VECTORGYM: A MULTI-TASK BENCHMARK FOR SVG CODE GENERATION AND MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce VectorGym, a new comprehensive multi-task benchmark for evaluating Vision-Language Models (VLMs) on Scalable Vector Graphics (SVG) code generation and manipulation. VectorGym addresses the critical need for systematic evaluation across diverse SVG-related capabilities in the emerging field of visual code generation. Our benchmark comprises four complementary tasks: Sketch2SVG conversion, SVG editing with natural language instructions, Text2SVG generation, and SVG captioning. It introduces Sketch2SVG and the first dataset of complex, human-authored SVG edits, with gold-standard human annotations across all tasks. We propose a novel automatic VLM-as-judge evaluation metric specifically tailored for SVG generation tasks, validated through human correlation studies across multiple state-of-the-art models. We provide a comprehensive evaluation of leading closed-source and open-source VLMs, which reveals significant performance variations across tasks, highlighting both current capabilities and critical limitations. VectorGym establishes a new standard for evaluating and advancing SVG generation capabilities, offering the research community a robust framework for measuring progress in this emerging field.

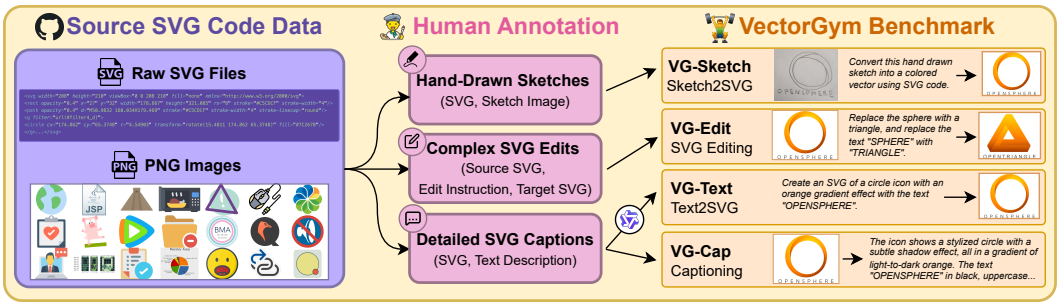


Figure 1: **Overview of VectorGym Benchmark.** VectorGym is a suite of human-annotated datasets covering Sketch2SVG (**VG-Sketch**), SVG Editing (**VG-Edit**), Text2SVG (**VG-Text**), and SVG Captioning (**VG-Cap**). Unlike prior benchmarks, it is built from diverse real-world SVGs sourced from GitHub. Human experts annotate each SVG by hand-drawing sketches, creating complex edits, and writing detailed text descriptions, which are further cleaned and adapted into instruction-style prompts at varying levels of detail. We evaluate state-of-the-art models in VectorGym.

1 INTRODUCTION

Scalable Vector Graphics (SVG) (Ferraiolo et al., 2000; Quint, 2003) are widely used across the web, design tooling, and digital media. Unlike raster images (Rodriguez et al., 2023a;b; Rombach et al., 2021), SVGs are programs: their code exposes geometry, style, and structure, enabling precise editing, scalable rendering, and semantic manipulation. Evaluating models on SVG therefore requires not only visual understanding but also reliable, syntax-aware code generation.

Despite rapid progress in Vision-Language Models (VLMs), existing evaluations of SVG generation remain limited. Prior datasets often target icons or basic shapes, rely on synthetic programmatic

Table 1: Compact comparison of key datasets. "Human" denotes fully human-authored eval supervision. "Test" reflects public splits when stated. See App. A for an extended table.

Dataset	Total Samples	Tasks	Complex Human Annotations
SVG-Bench (Rodriguez et al., 2025b)	10k	Image, Text	✗
VGBench (Xia et al., 2024)	9.5k	Image, Text, SVG Understanding	✗
SVGEditBench (Shu et al., 2025)	1.6k	Edit	✗
UniSVG (Li et al., 2025)	525k	Image, SVG Understanding	✗
VectorGym (ours)	7k	Sketch, Edit, Text, Captioning	✓

edits, rarely assess sketch-conditioned generation nor provide human gold labels (Rodriguez et al., 2025b; Wu et al., 2023; Zhang et al., 2023; Xing et al., 2025; Yang et al., 2025; Rodriguez et al., 2025a). As a result, the field lacks a unified, realistic benchmark that stresses, visual understanding, vector generation and structured SVG code manipulation.

We introduce **VectorGym**, a new comprehensive multi-task benchmark for SVG generation and manipulation spanning four tasks: (1) **Sketch2SVG**, converting rough sketches to clean vector code; (2) **SVG Editing**, applying natural-language edits to existing SVGs; (3) **Text2SVG**, generating SVGs from text; and (4) **SVG Captioning**, describing SVG content. VectorGym introduces Sketch2SVG and releases the first dataset of complex, human-authored SVG edits; all tasks use gold-standard human annotations.

