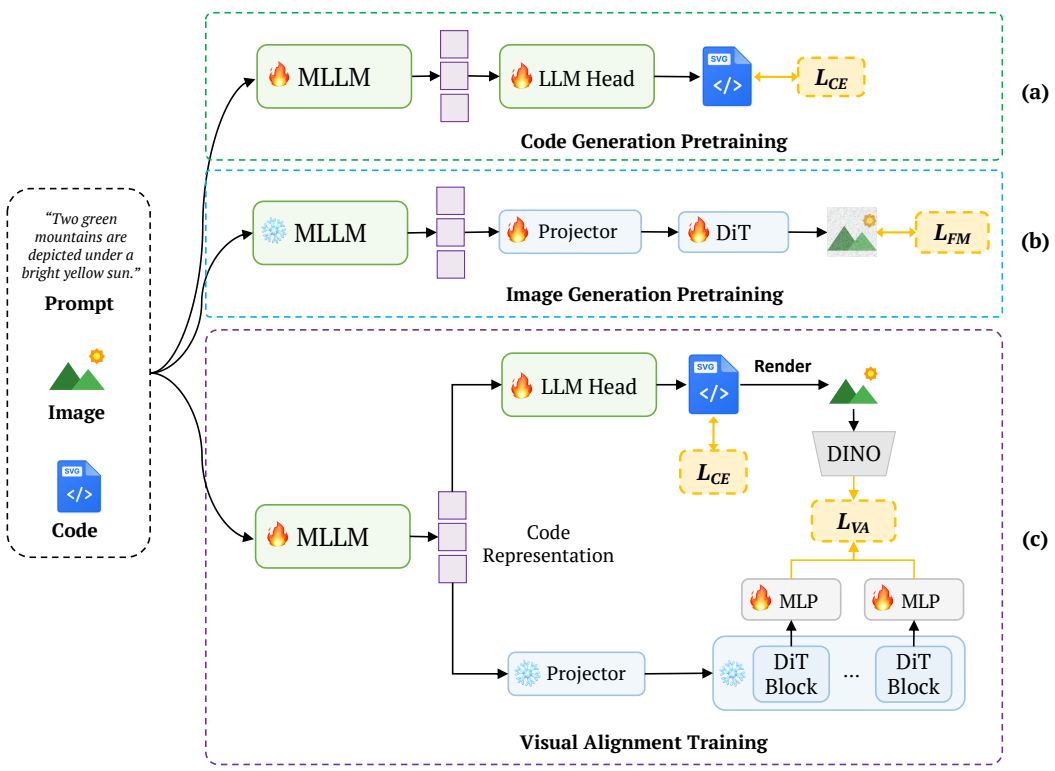


Figure 2: Overview of VA-SVG. Starting from our pretrained SVG code generator (Fig.2a), we train a DiT-based surrogate renderer to map MLLM code representations into the visual domain (Fig.2b). We then freeze the visual branch and align its intermediate representations with DINO features of rendered SVG predictions, allowing visual alignment gradients to update the MLLM (Fig.2c). Flames and snowflakes denote trainable and frozen modules, respectively.

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{L}\sum_{i=1}^{L} \log p_\theta\left(s_i \mid \mathbf{x}, \mathbf{s}_{<i}\right). \tag{1}$$

This equips the MLLM with autoregressive SVG code generation capability. Nevertheless, $\mathcal{L}_{\mathrm{CE}}$ only constrains predictions in the discrete token space and does not explicitly indicate whether the predicted geometry and layout correspond to the desired visual appearance, motivating the introduction of an additional visual generation branch.

SVG Image Generation Pretraining We next establish a differentiable mapping from the MLLM's SVG representations to the visual domain, as illustrated in the blue box of Fig. 2. Under teacher forcing with the target SVG sequence, we extract the final-layer hidden states corresponding to SVG tokens: $\mathbf{H}_{\mathrm{svg}} = \mathrm{Select}_{\mathrm{svg}}(\mathbf{H}) \in \mathbb{R}^{L\times d_{\mathrm{llm}}}$. Unlike discrete SVG tokens, these contextualized code representations encode the MLLM's predictions of SVG structure, geometry, and semantics under the given multimodal condition. We employ a lightweight learnable projector $\mathcal{P}_\phi$ to map them into the conditioning space of the DiT: $\mathbf{C}_{\mathrm{svg}} = \mathcal{P}_\phi\left(\mathbf{H}_{\mathrm{svg}}\right) \in \mathbb{R}^{L\times d_{\mathrm{dit}}}$. The projected sequence $\mathbf{C}_{\mathrm{svg}}$ serves as semantic conditioning for visual generation. Given target image latent $\mathbf{z}$, Gaussian noise $\boldsymbol{\epsilon}$, and $t \sim \mathcal{U}(0,1)$, we construct $\mathbf{z}_t = (1-t)\boldsymbol{\epsilon} + t\mathbf{z}$ and optimize the conditional flow-matching objective (Lipman et al., 2023):

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{\mathbf{z},\boldsymbol{\epsilon},t}\left[\left\|v_\psi\left(\mathbf{z}_t, t, \mathbf{C}_{\mathrm{svg}}\right) - (\mathbf{z}-\boldsymbol{\epsilon})\right\|_2^2\right]. \tag{2}$$

During this stage, we freeze the pretrained MLLM and optimize only the projector and DiT, so that the visual branch to learn a mapping from code representations to SVG appearance.

Visual Alignment Training To transfer visual generative knowledge to SVG code modeling, we introduce a visual alignment objective based on representations learned by the pretrained DiT. REPA (Yu et al., 2025) demonstrates that intermediate DiT representations encode meaningful visual semantics by aligning them with features from pretrained visual encoders. Inspired by this observation, we reverse the optimization direction: the DiT is frozen and used as a differentiable visual supervisor for the upstream MLLM.

