LLM-based Text-to-SVG Generation

Aylin Aydın
Department of Computer Engineering
Bogazici University
aylinaydin216@gmail.com

Abstract

Generating Scalable Vector Graphics (SVG) is challenging due to their structured, hierarchical nature. Existing approaches rely on large, paired datasets and expensive model fine-tuning, which limits accessibility and reproducibility. We investigate whether state-of-the-art large language models (LLMs) can generate high-quality SVGs through prompt engineering alone, without any training. We compare single-shot prompting against a multi-agent pipeline that decomposes generation into planning, synthesis, validation, and refinement stages. To assess quality, we combine structural code metrics with visual similarity measures using CLIP and BLIP. Our experiments show that multi-agent prompting substantially improves both structural correctness and visual fidelity across all models, offering a practical, zero-cost alternative to training-based methods.

1 Introduction

Scalable Vector Graphics (SVG) are crucial for modern digital content because they can scale without losing quality, remain editable, and compress efficiently. However, creating SVGs manually can be expensive, which is why we need professionals in the field. In today's agile coding environment, SVGs are vital for mobile and web applications, as well as websites. To generate them in large quantities from text descriptions, we can utilize large language models (LLMs). While LLMs excel at generating text and even raster images, consistently producing structured SVG code is still challenging. Unlike pixel arrays, SVGs require precise geometric paths, a proper XML hierarchy, and semantic coherence between visual elements and their descriptions.

Recent work addresses this through specialized training. LLM4SVG fine-tunes models on curated SVG datasets (1), SVGen builds a million-scale corpus with curriculum learning (2), and Chat2SVG combines language models with diffusion-based refinement (4). These methods advance the field but share a critical limitation: they require extensive datasets, costly training infrastructure, and complex optimization pipelines that hinder reproducibility.

We take a lightweight and accessible approach to SVG generation with large language models. Instead of training specialized models or relying on complex optimization pipelines, we ask a simpler question: how far can modern, off-the-shelf LLMs go with careful prompt design alone? We treat SVG generation as a structured process, similar to writing software, where breaking a complex task into smaller steps can lead to more reliable and interpretable results.

Based on this idea, we compare direct single-shot prompting with a multi-agent setup that divides SVG generation into complementary roles for planning, design, generation, validation, and refinement. We evaluate this approach across three leading LLMs—ChatGPT-5.1, Gemini-3-Pro, and Claude Sonnet-4.5—without any fine-tuning. To measure quality in a consistent and automated way, we combine simple structural checks on SVG code with visual-semantic similarity scores computed using CLIP and BLIP on rendered outputs.

2 Related Works

Recent advances in vector graphics generation have explored both specialized neural architectures and large language model adaptations, demonstrating the feasibility of automated SVG creation from natural language descriptions.

LLM4SVG (1) introduces domain-specific fine-tuning for SVG generation, training language models on large SVG corpora. Their key contribution is demonstrating that multi-stage processing—decomposing generation into planning and execution phases—significantly improves output quality. They report strong visual fidelity but require substantial computational resources (estimated 100 GPU hours) for training, with weights remaining proprietary.

OmniSVG (3) extends this paradigm with multi-modal understanding, incorporating visual reasoning capabilities to improve structural quality. Their architecture processes both textual descriptions and visual references, showing that cross-modal learning enhances geometric accuracy. However, the model requires extensive paired data and training infrastructure, with trained weights not publicly released.

SVGen (2) proposes an end-to-end neural synthesis approach, learning to map natural language directly to SVG primitives. Their work validates that learned representations can capture geometric relationships effectively. The training process requires substantial paired datasets (text-SVG pairs) and computational resources, with the complete training pipeline estimated at 200 GPU hours.

Chat2SVG (4) explores conversational interfaces for iterative SVG refinement. Their key insight is that human-in-the-loop feedback significantly improves generation quality—initial outputs are enhanced through multiple rounds of natural language corrections. While effective, their approach relies on fine-tuned models for optimal performance and requires user interaction for each refinement cycle.

StarVector (5) investigates compositional vector graphics generation, explicitly modeling spatial relationships and hierarchical structure. Their work demonstrates that treating SVG generation as hierarchical composition—rather than monolithic synthesis—improves both structural validity and visual quality. However, their method requires specialized training on compositionally annotated data.