Our benchmark covers in-the-wild diversity—icons, diagrams, emojis, fonts, logotypes, and complex illustrations—sourced from SVG-Stack (Rodriguez et al., 2025b). We pair this with careful human curation to ensure realistic task difficulty. We evaluate leading proprietary and open-source VLMs, providing a clear view of current capabilities and gaps.

Our main contributions are:

1. We introduce a comprehensive multi-task benchmark for real-world SVG code generation with gold-standard human annotations across all tasks;
2. We introduce the Sketch2SVG (VG-Sketch) task and release the first dataset of complex, human-authored SVG edits (VG-Edit);
3. We introduce a novel VLM-as-judge evaluation metric tailored to sketch, text, and editing tasks, validated through human correlation studies;
4. We provide extensive evaluation and analysis of current VLMs across diverse SVG generation scenarios.

2 Related Work

Vector Graphics Generation. Classical vectorization and shape-fitting methods (Li et al., 2020; Reddy et al., 2021) struggle with complex semantics; recent neural methods add structure via hierarchical programs and transformers (Carlier et al., 2020; Cao et al., 2023; Lopes et al., 2019), with sketch abstraction (Vinker et al., 2022) and text-conditioned icon synthesis (Wu et al., 2023).

VLMs for SVG Generation. Modern VLMs now produce syntactic code from visual inputs; StarVector (Rodriguez et al., 2025b) frames SVG creation as language–to–code, motivating benchmarks that jointly test visual understanding and program generation. Subsequent work further values this approach (Zhang et al., 2023; Cai et al., 2023; Yang et al., 2025).

SVG Datasets and Benchmarks. Foundational corpora include DeepSVG icons (Carlier et al., 2020), FIGR-8 (Clouâtre & Demers, 2019), and SVG–Stack (Rodriguez et al., 2025b). Closest to our scope, UniSVG (Li et al., 2025) unifies 525k items for understanding and generation; VGBench (Xia et al., 2024) aggregates multiple sources to evaluate image–to–SVG, text–to–SVG, and diagram code; and SVGEditBench (Shu et al., 2025) targets instruction-based editing. In contrast, **VectorGym** offers a unified *multi-task* evaluation spanning Sketch2SVG, SVG editing, Text2SVG, and captioning, with *human-authored* supervision across all tasks and realistic, high-difficulty edits. See figure 1 for a dataset comparison, and refer to Appendix A for more details.

3 VECTORGYM

VectorGym consists of four complementary tasks that comprehensively evaluate different aspects of SVG understanding and generation. Each task is designed to assess specific capabilities while contributing to a holistic understanding of visual2code generation performance.

3.1 TASK DEFINITIONS

Sketch2SVG Generation (VG-Sketch). This task evaluates the ability to convert rough, hand-drawn sketches into clean SVG code. Given a bitmap sketch image with approximate shapes and imperfect lines, models must generate SVG code that captures the essential geometric structure while producing a clean, scalable vector representation. This task tests spatial reasoning, shape recognition, and the ability to abstract from noisy visual input to structured geometric primitives.

SVG Editing (VG-Edit). In this task, models are given an SVG along with a natural language editing instruction and must produce a new SVG with the specified edit applied. The challenge lies in correctly parsing the intent, identifying the relevant elements, and applying the transformation while preserving code validity, visual coherence, and the integrity of unmodified parts. Since instructions and targets were created by skilled human annotators, the edits are non-trivial—for example, adding new objects, modifying logo content or text, converting a pie chart to a bar chart, or changing facial expressions. This task evaluates both SVG structure understanding and the ability to follow complex editing instructions. Figure 3 shows examples from our test set. Unlike prior benchmarks Shu et al. (2025), which focus on simple synthetic programmatic edits, *VG-Edit introduces realistic, high-difficulty editing scenarios.*

Text2SVG Generation (VG-Text). Given natural language descriptions of visual content, models must generate complete SVG code that accurately represents the described objects, scenes, or abstract concepts. Descriptions range from simple geometric shapes ("red circle with blue border") to complex illustrations ("minimalist icon of a house with a tree"). This task tests creative generation capabilities and the ability to translate semantic concepts into precise geometric representations.

SVG Captioning (VG-Cap). The inverse of text2SVG generation, this task requires models to analyze existing SVG code and generate natural language descriptions that accurately capture the visual content, style, and key characteristics. High-quality captions should describe both the semantic content ("house icon") and relevant visual properties ("minimalist style," "blue and white color scheme"). This task evaluates SVG code comprehension and visual understanding.

3.2 DATASET CONSTRUCTION

Our datasets are built upon carefully curated SVG collections, ensuring diversity across content types, complexity levels, and visual styles. We source high-quality SVGs from the SVG-Stack dataset (Rodriguez et al., 2025b) and established SVG collections, encompassing icons, diagrams, emojis, fonts, logotypes, and complex illustrations. Our curation process builds upon insights from existing SVG datasets including DeepSVG (Carlier et al., 2020) and FIGR-8-SVG variants (Clouâtre & Demers, 2019).