As illustrated in the purple box of Fig. 2, given the SVG code $\hat{\mathbf{s}}$ predicted by the MLLM, we first obtain its rasterized image using the CairoSVG renderer $\mathcal{R}$ and extract its visual representation using a frozen DINO encoder (Oquab et al., 2023): $\mathbf{Y} = \mathcal{E}_{\text{DINO}}\left(\hat{\mathbf{I}}\right)$. Meanwhile, the MLLM code embeddings are projected into the frozen DiT, from which we extract intermediate features at multiple blocks:

$$\mathbf{F}_t^{(l)} = \text{DiT}_\psi^{(l)}\left(\mathbf{z}_t, t, \mathcal{P}_\phi(\mathbf{H}_{\text{svg}})\right), \qquad l \in \mathcal{S}. \tag{3}$$

Since DiT and DINO features lie in different representation spaces, a lightweight MLP $\mathcal{G}_l$ is introduced for each selected block. We then align the projected DiT representations with DINO features using cosine distance:

$$\mathcal{L}_{\text{align}} = \frac{1}{|\mathcal{S}|}\sum_{l\in\mathcal{S}}\left(1 - \cos\left(\mathcal{G}_l\left(\mathbf{F}_t^{(l)}\right), \mathbf{Y}\right)\right). \tag{4}$$

During visual alignment training, the MLLM, LLM head, and multiple MLPs are optimized, while the projector and DiT remain frozen. Since the renderer itself is non-differentiable, the DINO branch only provides the visual target, gradients from $\mathcal{L}_{\text{align}}$ are propagated through the frozen DiT and projector to the MLLM. Finally, we combine visual supervision with the original SVG code prediction objective:

$$\mathcal{L}_{\text{joint}} = \mathcal{L}_{\text{CE}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}}. \tag{5}$$

This joint objective preserves autoregressive code generation while introducing dense visual supervision that encourages more accurate modeling of geometry, layout, and visual appearance.

3.2 Why Does Visual Alignment Improve Code Prediction?

We analyze the underlying mechanism from two complementary perspectives. First, we examine whether the pretrained diffusion branch provides a sufficiently strong visual prior; second, we investigate whether transferring this prior facilitates SVG code optimization.

Diffusion provides a stronger visual prior. As shown in Fig. 3, the pretrained DiT generally produces visually cleaner and more coherent SVG images than the renderings obtained from the pretrained MLLM predictions. The advantage is particularly evident in fine-grained geometry and component structure, such as the leaf-body separation of the tomato, and the layout of interface elements. This qualitative comparison suggests that the diffusion branch has already learned a stronger visual prior than the code generation branch after training. Therefore, aligning MLLM representations with the pretrained DiT provides a meaningful optimization target rather than merely introducing an auxiliary objective.

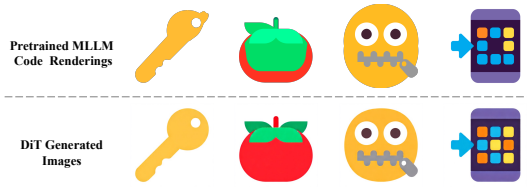
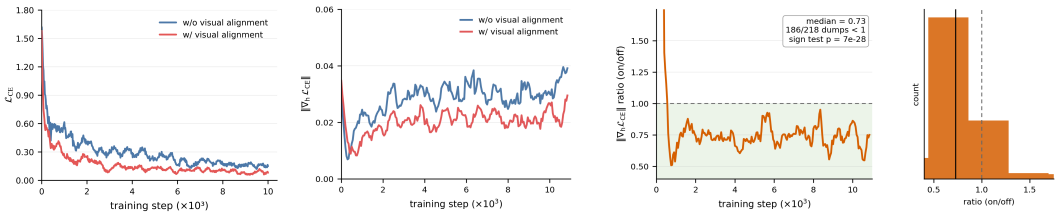


Figure 3: Comparison between pretrained MLLM code renderings and generated SVG images of DiT. The DiT produces more coherent geometry and finer visual details, indicating a stronger visual prior for subsequent alignment.

Visual alignment facilitates code optimization. Having established that the DiT provides a stronger visual prior, we next examine whether this information is effectively transferred to SVG code modeling. As shown in Fig. 4, under identical initialization, data, and optimization settings, visual aligned training consistently produces a faster decrease in $\mathcal{L}_{\text{CE}}$ and maintains a lower loss throughout training. We further measure the CE gradient norm at the SVG code representations



(a) CE Loss (b) CE Gradient Norm (c) Relative CE Gradient Norm

Figure 4: Visual alignment facilitates SVG code optimization, yielding consistently lower CE loss and CE gradient magnitude with statistically improvements across training.

H_{svg} shared by the code and visual branches. Visual alignment yields smaller CE gradients at 186 of 218 checkpoints, with a median paired ratio of 0.73 and a two-sided sign-test p-value of 7×10^{-28}. Together with the consistently lower CE loss, the reduced CE gradient magnitude is consistent with visual alignment reaching lower-error regions of the code prediction landscape earlier.