While these approaches demonstrate impressive capabilities, several practical barriers emerge:

- **Accessibility:** Trained weights are typically proprietary or unreleased, limiting reproducibility and preventing researchers from building on prior work.
- **Computational Cost:** Fine-tuning requires substantial resources—GPU clusters, large curated datasets, and extended training time (100-200 GPU hours reported across methods).
- **Flexibility:** Specialized models trained on specific SVG distributions may not generalize well to novel styles or domains without retraining.
- **Reproducibility:** Closed implementations and proprietary architectures make it difficult for the research community to validate results or extend methods.

These limitations motivate alternative approaches that balance quality with accessibility, a gap we address in this work.

3 Method

3.1 SVG Generation Pipeline

We compare two prompting strategies across three frontier LLMs (ChatGPT-5.1, Gemini-3-Pro, Claude Sonnet-4.5):

Single-Shot: Direct generation from description. The prompt specifies output format (raw SVG, no markdown), quality requirements (gradients, flat/material design), and technical constraints (viewBox, 512×512 canvas, self-contained code).

Multi-Agent: Five-stage pipeline executed sequentially:

1. *Planner:* Decomposes input to structured JSON (elements, colors, layout)
2. *Designer:* Converts plan to exact geometry (coordinates, shapes, transforms)
3. *Generator:* Produces valid SVG with proper XML structure
4. *Validator:* Scores output 0-100; threshold $\geq$80
5. *Refiner:* Applies fixes once if score <80

We test two input formats: *name-column* (object name only) and *sentence* (descriptive phrase). This yields 9 configurations total (3 pipelines × 3 models).

3.2 Evaluation Metrics

- **CLIP** (6): Contrastive Language-Image Pre-training model encodes images into a shared embedding space where semantically similar images cluster together. We compute cosine similarity between CLIP-ViT-B/32 embeddings of generated and original images, measuring overall visual resemblance (0=completely different, 1=identical).
- **BLIP** (7): Bootstrapping Language-Image Pre-training generates natural language captions describing image content. We compute cosine similarity between BLIP caption embeddings, capturing semantic content alignment—whether generated SVGs depict the same objects and concepts as originals, even if stylistically different.

Together, CLIP captures low-level visual features (colors, shapes, composition) while BLIP captures high-level semantic content (objects, actions, attributes), providing complementary evaluation dimensions.

Code Quality: Automated scoring (0-1) across four dimensions:

- Validity (40%): XML parsing, namespace, balanced tags
- Structure (25%): viewBox, defs usage, semantic grouping
- Optimization (20%): File size <50KB, coordinate precision
- Readability (15%): Indentation, IDs, formatting

Combined Score: Weighted average:

$$\text{Overall} = 0.35 \times \text{CLIP} + 0.35 \times \text{BLIP} + 0.30 \times \text{Code} \quad (1)$$

3.3 Dataset and Implementation

Dataset: We sample 12 SVGs from the SVGX-SFT dataset (8), a large-scale collection of 250K high-quality vector graphics from diverse sources including Google (Noto Emoji), Twitter (Twemoji), and community repositories. Our sample spans multiple categories: emoji, icons, symbols, and illustrations. Each SVG includes the original vector code and pre-rendered PNG (512×512) for visual comparison.

Generation: Each of 12 samples is regenerated by all 9 configurations (3 pipelines × 3 models) = 108 total outputs. We provide two input formats: *name* (e.g., "face blowing a kiss") and *sentence* (e.g., "A smiling face with heart-shaped eyes emoji").

Implementation: Originals pre-rendered to PNG using CairoSVG for consistent evaluation. All prompts, evaluation code, and sample data open-sourced for reproducibility.

4 Experiments

Dataset: From SVGX-SFT's 250K SVGs (8), we select 12 diverse samples spanning emoji (Noto, Twemoji), icons, and symbols from multiple sources (Google, Twitter, community repositories).

Models: ChatGPT-5.1 (ChatGPT UI), Gemini-3-Pro (Google AI Studio), Claude Sonnet-4.5 (Claude UI).