Human Annotation Process. We partnered with two specialized data annotation vendors to ensure high-quality annotations across all tasks. We extracted 7,000 high-quality samples from SVG-Stack through multi-stage filtering: visual quality assessment, token length filtering (2k-8k tokens for meaningful complexity), color entropy thresholding, and random sampling. From these, 300 samples were reserved for testing. Our annotation process employed over 20 annotators with diverse demographics and specialized skills in design, vector graphics, and coding. See Appendix A.1 for comprehensive details on the annotation methodology, quality assurance procedures, and complexity requirements.

4 EXPERIMENTS

We conduct comprehensive evaluation across all four VectorGym tasks using state-of-the-art VLMs. Our experimental setup is designed to provide fair comparison while highlighting the unique challenges of SVG code generation.

4.1 BASELINE MODELS

We conduct a comprehensive evaluation using all available state-of-the-art VLMs that support code generation capabilities. Our baseline selection follows a systematic approach to ensure comprehensive coverage of the current landscape.

Closed-Source Models. We evaluate leading commercial VLMs that demonstrate strong performance on visual understanding and code generation tasks: GPT-4O, GPT-5, CLAUDE-3.5, CLAUDE-3.7, CLAUDE-4, GEMINI-2.5-PRO, and GEMINI-2.5-FLASH. These models represent the current state-of-the-art in multimodal understanding and have shown exceptional capabilities in various visual2text and code generation benchmarks.

Open-Source Models. To ensure comprehensive coverage and reproducible research, we include leading open-source alternatives: QWEN-2.5-VL-3B, QWEN-2.5-VL-7B, QWEN-2.5-VL-32B, QWEN-2.5-VL-72B, GLM-4.1-VL-9B, and GLM-4.5-VL-106B-A12B. We made best efforts to identify and include all available VLM models with public code implementations that could be executed on our tasks.

Inference. All models are evaluated using their standard inference configurations with temperature settings optimized for code generation tasks. Given the stochastic nature of VLMs, we employ a best-of-n sampling strategy with n=5 generations per input, selecting the best sample based on the target metric for each task using a VLM-as-judge approach. This methodology ensures that we capture the models' peak performance capabilities while maintaining evaluation consistency.

4.2 EVALUATION METRICS

SVG Validity. We first assess whether generated SVG code is syntactically correct and can be successfully parsed by standard SVG parsers. This fundamental metric ensures that outputs are usable in real-world applications.

Visual Similarity. For tasks involving visual reproduction (Sketch2SVG, Text2SVG), we measure visual similarity between generated and target SVGs using perceptual metrics including structural similarity index (SSIM) and learned perceptual image patch similarity (LPIPS) applied to rendered images.

Semantic Accuracy. We evaluate whether generated content captures the intended semantic meaning through both automated metrics (using CLIP-based similarity) and human evaluation protocols. For editing tasks, we verify that instructions are correctly interpreted and applied.

Code Quality. Beyond correctness, we assess code quality through metrics including element efficiency (minimizing unnecessary elements), readability, and adherence to SVG best practices. These metrics evaluate whether models generate maintainable and efficient code.

Human Evaluation. A subset of outputs undergoes human evaluation by expert annotators who assess overall quality, semantic correctness, and task-specific criteria. Human evaluation provides crucial insights into aspects that automated metrics may miss.

SVG Captioning Metrics. For captioning, we report corpus BLEU (0–100, higher better), chrF++ (0–100, higher better), ROUGE-L F1 (0–100, higher better), BERTScore F1 (0–100, higher better), BGE-M3 cosine similarity (0–100, higher better), and an LLM-based rubric score (GPT-5; 0–100, higher better via 0–5 mapped to 0–100). All metrics are computed pairwise over aligned (reference, prediction) captions and averaged across the corpus.

4.2.1 VLM-AS-JUDGE EVALUATION METRIC

Traditional evaluation metrics for SVG generation (typically based on image reconstruction or text–image alignment) often fall short in capturing the nuanced visual and semantic qualities that determine the success of generated vector graphics. Existing work lacks comprehensive evaluation frameworks tailored to SVG generation, particularly metrics that can jointly assess visual fidelity and semantic alignment in vector code outputs (Xia et al., 2024; Shu et al., 2025; Li et al., 2025).

To address this gap, we introduce a novel *VLM-as-judge* evaluation metric designed specifically for the four SVG generation tasks presented in this paper. In particular, prior approaches proved insuf-

ficient for tasks such as sketch editing, where no reliable metric existed. Our method leverages the multimodal reasoning capabilities of state-of-the-art VLMs to produce holistic quality assessment scores (ranging from 0 to 10) for generated SVGs. In this section, we design four tailored VLM-judge metrics, one for each of our benchmark tasks, enabling a more faithful evaluation of both structural accuracy and semantic alignment.

Metric Development Process. We develop task-specific evaluation prompts designed to guide VLMs in assessing different aspects of SVG generation quality. For each of the four main generation tasks, we craft specialized prompts that encourage models to evaluate: (1) visual accuracy and fidelity; (2) semantic alignment with input requirements; (3) code quality and efficiency; and (4) overall aesthetic appeal.