4 Experiments

4.1 Experimental Setup

Datasets and Benchmarks. For SVG code generation pretraining, we construct the training corpus by combining the Text-to-SVG and Image-to-SVG data from MMSVG (Yang et al., 2026) with SVGX-SFT-GEN-51k (Xing et al., 2025b), resulting in approximately 1.3M training examples. We further curate a high-quality subset of approximately 0.4M samples for image generation pretraining and visual alignment training. The same subset is used for both stages to maintain a consistent training distribution. For each SVG program, we render the corresponding raster image and use it as the target for visual generation and representation alignment. We evaluate VA-SVG on MMSVG-Bench (Yang et al., 2026), covering both Text-to-SVG and Image-to-SVG tasks on the Icon and Illustration subsets, following its standard evaluation protocol. We adopt complementary metrics to evaluate semantic consistency, perceptual quality, and reconstruction fidelity. For Text-to-SVG generation, we report FID (Heusel et al., 2017) for distribution-level visual quality and CLIP Score (Radford et al., 2021) for text-image alignment. We additionally use Aesthetic Score (Schuhmann, 2022) and HPS (Wu et al., 2023b) to assess perceptual quality and human preference. For Image-to-SVG reconstruction, we report DINO Score (Oquab et al., 2023) to measure high-level semantic similarity, together with SSIM (Wang et al., 2004), LPIPS (Zhang et al., 2018), and MSE to capture structural and perceptual reconstruction quality.

Baselines. We compare VA-SVG with representative optimization-based and autoregressive SVG generation methods. Optimization-based methods include VectorFusion (Jain et al., 2023), SVGDreamer (Xing et al., 2024), LIVE (Ma et al., 2022), and DiffVG (Li et al., 2020). Autoregressive methods include StarVector (Rodriguez et al., 2025), IconShop (Wu et al., 2023a), SVGen (Wang et al., 2025a), OmniSVG (Yang et al., 2026), InternSVG (Wang et al., 2026c), and Render-in-the-Loop (RITL) (Liang et al., 2026). For reference, we additionally report a Qwen3-VL model (Bai et al., 2025) trained with conventional SVG code SFT, and further including GPT-5 (Singh et al., 2025) as a strong closed-source model.

Implementation Details. We initialize the multimodal language model and diffusion branch from Bailing-V2 (AI et al., 2025) and Z-Image (Cai et al., 2025), respectively. The maximum SVG code length is set to 2,048 tokens. During visual alignment training, we use a frozen DINO-v2 encoder (Oquab et al., 2023) to provide visual supervision. To capture complementary visual semantics at different depths of the diffusion model, we extract intermediate representations from the 8th, 14th, and 20th blocks of the 30-layer DiT, selected based on preliminary experiments. Each alignment MLP consists of two linear layers followed by LayerNorm. We set $\lambda_{\text{align}} = 1.5$ and apply a 0.5 learning rate scale to the MLPs to prevent the projection heads from dominating alignment optimization. Both DiT pretraining and visual alignment training are performed for 3 epochs with a batch size of 6 and learning rate of 5×10^{-5} on 8 NVIDIA H800 GPUs.

Table 1: Quantitative comparison on Text-to-SVG and Image-to-SVG generation on MMSVG-Bench. VA-SVG achieves strong performance across both Icon and Illustration subsets. Best results are in bold, and second best are underlined.

Dataset	Methods	Text-to-SVG				Image-to-SVG			
		FID↓	CLIP↑	Aes↑	HPS↑	DINO↑	SSIM↑	LPIPS↓	MSE↓
Icon	*Optimization-based Methods*								
	VectorFusion	**250.77**	**0.240**	**4.76**	**0.237**	–	–	–	–
	SVGDreamer	308.94	0.207	4.26	0.221	–	–	–	–
	LIVE	–	–	–	–	0.932	0.943	0.106	0.011
	DiffVG	–	–	–	–	**0.940**	**0.954**	**0.066**	**0.002**
	Autoregressive Methods								
	StarVector-8B	–	–	–	–	0.895	0.881	0.231	0.059
	IconShop	213.28	0.288	4.55	0.244	–	–	–	–
	SVGen	129.23	0.281	4.78	0.244	–	–	–	–
	GPT-5	203.45	0.247	4.80	0.236	0.936	0.842	0.208	0.041
	OmniSVG-8B	130.56	0.276	4.60	0.242	0.922	0.893	0.235	0.040
	InternSVG-8B	128.80	0.291	4.75	0.246	0.926	0.901	0.182	0.034
	Qwen3-VL(SFT)	148.72	0.262	4.41	0.232	0.889	0.842	0.312	0.094
	RITL	**127.64**	**0.293**	4.86	0.251	0.931	0.914	0.172	0.027
	VA-SVG	129.54	**0.293**	**4.91**	**0.256**	**0.969**	**0.925**	**0.101**	**0.023**
Illustration	*Optimization-based Methods*								
	VectorFusion	**253.94**	0.185	**4.94**	**0.226**	–	–	–	–
	SVGDreamer	419.70	**0.201**	4.37	0.221	–	–	–	–
	LIVE	–	–	–	–	0.935	0.950	0.111	0.008
	DiffVG	–	–	–	–	**0.945**	**0.955**	**0.065**	**0.001**
	Autoregressive Methods								
	StarVector-8B	–	–	–	–	0.877	0.900	0.238	0.046
	IconShop	137.93	0.233	4.46	0.224	–	–	–	–
	SVGen	139.53	0.220	4.61	0.230	–	–	–	–
	GPT-5	186.79	0.212	4.64	0.232	0.949	0.898	0.150	0.027
	OmniSVG-8B	138.42	0.231	4.51	0.232	0.905	0.907	0.231	0.031
	InternSVG-8B	138.10	0.229	4.58	0.234	0.912	0.915	0.205	0.026
	Qwen3-VL(SFT)	155.84	0.221	4.33	0.221	0.882	0.861	0.318	0.082
	RITL	137.79	0.237	**4.73**	0.242	0.918	**0.928**	0.178	**0.021**
	VA-SVG	**134.67**	**0.239**	4.70	**0.245**	**0.953**	0.922	**0.132**	0.022

4.2 Main Results

Quantitative Comparison. Quantitative Comparison. Tab. 1 compares VA-SVG with optimization-based and autoregressive methods on MMSVG-Bench. For Text-to-SVG, VA-SVG achieves an Icon CLIP score of 0.293, matching RITL, while improving the aesthetic score from 4.86 to 4.91 and HPS from 0.251 to 0.256. On Illustration, it reduces FID from 137.79 to 134.67 and increases CLIP and HPS from 0.237 to 0.239 and from 0.242 to 0.245, respectively, although its aesthetic score is slightly lower. These results show consistent gains in text alignment and preference scores, with a modest improvement in FID.