Procedure: For each of 12 SVGs: (1) Extract name and description from SVGX-SFT metadata, (2) Generate via single-shot and both multi-agent variants across 3 models, (3) Render outputs to

PNG, (4) Compute CLIP/BLIP against original PNG from dataset, (5) Analyze code structure, (6) Aggregate scores.

Baseline: Single-shot prompting represents standard direct LLM code generation.

Table 1: Sample distribution from SVGX-SFT dataset

Source	Type	Count
Noto Emoji (Google)	Emoji, Symbols	4
Twemoji (Twitter)	Emoji, Icons	5
Community	Icons, Illustrations	3
Total	-	**12**

5 Results

5.1 Main Results

Table 2 shows aggregate performance. Multi-agent pipelines consistently outperform single-shot.

Table 2: Performance by pipeline and model (mean ± std across 12 SVGs)

Configuration	CLIP	BLIP	Code	Overall
By Pipeline (36 samples each)				
5-Agents-Sentence	**0.937**	**0.891**	0.940	**0.922**
5-Agents-NameColumn	0.922	0.879	**0.963**	0.919
Single-Shot	0.919	0.875	0.875	0.890
By Model (36 samples each)				
Gemini-3-Pro	0.930	**0.889**	**0.946**	**0.921**
ChatGPT-5.1	**0.932**	0.880	0.903	0.905
Sonnet-4.5	0.915	0.875	0.930	0.906

Key Findings:

Multi-agent superiority: 5-Agents-Sentence achieves 0.922 overall, +3.6% absolute over single-shot (0.890). Gains span all metrics: CLIP +2.0%, BLIP +1.8%, Code +7.4%.

Code quality boost: Multi-agent shows largest improvement in code quality (+6.5-8.8%). Structure improves from 0.875 (single-shot) to 0.940-0.963 (multi-agent), confirming that decomposition enhances XML validity and organization.

Model comparison: Gemini leads (0.921 overall) with highest visual similarity (0.930 CLIP, 0.889 BLIP) and code quality (0.946). ChatGPT and Claude perform comparably (0.905-0.906).

Input format: Sentence descriptions slightly outperform name-only (+0.3% overall), providing richer semantic context.

5.2 Per-Configuration Analysis

Best configuration: **5-Agents-Sentence + Gemini** achieves 0.931 overall (0.940 CLIP, 0.895 BLIP, 0.961 code).

Worst configuration: **Single-Shot + ChatGPT** scores 0.873 (0.933 CLIP, 0.869 BLIP, 0.807 code). Despite strong visual scores, code quality suffers without validation.

Figure 1 (center) shows configuration heatmap. All multi-agent combinations score > 0.90, while single-shot ranges 0.873-0.905.

5.3 Human Evaluation

To complement automated metrics, we conducted qualitative assessment of generated SVGs across all configurations. Two evaluators independently examined 36 outputs (12 samples × 3 models in multi-agent mode) and rated them on visual quality, semantic accuracy, and overall coherence.

Model-Specific Observations:

Gemini demonstrates superior visual reasoning capabilities. Element positioning is consistently accurate, with proper spatial relationships maintained across components. Color selection aligns well with semantic context for instance, generating warm tones for emoji expressions and appropriate corporate colors for brand icons. Gemini produces visually coherent outputs regardless of input verbosity; minimal descriptions (name only) yield results comparable to detailed sentence prompts. However, the multi-agent pipeline provides marginal improvement over single-shot generation, as Gemini's outputs already exhibit high structural validity.

ChatGPT shows strong conceptual understanding but struggles with component placement. While correctly identifying required elements (e.g., recognizing that a "smartwatch icon" needs a circular display, wristband, and UI elements), it frequently misaligns these components spatially. Borders overlap incorrectly, text appears outside intended boundaries, and layering order causes occlusion artifacts. Input verbosity has minimal impact—detailed descriptions do not significantly improve spatial accuracy. The multi-agent pipeline's validation and refinement stages fail to address these geometric errors effectively, as the Validator agent focuses on XML structure rather than spatial coherence.