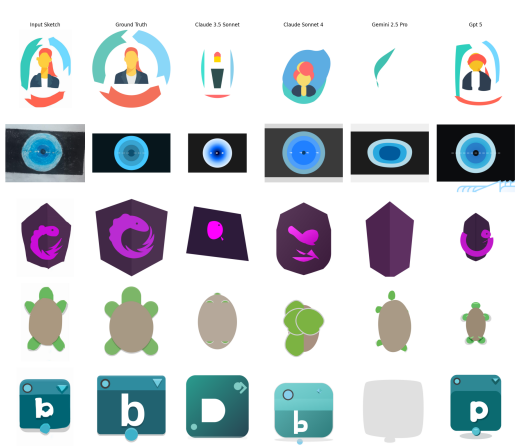
Judge Model Selection. To identify the most reliable VLM judge, we conduct a systematic comparison across four state-of-the-art models: GPT-4o, GPT-4.5, Gemini-2.5-Flash, and Claude-3.5-Sonnet. We evaluate each model's performance as a judge using a held-out validation set of 50 samples per task, comprising 25 ground truth examples and 25 model-generated outputs representing varying quality levels.

Human Correlation Validation. We validate our VLM-as-judge metric through extensive human evaluation studies. Expert human annotators independently evaluate the same 50 samples using identical scoring criteria (0-10 scale). We compute Pearson correlation coefficients between human judgments and each candidate VLM judge to identify the model that best aligns with human perception of SVG generation quality.

Final Judge Selection. Based on the correlation analysis in Table 2, we select the VLM with the highest agreement with human evaluators as our primary judge. This choice ensures that automated evaluation aligns closely with human perception of SVG quality, enabling a reliable and scalable assessment at large scale. Concretely, we adopt **GPT-4o** as the primary VLM-as-judge, as it achieves consistently high correlations across all tasks for both generated and ground-truth outputs. While GPT-5 and Gemini also perform well, their higher latency and cost make them less practical. Thus, GPT-4o offers the best balance of accuracy and efficiency.

Table 2: Pearson correlation between VLM-as-judge scores and human ratings across SVG generation tasks. The left block reports correlations on *Generated Samples*, and the right block on *Ground Truth Samples* (GT). Columns correspond to different VLM judges (GPT-4o, Claude 3.5 Sonnet, GPT-5, and Gemini 2.5 Flash), while rows indicate the model being evaluated. Overall, GPT-4o shows the strongest and most consistent alignment with human judgments across both generated and ground-truth samples, making it a good and cost-effective choice as primary judge. The GPT-4o columns are highlighted in orange to indicate our selection.

		Generated Samples				**Ground Truth Samples**			
Task	**Generator**	**GPT-4o**	**Claude**	**GPT-5**	**Gemini**	**GPT-4o**	**Claude**	**GPT-5**	**Gemini**
Text2SVG									
	Claude-4-Sonnet	0.695	0.050	0.401	**0.794**	**0.592**	0.255	0.334	0.410
	Gemini 2.5 Flash	**0.823**	0.638	0.748	0.770	**0.655**	0.350	0.344	0.477
	GPT-4o	0.695	0.050	0.401	**0.794**	**0.592**	0.255	0.334	0.410
Sketch2SVG									
	Claude-4-Sonnet	**0.312**	0.097	-0.237	0.289	**0.382**	-0.137	0.126	-0.079
	Gemini 2.5 Flash	0.452	0.381	0.316	**0.590**	-0.171	**0.156**	-0.101	0.020
	GPT-4o	**0.328**	0.293	0.036	0.062	-0.343	0.043	**0.170**	0.006
SVG Edit									
	Claude-4-Sonnet	**0.815**	0.729	0.684	0.814	-0.049	0.664	0.594	**0.666**
	Gemini 2.5 Flash	0.789	0.657	0.857	**0.869**	0.000	0.740	0.624	**0.773**
	GPT-4o	**0.815**	0.729	0.684	0.814	-0.049	0.664	0.594	**0.666**
SVG Captioning									
	Claude-4-Sonnet	0.812	0.603	0.714	0.820	0.678	0.441	0.512	0.662
	Gemini 2.5 Flash	0.835	0.620	0.709	**0.846**	0.701	0.452	0.521	**0.745**
	GPT-4o	**0.848**	0.639	0.720	0.842	**0.732**	0.468	0.544	0.726



(a) **VG-Sketch Results.** The left panel shows the colored sketch given to the model, followed by the ground-truth image and outputs from top-performing models. GPT-5 achieves the best results.

(b) **VG-Edit Results.** From left to right: the edit instruction, the input vector, the ground-truth edited vector, and outputs from top-performing models. Both GPT-5 and Claude perform well, with Claude showing stronger results overall.

Experimental Setup We evaluate all baseline models on our test sets without task-specific training. This establishes the inherent SVG generation capabilities of current VLMs. We limit our evaluation to inference and we do not train any models on our training dataset. We do this due to limited GPU resources for training.