The advantage is particularly evident for Image-to-SVG reconstruction. VA-SVG achieves the highest DINO scores on both Icon and Illustration, reaching 0.969 and 0.953, respectively. Compared with RITL, it reduces LPIPS from 0.172 to 0.101 on Icon and from 0.178 to 0.132 on Illustration, indicating better preservation of visual structure and details. It also lowers Icon MSE from 0.027 to 0.023, although RITL performs slightly better on Illustration MSE. Instance-specific optimization methods such as DiffVG and LIVE remain stronger on some pixel-level metrics, but optimize vector primitives separately for each target image. VA-SVG instead generates SVG code autoregressively and achieves these reconstruction gains without iterative rendering feedback at inference.

Qualitative Comparison. Fig. 5 presents visual comparisons with representative SVG generation methods. In Text-to-SVG generation, VA-SVG demonstrates stronger instruction following and structural fidelity. For example, it correctly preserves the specified number in the calendar example,

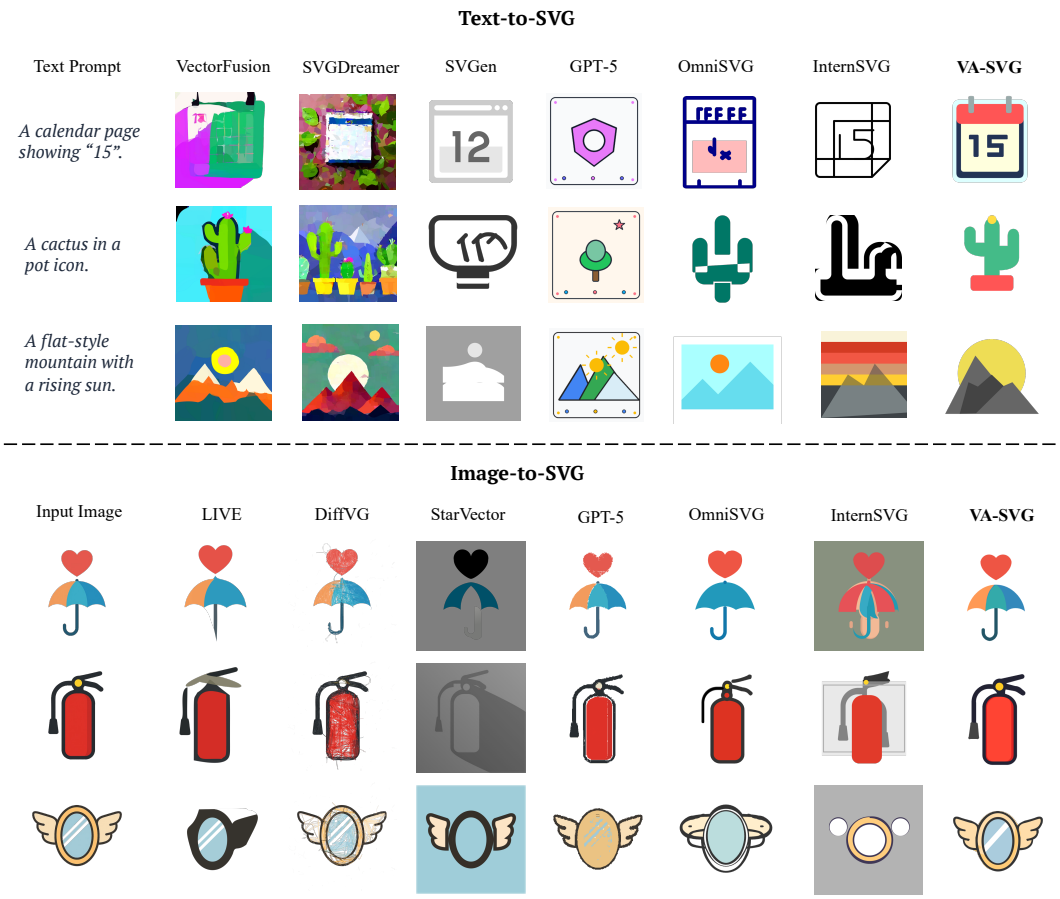


Figure 5: Qualitative comparison on Text-to-SVG and Image-to-SVG tasks. VA-SVG produces SVGs with stronger instruction fidelity, structural consistency, and preservation of fine-grained visual details.

produces a clear composition of cactus and pot, and generates a compact mountain scene with a coherent rising sun. In contrast, competing methods often capture the coarse concept while missing key attributes or introducing distorted structures.

For Image-to-SVG reconstruction, VA-SVG better preserves both global geometry and fine-grained components, including the colored umbrella panels, the contour and nozzle of the fire extinguisher, and the symmetric wings and internal structure of the mirror. Optimization-based methods can closely fit local pixels but often produce less compact or less semantically structured vector representations, whereas autoregressive baselines tend to simplify or distort important details and may introduce irrelevant background elements.

4.3 Ablation Study

We further conduct ablation studies to verify the effectiveness of the proposed visual alignment strategy. Quantitative results are reported in Tab. 2, and qualitative comparisons are illustrated in Fig. 6.