Claude Sonnet exhibits over-elaboration in single-shot mode, adding unnecessary details that deviate from the intended simplicity. For icon generation tasks requiring minimalist design, Sonnet introduces decorative elements, gradients, and embellishments that increase visual complexity beyond requirements. It struggles to distill concepts into essential geometric primitives. However, the multi-agent pipeline dramatically improves Sonnet's outputs—the Planning Agent enforces constraint specification, and the Validation Agent penalizes excessive complexity. Multi-agent Sonnet produces cleaner, more focused designs that human evaluators rated as "noticeably improved" and "approaching professional quality." Despite this improvement, Sonnet outputs remain less visually satisfying than Gemini's, often appearing "technically correct but aesthetically flat."

6 Discussion

Why Decomposition Works: Results confirm that SVG generation benefits from multi-stage processing. The planning phase enforces hierarchical thinking, designer ensures geometric validity, and validator catches errors before output. This mirrors compilation (parsing $\rightarrow$ IR $\rightarrow$ codegen $\rightarrow$ optimization) rather than end-to-end translation.

Model Strengths: Gemini's lead (0.921) reflects strong visual reasoning (0.930 CLIP) and code quality (0.946). ChatGPT and Claude balance both dimensions comparably. These differences may stem from training data or architectural choices.

Cost-Quality Tradeoff: Multi-agent adds latency (5 sequential API calls vs 1) and cost (5× tokens). For our 108 samples, single-shot used $\sim$50K tokens vs $\sim$240K for multi-agent. However, the 3.6% quality gain may justify this for production use where correctness is critical.

Automated Evaluation: CLIP/BLIP scores correlate with human perception of visual fidelity, validating vision-language models as reliable quality proxies. Code metrics capture structural correctness orthogonal to visual similarity.

Limitations: Our 12-sample dataset limits statistical power. Larger benchmarks would strengthen conclusions. Code metrics are heuristic—human evaluation would complement automation. We focus on static SVGs; animated graphics pose additional challenges.

Accessibility: Single-shot achieves 96.5% of best multi-agent performance (0.890 vs 0.922), suggesting reasonable quality even without decomposition. However, multi-agent's +3.6% demonstrates headroom when quality matters.

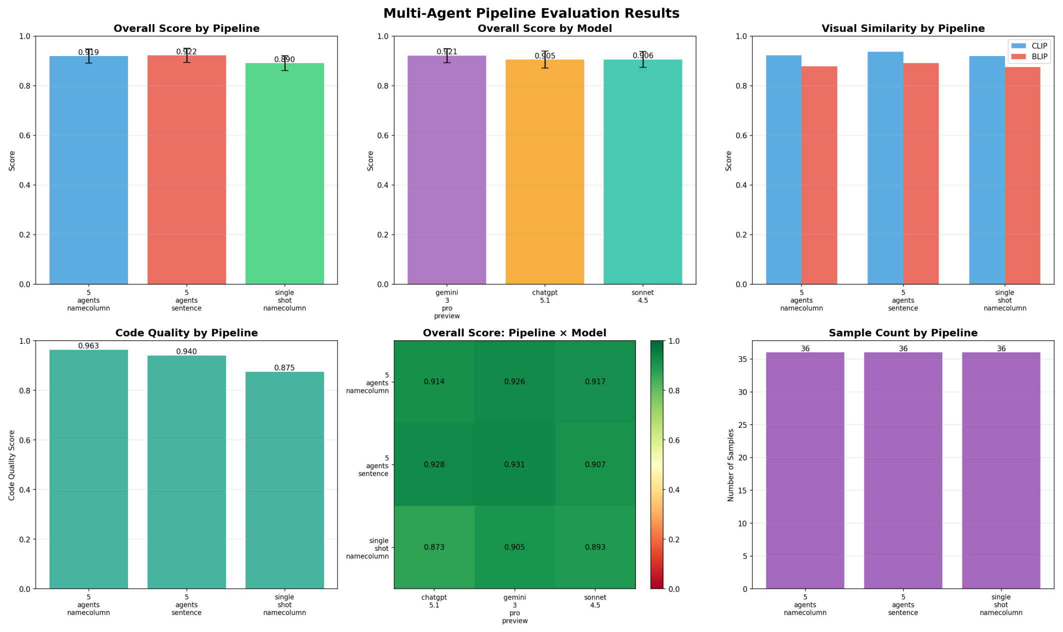

Figure 1: Multi-agent pipeline evaluation results. **Top**: Overall and visual similarity scores by pipeline (left, right) and model (center). **Bottom**: Code quality by pipeline (left), configuration heatmap (center), and sample counts (right). Multi-agent pipelines consistently outperform single-shot across all metrics.