5 Results

We present comprehensive evaluation results across all four VectorGym tasks. Our results demonstrate significant variation in model performance across different SVG generation and manipulation capabilities, revealing both strengths and limitations of current VLMs.

5.1 Text2SVG Generation

Table 3 presents our Text2SVG generation results, revealing clear performance hierarchies and interesting patterns. Among proprietary models, **GPT-5 achieves the highest VLM Judge score of 8.95**, followed closely by Claude-4 (8.81) and Claude-3.7 (8.79). This establishes GPT-5 as the current state-of-the-art for text-to-SVG generation. Notably, the top proprietary models cluster within a narrow range (8.79-8.95), suggesting we may be approaching performance saturation for this task with current architectures.

The open-source landscape shows **GLM-4.5 leading with 7.84**, demonstrating that open-source models can achieve competitive performance, though still trailing the best proprietary models by approximately 1.1 points. Interestingly, GLM-4.5 also achieves the highest CLIP score (25.07) among all models, indicating strong visual-semantic alignment despite lower judge scores.

Speed-performance trade-offs are particularly evident: Gemini-2.5-Flash achieves the fastest inference (2.87s) while maintaining reasonable quality (8.22), making it attractive for production scenarios. Conversely, GPT-5's superior performance comes at the cost of significantly longer inference times (63.44s).

5.2 SVG Editing

SVG Editing represents one of the most challenging tasks in our benchmark, requiring models to understand natural language instructions, parse existing SVG structures, and apply precise modifications. Table 4 reveals that **GPT-5 leads with a VLM Judge score of 7.92**, followed by Claude-4 (7.68) and GPT-4o (7.43). Notably, the performance gap between top proprietary and open-source models is smallest in this task, with GLM-4.5 achieving 7.57—only 0.35 points behind GPT-5.

Table 3: **Text2SVG and SVG Captioning Performance.** Higher values are better for CLIP Score, VLM Judge (Text2SVG), BLEU, CHRF++, BGE-M3, ROUGE, and LLM-as-Judge. Lower values are better for Time.

	Text2SVG			SVG Captioning				
Model	**VLM J ↑**	**CLIP ↑**	**Time (s) ↓**	**BLEU ↑**	**CHRF++ ↑**	**BGE-M3 ↑**	**ROUGE ↑**	**LLM J ↑**
Open-source Models								
QWEN-2.5-VL-7B	4.02	24.51	**10.54**	3.84	24.46	63.03	16.33	22.87
QWEN-2.5-VL-32B	6.87	24.66	22.00	1.94	18.96	57.77	8.42	21.91
QWEN-2.5-VL-72B	6.95	24.62	12.54	5.37	**31.24**	67.03	20.04	33.58
GLM-4.1-VL-9B	6.74	25.03	30.70	**6.55**	26.47	68.70	20.55	39.78
GLM-4.5-VL-106B-A12B	**7.84**	**25.07**	39.27	5.32	28.10	**69.99**	**21.19**	**45.60**
Proprietary Models								
GPT-4O	8.00	24.45	3.77	4.7243	30.67	68.04	20.23	40.82
GPT-5	**8.95**	24.41	63.44	3.21	24.90	69.72	21.40	**53.65**
CLAUDE-3.5	8.48	24.41	10.12	**6.36**	**31.22**	71.03	21.25	47.37
CLAUDE-3.7	8.79	24.33	11.03	4.93	30.56	69.89	20.77	45.73
CLAUDE-4	8.81	24.42	7.04	5.32	31.05	70.70	21.29	46.82
GEMINI-2.5-PRO	7.98	24.51	47.55	5.64	27.78	**71.13**	**23.96**	49.83
GEMINI-2.5-FLASH	8.22	**24.67**	**2.87**	4.90	28.49	68.46	21.38	39.32

This task shows the most **promise for open-source competitiveness**. GLM-4.1 demonstrates particularly strong performance across visual similarity metrics, achieving the best MSE (0.10) and DINO scores (0.89) while maintaining competitive LPIPS (0.26). This suggests that open-source models may be developing sophisticated understanding of SVG structure modification.

Interestingly, **Claude models show mixed performance**: while Claude-4 ranks second overall, Claude-3.5 and Claude-3.7 perform surprisingly poorly (6.49 and 6.51 respectively), despite excelling in other tasks. This inconsistency suggests that SVG editing requires specific capabilities that aren't uniformly developed across model versions.

The task also reveals clear **speed advantages for certain models**: Gemini-2.5-Flash achieves the fastest inference (2.77s) while maintaining reasonable quality (6.43), making it practical for interactive editing applications.

Table 4: **SVG Editing Performance.** Higher values are better for VLM Judge, SSIM, and DINO. Lower values are better for MSE, LPIPS, and Time.