Effect of visual alignment. As shown in Tab. 2, RL with image consistency reward improves several Text-to-SVG metrics but provides limited or even degraded Image-to-SVG reconstruction, whereas visual-aligned SFT yields consistent improvements across both tasks. CE-based SFT already improves the pretrained model, whereas augmenting it with visual alignment further improves all evaluated metrics on both Text-to-SVG and Image-to-SVG. The gains are especially pronounced for reconstruction-oriented metrics, indicating that the additional visual supervision improves not

Table 2: Ablation studies on MMSVG-Bench Icon under different post-training strategies and visual encoders. Best results are in bold.

Factor	Setting	Text-to-SVG				Image-to-SVG			
		FID↓	CLIP↑	Aes↑	HPS↑	DINO↑	SSIM↑	LPIPS↓	MSE↓
Training Mode	Pretrain	145.20	0.269	4.70	0.244	0.902	0.880	0.228	0.049
	RL	**128.24**	0.290	4.73	0.240	0.905	0.865	0.229	0.062
	SFT (CE)	136.55	0.281	4.83	0.246	0.926	0.902	0.205	0.053
	SFT (CE+VA)	129.54	**0.293**	**4.91**	**0.256**	**0.969**	**0.925**	**0.101**	**0.023**
Vision Encoder	SigLIP	138.43	0.289	**4.91**	0.253	0.957	0.920	0.117	0.034
	DINO	**129.54**	**0.293**	**4.91**	**0.256**	**0.969**	**0.925**	**0.101**	**0.023**

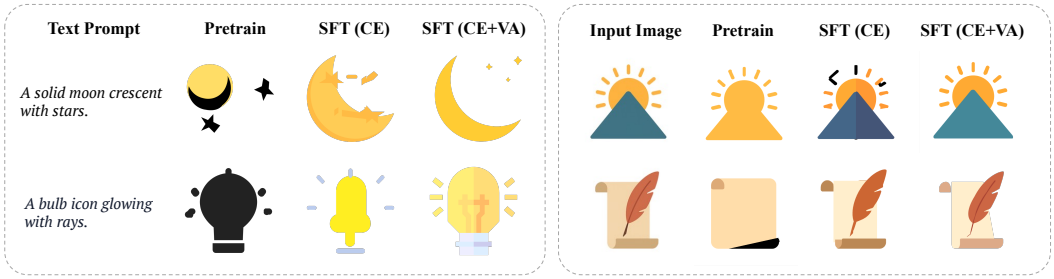


Figure 6: Qualitative ablation study. Visual alignment consistently improves structural fidelity and fine-grained visual details over alternative training strategies.

only semantic code modeling but also the fine-grained visual fidelity encoded by the predicted SVG. These results complete the optimization picture observed in Sec. 3.2: the stronger visual prior from the DiT facilitates MLLM training and ultimately leads to better SVG prediction. Fig. 6 provides qualitative comparisons of different training variants. Compared with the pretrained model and CE-only SFT, our full model produces SVGs with more accurate structures, cleaner geometry, and better preservation of fine-grained details.

Effect of the visual representation. We further investigate whether the source of the visual representation used for alignment affects the transfer of generative supervision. DINO yields overall stronger performance than SigLIP (Tschannen et al., 2025) across both tasks, suggesting that the effectiveness of visual alignment depends on the supervisory representation rather than merely the presence of an additional loss. The stronger performance of DINO further supports our premise that fine-grained visual representations are critical for transferring useful generative knowledge to SVG code modeling.

5 Conclusion

We present VA-SVG, a unified MLLM+DiT framework that improves SVG code prediction through generative visual alignment. Our key finding is that a pretrained visual generative model can provide a stronger visual prior than the code generation model itself, and that this knowledge can be transferred back to the MLLM through differentiable representation alignment. Our analysis shows that such visual supervision facilitates optimization of the code prediction, while the ablation studies demonstrate that this advantage translates into consistent improvements in SVG generation and reconstruction. These results provide evidence linking stronger generative priors and improved structured code prediction, suggesting generative visual alignment training as a promising post-training paradigm for multimodal code generation.

AI USE STATEMENT

In this work, we used generative AI tools to aid or polish writing. We have not used generative AI tools for retrieval and discovery, research ideation or execution, or drafting sections of the paper. Generating synthetic datasets and proving mathematical claims are not applicable to this work. We have reviewed all AI-assisted work. We checked the LLM-polished text for potential plagiarism through a manual literature survey. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Inclusion AI, Biao Gong, Cheng Zou, Chuanyang Zheng, Chunluan Zhou, Canxiang Yan, Chunxiang Jin, Chunjie Shen, Dandan Zheng, Fudong Wang, et al. Ming-omni: A unified multimodal model for perception and generation. *arXiv preprint arXiv:2506.09344*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Huanqia Cai, Sihan Cao, Ruoyi Du, Peng Gao, Aiming Hao, Steven Hoi, Zhaohui Hou, Shijie Huang, Dengyang Jiang, Yuming Jiang, et al. Z-image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025.

Hanqi Chen, Zhongyin Zhao, Ye Chen, Zhujin Liang, and Bingbing Ni. Svgthinker: Instruction-aligned and reasoning-driven text-to-svg generation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 11004–11012, 2025.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

Zhen Fang, Ruiyan Han, XinYu Sun, Yuchen Ma, Ziheng Wang, Yu Zeng, Zehui Chen, Lin Chen, Wenxuan Huang, Wei-Jie Xu, et al. Unicorn: Towards self-improving unified multimodal models through self-generated supervision. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13638–13669, 2026.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1911–1920. IEEE, 2023.