7 Conclusion

We investigated whether frontier LLMs can generate quality SVGs through prompt engineering alone. Across ChatGPT, Gemini, and Claude, multi-agent decomposition consistently improves structural correctness (+7.4% code quality) and visual fidelity (+2.0% CLIP). Our best configuration (5-Agents-Sentence + Gemini) achieves 0.922 overall without training.

These results show that carefully designed prompt pipelines approach specialized model performance with full reproducibility and zero training cost. For structured code generation, decomposing into planning, synthesis, validation, and refinement provides a practical alternative to dataset collection and fine-tuning.

Future work should explore: (1) larger-scale evaluation, (2) human quality assessment, (3) animated/interactive SVGs, (4) adaptive architectures that adjust complexity per input. We release all prompts, code, and results for reproducibility.

References

[1] Xing, X., Hu, J., Liang, G., Zhang, J., Xu, D. & Yu, Q. Empowering LLMs to Understand and Generate Complex Vector Graphics. (2025), https://arxiv.org/abs/2412.11102

[2] Wang, F., Zhao, Z., Liu, Y., Zhang, D., Gao, J., Sun, H. & Li, X. SVGen: Interpretable Vector Graphics Generation with Large Language Models. *Proceedings Of The 33rd ACM International Conference On Multimedia*. pp. 9608-9617 (2025,10), http://dx.doi.org/10.1145/3746027.3755011

[3] Yang, Y., Cheng, W., Chen, S., Zeng, X., Yin, F., Zhang, J., Wang, L., Yu, G., Ma, X. & Jiang, Y. OmniSVG: A Unified Scalable Vector Graphics Generation Model. (2025), https://arxiv.org/abs/2504.06263

[4] Wu, R., Su, W. & Liao, J. Chat2SVG: Vector Graphics Generation with Large Language Models and Image Diffusion Models. (2025), https://arxiv.org/abs/2411.16602

[5] Rodriguez, J., Puri, A., Agarwal, S., Laradji, I., Rodriguez, P., Rajeswar, S., Vazquez, D., Pal, C. & Pedersoli, M. StarVector: Generating Scalable Vector Graphics Code from Images and Text. (2025), https://arxiv.org/abs/2312.11556

[6] Radford, A., Kim, J., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G. & Sutskever, I. Learning Transferable Visual Models From Natural Language Supervision. (2021), https://arxiv.org/abs/2103.00020

[7] Li, J., Li, D., Xiong, C. & Hoi, S. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. (2022), https://arxiv.org/abs/2201.12086

[8] Ming, X., Li, Y., Chen, H., et al. Xing, X., Hu, J., Zhang, J., Xu, D. & Yu, Q. SVGFusion: Scalable Text-to-SVG Generation via Vector Space Diffusion. *ArXiv Preprint ArXiv:2412.10437*. (2024)

A Generated SVG Examples

Figure 2 shows generated SVGs across all 9 configurations for test samples. Each row: original (left), 5-Agents-NameColumn (Gemini/ChatGPT/Sonnet), 5-Agents-Sentence (Gemini/ChatGPT/Sonnet), Single-Shot (Gemini/ChatGPT/Sonnet).

Key Observations:

Gemini maintains consistent quality across modes—samples achieve >0.92 scores with minimal variation between multi-agent and single-shot.

ChatGPT shows spatial placement issues in complex scenes—samples demonstrate persistent geometric errors across both modes.

Sonnet improves dramatically with multi-agent—samples show reduced over-elaboration and cleaner composition compared to single-shot.

By Complexity: Simple icons achieve 0.90-0.95 across all models. Complex scenes show more variation at 0.85-0.91, reflecting hierarchical structure challenges.