Model	**VLM Judge ↑**	**MSE ↓**	**SSIM ↑**	**DINO ↑**	**LPIPS ↓**	**Time (s) ↓**
Open-source Models						
QWEN-2.5-VL-7B	6.94	0.14	0.70	0.85	0.32	**22.04**
QWEN-2.5-VL-32B	7.01	0.12	0.73	0.87	0.29	55.95
QWEN-2.5-VL-72B	7.18	0.11	0.73	0.87	0.28	74.23
GLM-4.1-VL-9B	7.07	**0.10**	0.76	**0.89**	**0.26**	30.12
GLM-4.5-VL-106B-A12B	**7.57**	0.11	**0.77**	0.86	**0.26**	49.03
Proprietary Models						
GPT-4O	7.43	0.09	0.78	0.91	0.23	24.85
CLAUDE-3.5	6.49	0.08	0.79	0.94	0.20	10.89
CLAUDE-3.7	6.51	0.09	0.78	0.91	0.23	13.83
GEMINI-2.5-PRO	7.04	0.13	0.74	0.79	0.30	61.75
GEMINI-2.5-FLASH	6.43	0.10	0.76	0.89	0.26	**2.77**
CLAUDE-4	7.68	0.07	**0.83**	**0.96**	0.17	27.68
GPT-5	**7.92**	**0.06**	**0.83**	**0.96**	**0.16**	79.42

5.3 SKETCH2SVG

Sketch2SVG emerges as the most challenging task in our benchmark, requiring models to interpret rough hand-drawn sketches and translate them into clean, scalable vector representations. Table 5 shows that even the best-performing model, **GPT-5, achieves only 8.01**—the lowest top score across all tasks, indicating the inherent difficulty of sketch-to-vector conversion.

The task reveals a **clear performance hierarchy**: GPT-5 (8.01) ¿ Claude-4 (7.30) ¿ GPT-4o (6.75), with substantial gaps between tiers. Notably, GLM-4.5 leads open-source models with 7.00, demonstrating competitive performance despite the task's complexity.

Visual similarity metrics provide additional insights: GPT-5 and Claude-4 achieve the best DINO scores (0.84 and 0.82 respectively), indicating superior semantic understanding of sketch content. However, GLM-4.1 and GLM-4.5 achieve the lowest MSE scores (0.18), suggesting they excel at pixel-level reconstruction even if semantic understanding lags.

The task shows interesting **speed-performance relationships**: while Gemini-2.5-Flash maintains its speed advantage (2.68s), the performance penalty is more pronounced here (6.62 vs 8.01 for GPT-5). This suggests that sketch interpretation may require more computational resources than other SVG generation tasks.

Table 5: Sketch2SVG Performance. Higher values are better for VLM Judge, SSIM, and DINO. Lower values are better for MSE, LPIPS, and Time.

Model	VLM Judge ↑	MSE ↓	SSIM ↑	DINO ↑	LPIPS ↓	Time (s) ↓
Open-source Models						
QWEN-2.5-VL-7B	4.50	0.19	0.61	0.74	0.46	**16.94**
QWEN-2.5-VL-32B	5.49	0.19	0.58	0.78	0.45	29.28
GLM-4.1-VL-9B	5.63	**0.18**	**0.65**	0.74	0.44	37.03
QWEN-2.5-VL-72B	6.21	0.20	0.58	0.77	0.44	17.63
GLM-4.5-VL-106B-A12B	**7.00**	**0.18**	0.62	**0.79**	**0.43**	49.24
Proprietary Models						
GEMINI-2.5-FLASH	6.62	0.16	0.64	0.82	**0.42**	**2.68**
GPT-4O	6.75	0.16	0.65	0.79	**0.42**	7.61
CLAUDE-3.5	6.95	0.16	**0.66**	0.79	**0.42**	3.04
GEMINI-2.5-PRO	6.97	0.17	0.64	0.79	**0.42**	73.69
CLAUDE-4	7.30	**0.15**	0.63	0.82	0.43	10.41
GPT-5	**8.01**	0.17	0.62	**0.84**	0.43	59.24

5.4 CROSS-TASK ANALYSIS AND KEY INSIGHTS

Our comprehensive evaluation across Text2SVG, SVG Editing, and Sketch2SVG reveals several critical insights about current VLM capabilities in vector graphics generation.

Task Difficulty Ranking. The results establish a clear difficulty hierarchy: **Text2SVG** (easiest, GPT-5: 8.95) > **SVG Editing** (intermediate, GPT-5: 7.92) > **Sketch2SVG** (hardest, GPT-5: 8.01). This ranking aligns with intuitive expectations: text descriptions provide explicit semantic guidance, editing requires understanding existing structures, while sketches demand interpretation of imprecise visual input.

Model Consistency Across Tasks. GPT-5 demonstrates remarkable consistency, leading in all three tasks with scores of 8.95, 7.92, and 8.01. Claude-4 consistently ranks second (8.81, 7.68, 7.30), while GPT-4o maintains third place (8.00, 7.43, 6.75). This consistency suggests these models have developed robust, generalizable SVG generation capabilities rather than task-specific optimizations.