Weiyang Jin, Yuwei Niu, Jiaqi Liao, Chengqi Duan, Aoxue Li, Shenghua Gao, and Xihui Liu. Srum: Fine-grained self-rewarding for unified multimodal models. *arXiv preprint arXiv:2510.12784*, 2025.

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6): 1–15, 2020.

Guotao Liang, Zhangcheng Wang, Juncheng Hu, Haitao Zhou, Ziteng Xue, Jing Zhang, Dong Xu, and Qian Yu. Render-in-the-loop: Vector graphics generation via visual self-feedback. *arXiv preprint arXiv:2604.20730*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Shijie Ma, Yuying Ge, Teng Wang, Yuxin Guo, Yixiao Ge, and Ying Shan. Genhancer: Imperfect generative models are secretly strong vision-centric enhancers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 24402–24412. IEEE, 2025.

Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16314–16323. IEEE, 2022.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Xichen Pan, Satya Narayan Shukla, Aashu Singh, Zhuokai Zhao, Shlok Kumar Mishra, Jialiang Wang, Zhiyang Xu, Jiuhai Chen, Kunpeng Li, Felix Juefei-Xu, et al. Transfer between modalities with metaqueries. *arXiv preprint arXiv:2504.06256*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Juan A Rodriguez, Abhay Puri, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images and text. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16175–16186, 2025.

Christoph Schuhmann. Improved aesthetic predictor. `https://github.com/christophschuhmann/improved-aesthetic-predictor`, 2022. GitHub repository.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Zhaolong Su, Wang Lu, Hao Chen, Sharon Li, and Jindong Wang. Unigame: Turning a unified multimodal model into its own adversary. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 37632–37641, 2026a.

Zihan Su, Hongyang Wei, Kangrui Cen, Yong Wang, Guanhua Chen, Chun Yuan, and Xiangxiang Chu. Generation enhances understanding in unified multimodal models via multi-representation generation. *arXiv preprint arXiv:2601.21406*, 2026b.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Feiyu Wang, Zhiyuan Zhao, Yuandong Liu, Da Zhang, Junyu Gao, Hao Sun, and Xuelong Li. Svgen: Interpretable vector graphics generation with large language models. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 9608–9617, 2025a.

Feiyu Wang, Jiayuan Yang, Zhiyuan Zhao, Da Zhang, Bingyu Li, Peng Liu, and Junyu Gao. Introsvg: Learning from rendering feedback for text-to-svg generation via an introspective generator-critic framework. *arXiv preprint arXiv:2603.09312*, 2026a.

Haochen Wang, Anlin Zheng, Yucheng Zhao, Tiancai Wang, Zheng Ge, Xiangyu Zhang, and Zhaoxiang Zhang. Reconstructive visual instruction tuning. In *International Conference on Learning Representations*, volume 2025, pp. 14374–14399, 2025b.

Haomin Wang, Qi Wei, Qianli Ma, Shengyuan Ding, Jinhui Yin, Kai Chen, and Hongjie Zhang. Reliable reasoning in svg-llms via multi-task multi-reward reinforcement learning. *arXiv preprint arXiv:2603.16189*, 2026b.

Haomin Wang, Jinhui Yin, Qi Wei, Wenguang Zeng, Lixin Gu, Shenglong Ye, Zhangwei Gao, Yaohui Wang, Yanting Zhang, Yuanqi Li, et al. Internsvg: Towards unified svg tasks with multimodal large language models. In *International Conference on Learning Representations*, volume 2026, pp. 38649–38705, 2026c.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. Iconshop: Text-guided vector icon synthesis with autoregressive transformers. *ACM Transactions on Graphics (TOG)*, 42(6):1–14, 2023a.

Xiaoshi Wu, Keqiang Sun, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score: Better aligning text-to-image models with human preference. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2096–2105. IEEE, 2023b.

Ji Xie, Luke Zettlemoyer, Xudong Wang, et al. Reconstruction alignment improves unified multimodal models. In *International Conference on Learning Representations*, volume 2026, pp. 120095–120137, 2026.

Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. Svgdreamer: Text guided svg generation with diffusion model. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4546–4555. IEEE, 2024.

Ximing Xing, Yandong Guan, Jing Zhang, Dong Xu, and Qian Yu. Reason-svg: Hybrid reward rl for aha-moments in vector graphics generation. *arXiv preprint arXiv:2505.24499*, 4, 2025a.

Ximing Xing, Juncheng Hu, Guotao Liang, Jing Zhang, Dong Xu, and Qian Yu. Empowering llms to understand and generate complex vector graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19487–19497, 2025b.

Yiying Yang, Wei Cheng, Sijin Chen, Xianfang Zeng, Fukun Yin, Jiaxu Zhang, Liao Wang, Gang Yu, Xingjun Ma, and Yu-Gang Jiang. Omnisvg: A unified scalable vector graphics generation model. *Advances in Neural Information Processing Systems*, 38:113670–113696, 2026.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 87400–87442, 2025.