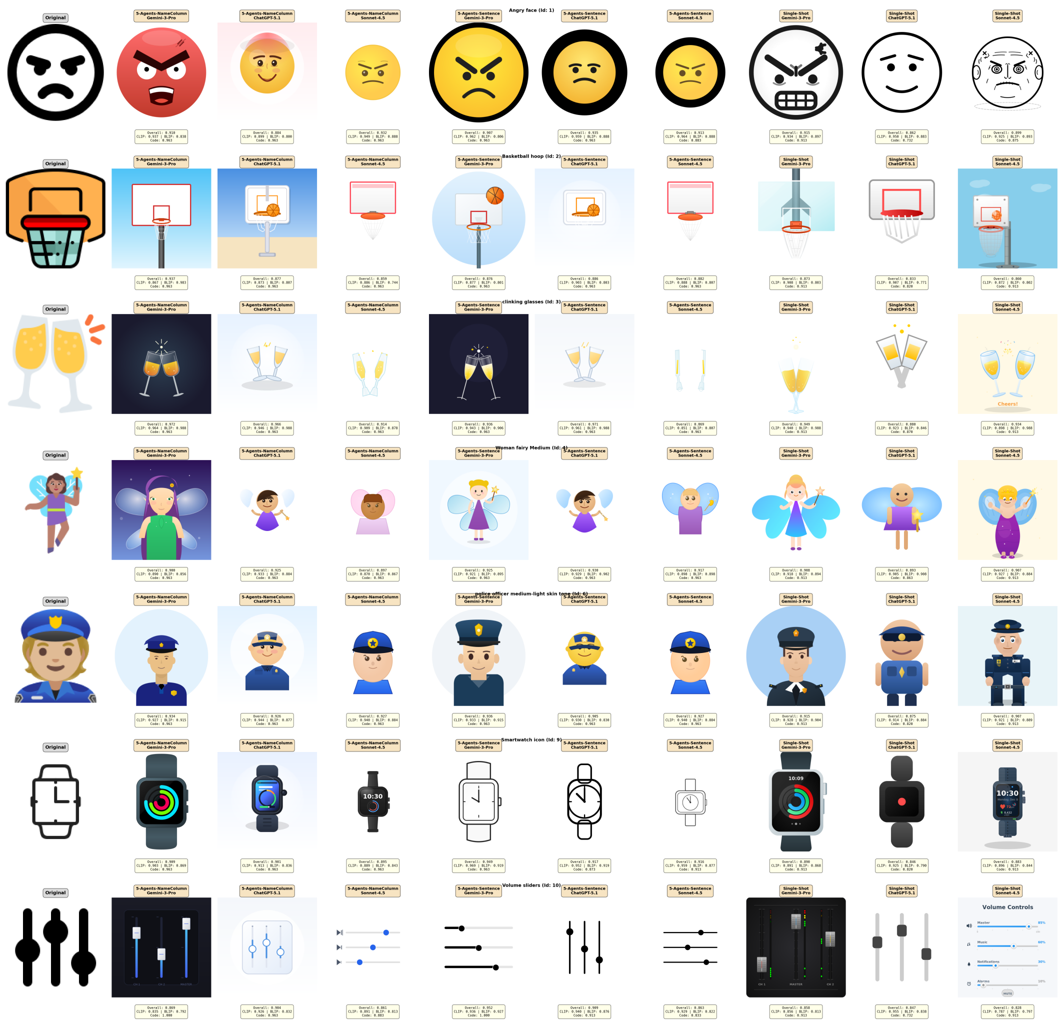
Angry face (Id: 1)
Basketball hoop (Id: 2)
clinking glasses (Id: 3)
Woman fairy Medium (Id: 4)
police officer medium-light skin tone (Id: 6)
Smartwatch icon (Id: 9)
Volume sliders (Id: 10)
Original
5-Agents-NameColumn Gemini-3-Pro
5-Agents-NameColumn ChatGPT-5.1
5-Agents-NameColumn Sonnet-4.5
5-Agents-Sentence Gemini-3-Pro
5-Agents-Sentence ChatGPT-5.1
5-Agents-Sentence Sonnet-4.5
Single-Shot Gemini-3-Pro
Single-Shot ChatGPT-5.1
Single-Shot Sonnet-4.5

Figure 2: Generated SVG examples