Peiying Zhang, Nanxuan Zhao, Matthew Fisher, Yiran Xu, Jing Liao, and Difan Liu. Duetsvg: Unified multimodal svg generation with internal visual guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10219–10229, 2026.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

A IMPLEMENTATION DETAILS OF VA-SVG

More Implementation Details. Our model is built upon BailingMM, which consists of a Qwen2.5-ViT-600M vision encoder (Wu et al., 2025) and a Bailing-MoE-v2-mini MLLM (AI et al., 2025), together with a frozen Z-Image DiT (Cai et al., 2025) as the visual generation branch. The last-layer hidden states of the MLLM at SVG code positions are projected into the DiT conditioning space using a RMSNorm+Linear projection head. We optimize the model with AdamW (Loshchilov & Hutter, 2017) using a peak learning rate of 1×10^{-4}, with a linear warmup for the first 500 steps followed by cosine decay to 1×10^{-6}. The global gradient norm is clipped to 1.0. Training is performed with BF16 automatic mixed precision, FlashAttention, and Fully Sharded Data Parallel (FSDP). The SVG vocabulary is augmented with 44,170 new tokens, and SVGs are represented on a 200×200 canvas. Images are processed at a resolution of 448 pixels, and the maximum text sequence length is set to 2048 tokens. During training, we use greedy decoding to generate SVG code, while during inference, unless otherwise specified, we use top-k and top-p sampling with $k = 50$ and $p = 0.95$ to ensure diversity.

Training Datasets. Our training corpus is constructed from three open-source SVG datasets: MMSVG-Icon, MMSVG-Illustration, and SVGX-SFT-Gen, as summarized in Tab. 3. For SVG code generation pretraining, we use a total of 1.67M SVG samples covering both Image-to-Code and Text-to-Code tasks. For image generation pretraining and subsequent visual alignment training, we further use 381K filtered SFT samples selected from these datasets. The resulting corpus covers a broad range of SVG structures, visual styles, and complexity levels, with substantial variation in SVG code length. Each sample provides SVG code together with its corresponding visual content, enabling the model to learn both accurate autoregressive code generation and effective visual alignment.

Table 3: Statistics of the training datasets.

Training Stage	Source	Total	Percentage	Token Length
Code Generation Pretrain	MMSVG-Icon	900k	53.9%	2.2k ± 0.9k
	MMSVG-Illustration	260k	15.6%	8.1k ± 3.3k
	SVGX-SFT-Gen	510k	30.5%	1.2k ± 0.5k
Image Generation Pretraining & Visual Alignment Training	MMSVG-Icon	358k	94.0%	1.5k ± 0.2k
	MMSVG-Illustration	18k	4.7%	3.8k ± 0.3k
	SVGX-SFT-Gen	5k	1.3%	0.9k ± 0.2k

B SFT DATA FILTERING

To ensure geometric fidelity and consistent styling in the SVG SFT corpus, we apply a rule-based streaming cleaner to the raw SVG data. Samples are first discarded if they (i) contain any `<path>` element without an explicit six-digit hexadecimal `fill` attribute, thereby excluding inherited, `currentColor`, `none`, or missing fills that introduce ambiguity into color supervision; (ii) exceed 2,048 tokens under our SVG tokenizer; (iii) exhibit coordinate quantization collisions, where two distinct original coordinates are mapped to the same cell on the 200×200 canvas and thus lose geometric information; or (iv) contain two or more consecutive identical path commands, which often indicate degenerate or duplicated geometry. The cleaner operates in a streaming fashion with 32 parallel processes and records rejection counts for each filtering rule. In a second stage, we enforce the trainability of every retained sample by requiring its SVG code to successfully round-trip through the project SVG tokenizer in strict mode. Specifically, a sample is retained only if its assistant response contains a complete `<svg>...</svg>` fragment that can be parsed and encoded by `SVGTokenizer.encode_svg_string(strict=True)` without error, produces a non-empty token sequence used during training. Samples that are malformed, contain unsupported SVG constructs, or exceed the length limit are removed, and the rejected samples together with a breakdown of rejection categories are saved for auditing.

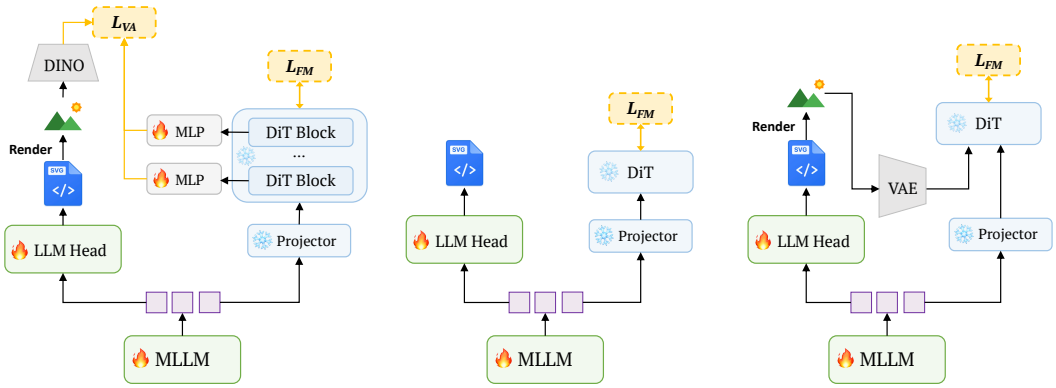
C ABLATION OF ALIGNED LAYERS

We further investigate how the choice of intermediate DiT layers affects visual alignment, as shown in 4. Aligning a single layer already provides strong performance, with layer 20 outperforming layer 12 on most reconstruction metrics, suggesting that intermediate visual representations at different depths provide complementary supervision. Using multiple well-separated layers generally yields better overall performance: the $\{8, 14, 20\}$ configuration achieves the most best scores among the evaluated settings, while obtaining a near-best HPS score. In contrast, the $\{8, 9, \ldots, 15\}$ configuration, which aligns a dense set of consecutive layers, performs consistently worse across both Text-to-SVG and Image-to-SVG metrics. These results suggest that visual supervision benefits from combining representations across different stages of the DiT, while excessive alignment on densely adjacent layers may provide less complementary information. Based on this observation, we use layers $\{8, 14, 20\}$ for visual alignment in the main experiments.

Table 4: Ablation study with different visual alignment layers. We present selected experimental results for different layer configurations.

Layer	**Text-to-SVG**				**Image-to-SVG**			
	FID↓	CLIP↑	Aes↑	HPS↑	DINO↑	SSIM↑	LPIPS↓	MSE↓
$\{12\}$	136.76	0.281	4.77	0.249	0.962	0.922	0.112	0.027
$\{20\}$	132.45	0.285	4.76	0.245	0.963	0.918	0.108	0.025
$\{5, 12, 20\}$	134.25	0.288	4.88	0.253	0.967	0.921	0.108	0.025
$\{8, 14, 20\}$	**129.54**	**0.293**	**4.91**	0.256	**0.969**	**0.925**	**0.101**	**0.023**
$\{8, 16, 24\}$	133.77	0.289	4.90	**0.257**	0.966	0.922	0.108	0.025
$\{8, 9, 10, 11, 12, 13, 14, 15\}$	136.20	0.285	4.83	0.248	0.960	0.917	0.119	0.028

D MORE VISUAL ALIGNMENT DESIGNS



(a) Additional Flow Matching Loss (b) No Visual Conditions (c) VAE Visual Condition

Figure 7: Additional visual alignment designs, including an extra flow-matching objective, removing visual conditions, and using VAE features as visual conditions. CE Loss is omitted for clarity.

In this section, we investigate several alternative designs for incorporating visual supervision, as illustrated in Fig. 6. Our default design aligns DINO features extracted from the rendered SVG with intermediate DiT features through trainable MLPs, while keeping the DiT and the projection layer frozen. In Fig. 6 (a), we additionally optimize the DiT with a flow-matching loss during visual alignment, providing an auxiliary supervision signal to regularize the intermediate DiT representations and mitigate potential representation degradation. As shown in Tab. 5, this modification does not improve the overall performance, suggesting that updating the visual generation objective is less beneficial than preserving the pretrained visual representation as a stable supervision target. In Fig. 6 (b), we remove the visual conditions from the DiT, i.e., the rendered SVG and its corresponding VAE representation are no longer provided to the visual branch. This leads to a substantial

Table 5: Experiment results with different visual alignment strategies.

Setting	Text-to-SVG				Image-to-SVG			
	FID↓	CLIP↑	Aes↑	HPS↑	DINO↑	SSIM↑	LPIPS↓	MSE↓
SFT (CE+VA)	**129.54**	**0.293**	**4.91**	**0.256**	**0.969**	0.925	**0.101**	0.023
SFT (CE+VA+FM)	134.41	0.292	4.90	0.253	0.966	**0.926**	0.107	**0.022**
No Visual Conditions	148.62	0.287	4.73	0.244	0.940	0.905	0.229	0.062
VAE Visual Condition	146.53	0.287	4.79	0.246	0.946	0.907	0.205	0.055

performance degradation across both Text-to-SVG and Image-to-SVG tasks, demonstrating that explicit visual conditioning is essential for establishing a meaningful connection between the predicted code and the visual representation. In Fig. 6 (c), we replace the rendered SVG condition with its VAE representation while retaining the visual alignment pathway. This variant partially recovers the performance, but remains inferior to our default design, indicating that directly using the rendered SVG provides more effective visual guidance than relying solely on compressed VAE features. Overall, these results demonstrate that both the visual condition and the choice of visual representation are important to the effectiveness of our alignment strategy, while keeping the pretrained DiT fixed provides a more reliable visual supervision signal.

E More visualization results

We present additional visualization results generated by VA-SVG in Fig. 8.

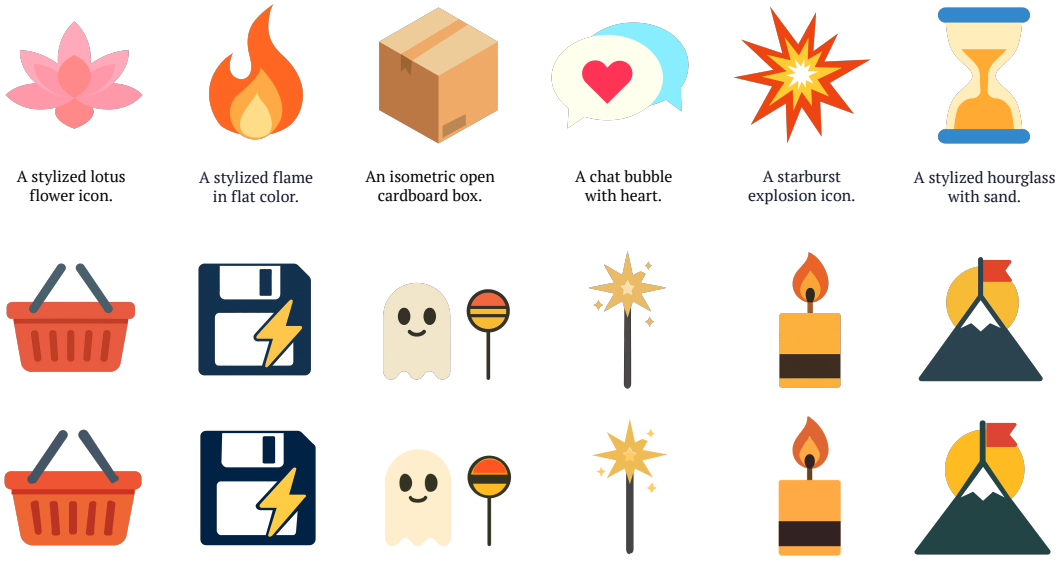


Figure 8: A gallery of diverse vector graphics generated by VA-SVG. The generated SVGs exhibit strong semantic consistency and visual quality across a wide range of subjects, while preserving clean and structured vector representations.