Open-Source Competitiveness Varies by Task. The proprietary-open source gap is most pronounced in Text2SVG (1.11 points) and Sketch2SVG (1.01 points), but narrows significantly in SVG Editing (0.35 points). This suggests that **SVG editing may be the most promising area for open-source models to achieve parity** with proprietary alternatives.

Speed-Performance Trade-offs Are Task-Dependent. While Gemini-2.5-Flash consistently offers the fastest inference (2.68-2.87s), the performance penalty varies: minimal in Text2SVG (8.22 vs 8.95), moderate in Sketch2SVG (6.62 vs 8.01), and more substantial in SVG Editing (6.43 vs 7.92). This suggests editing tasks may be less amenable to speed optimizations.

5.5 SVG CAPTIONING

The SVG Captioning results in Table 3 reveal interesting patterns distinct from the generation tasks.

GPT-5 dominates the LLM Judge metric (53.65), significantly outperforming other models, which aligns with its strong generation capabilities. However, the traditional NLP metrics show different rankings: Claude-3.5 leads in BLEU (6.36) and chrF++ (31.22), while Gemini-2.5-Pro achieves the highest BGE-M3 (71.13) and ROUGE scores (23.96).

Notably, **open-source models show competitive performance** in captioning: GLM-4.1 achieves the highest BLEU score (6.55) among all models, and GLM-4.5 performs well across multiple metrics. This suggests that caption generation may be less challenging than SVG generation tasks, allowing open-source models to close the performance gap.

6 CONCLUSION

We introduced VectorGym, a new comprehensive multi-task benchmark for SVG code generation that encompasses Sketch2SVG, SVG editing, Text2SVG, and SVG captioning. VectorGym introduces Sketch2SVG and releases the first dataset of complex, human-authored SVG edits, with gold-standard human annotations across all tasks. Our 7,000-sample evaluation and novel VLM-as-judge metrics reveal significant performance gaps between proprietary and open-source models, with open-source alternatives showing competitive results in editing and captioning. VectorGym establishes a new evaluation standard for visual code generation and provides robust benchmarks to advance SVG generation capabilities.

Use of LLMs We leveraged large language models (LLMs) to support different aspects of this work. They assisted with coding tasks needed to build the datasets and run experiments. Models such as GPT-4o, GPT-5, and Claude-4-Sonnet were also used to help with related work exploration and to ensure a comprehensive literature review. In addition, we employed LLMs for rephrasing and refinement while writing this paper, with the goal of improving flow, clarity, and correcting spelling errors. Importantly, we followed strict rules to preserve the accuracy and details of our contributions, and all generated content was carefully reviewed, manipulated, and edited by the authors.

Limitations VectorGym advances the capabilities that can be evaluated and optimized for more fine-grained control of current state-of-the-art SVG models. However, several limitations should be acknowledged. Our evaluation focuses on zero-shot and few-shot performance without training models specifically on our dataset, which was constrained by available GPU resources. Future work should focus on optimizing models for these specific tasks beyond zero-shot performance. Additionally, the evaluated models may contain inherent biases from their training data that could affect SVG generation quality and fairness across different visual styles and cultural representations.

Ethics Statement The VectorGym dataset is sourced from SVG-Stack, which aggregates content under appropriate licenses, and has been carefully filtered and curated to ensure quality and appropriateness. However, the models evaluated in this benchmark may exhibit biases inherited from their training data, potentially affecting the fairness and representation of generated SVG content across different demographics, cultures, and artistic styles. We encourage researchers to be mindful of these potential biases when using this benchmark and to consider fairness implications in future model development.

REFERENCES

Mu Cai, Zeyi Huang, Yuheng Li, Haohan Wang, and Yong Jae Lee. Leveraging large language models for scalable vector graphics-driven image understanding. *arXiv preprint arXiv:2306.06094*, 2023.

Defu Cao, Zhaowen Wang, Jose Echevarria, and Yan Liu. Svgformer: Representation learning for continuous vector graphics using transformers. In *CVPR*, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/papers/Cao_SVGformer_Representation_Learning_for_Continuous_Vector_Graphics_Using_Transformers_CVPR_2023_paper.pdf.

Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/file/bcf9d6bd14a2095866ce8c950b702341-Paper.pdf.

Louis Clouâtre and Marc Demers. Figr: Few-shot image generation with reptile. *arXiv:1901.02199*, 2019. URL https://arxiv.org/abs/1901.02199.

Jon Ferraiolo, Fujisawa Jun, and Dean Jackson. *Scalable vector graphics (SVG) 1.0 specification*. iuniverse Bloomington, 2000.

Jinke Li, Jiarui Yu, Chenxing Wei, Hande Dong, Qiang Lin, Liangjing Yang, Zhicai Wang, and Yanbin Hao. Unisvg: A unified dataset for vector graphic understanding and generation with multimodal large language models. *arXiv preprint arXiv:2508.07766*, 2025.

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM TOG (SIGGRAPH Asia)*, 2020. URL https://people.csail.mit.edu/tzumao/diffvg/.

Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. In *ICCV*, 2019. URL https://openaccess.thecvf.com/content_ICCV_2019/papers/Lopes_A_Learned_Representation_for_Scalable_Vector_Graphics_ICCV_2019_paper.pdf.

Antoine Quint. Scalable vector graphics. *IEEE MultiMedia*, 10(3):99–102, 2003.

Pradyumna Reddy et al. Im2vec: Synthesizing vector graphics without vector supervision. In *CVPR*, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/papers/Reddy_Im2Vec_Synthesizing_Vector_Graphics_Without_Vector_Supervision_CVPR_2021_paper.pdf.

Juan A Rodriguez, David Vazquez, Issam Laradji, Marco Pedersoli, and Pau Rodriguez. Figgen: Text to scientific figure generation. *arXiv preprint arXiv:2306.00800*, 2023a.

Juan A Rodriguez, David Vazquez, Issam Laradji, Marco Pedersoli, and Pau Rodriguez. Ocr-vqgan: Taming text-within-image generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3689–3698, 2023b.

Juan A Rodriguez, Haotian Zhang, Abhay Puri, Aarash Feizi, Rishav Pramanik, Pascal Wichmann, Arnab Mondal, Mohammad Reza Samsami, Rabiul Awal, Perouz Taslakian, et al. Rendering-aware reinforcement learning for vector graphics generation. *arXiv preprint arXiv:2505.20793*, 2025a.

Juan A. Rodriguez et al. Starvector: Generating scalable vector graphics code from images and text. In *CVPR*, 2025b. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Rodriguez_StarVector_Generating_Scalable_Vector_Graphics_Code_from_Images_and_Text_CVPR_2025_paper.pdf.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

Changyue Shu et al. Svgeditbench v2: Instruction-based editing for vector graphics. *arXiv:2504.15547*, 2025. URL https://arxiv.org/abs/2504.15547.

Yael Vinker, Ehsan Pajouheshgar, Jessica Y Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. Clipasso: Semantically-aware object sketching. *ACM Transactions on Graphics (TOG)*, 41(4):1–11, 2022.

Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. Iconshop: Text-based vector icon synthesis with autoregressive transformers. *arXiv preprint arXiv:2304.14400*, 2023.

Yingxue Xia et al. Vgbench: A comprehensive benchmark for vector graphics generation. In *EMNLP*, 2024. URL https://aclanthology.org/2024.findings-emnlp.1132/.

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 Computer Vision and Pattern Recognition Conference*, pp. 19487–19497, 2025.

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. *arXiv preprint arXiv:2504.06263*, 2025.

Tong Zhang, Haoyang Liu, Peiyan Zhang, Yuxuan Cheng, and Haohan Wang. Beyond pixels: Exploring human-readable svg generation for simple images with vision language models, 2023. URL https://arxiv.org/abs/2311.15543.

A VectorGym Datasets

Here we provide additional details on the VectorGym datasets. Figures 3 and 4 illustrate test samples for the Sketch2SVG (VG-Sketch) and SVG Editing (VG-Edit) tasks. We further describe the annotation methodology, data creation and sampling process, annotation details, and task definitions.

A.1 Annotation Methodology

A.1.1 Data Curation and Sampling

We extracted 7,000 high-quality samples from the SVG-Stack dataset through a rigorous multi-stage filtering process:

Visual Quality Assessment: Human experts manually reviewed SVG samples to identify visually appealing and well-formed graphics, filtering out corrupted, overly simplistic, or poorly designed samples.

Token Length Filtering: We applied token length constraints (2,000-8,000 tokens) to ensure meaningful complexity while maintaining computational feasibility. This range captures rich, detailed SVGs without exceeding practical processing limits for current VLMs.

Color Entropy Thresholding: We computed color entropy for each SVG to ensure visual diversity, filtering samples with insufficient color variation or monotonic palettes.

Random Sampling: Final samples were randomly selected to avoid systematic biases in content distribution.

From the curated 7,000 samples, we reserved 300 for testing across all tasks, with the remainder used for training and validation splits.

A.1.2 Annotation Vendor Partnership

We partnered with two specialized data annotation vendors to ensure task-specific expertise:

Vendor 1 - Sketch and Caption Generation: Specialized in visual content creation, responsible for sketch generation and text descriptions. Annotators were equipped with professional drawing tools (digital tablets, cameras for hand-drawn sketches) and trained on SVG visual analysis.

Vendor 2 - SVG Editing: Focused on technical SVG manipulation, staffed with annotators having design and vector graphics backgrounds. We developed custom SVG editing tools specifically for this project to enable precise modifications.

A.1.3 Annotator Demographics and Training

Our annotation team comprised over 20 annotators with diverse demographics and gender representation. All annotators underwent specialized training:

Technical Requirements: Background in design, vector graphics, or coding. Annotators were tested on SVG understanding and tool proficiency before assignment.

Equipment and Tools: Professional cameras for photographing hand-drawn sketches, digital drawing tablets, custom SVG editing software, and standardized annotation interfaces.

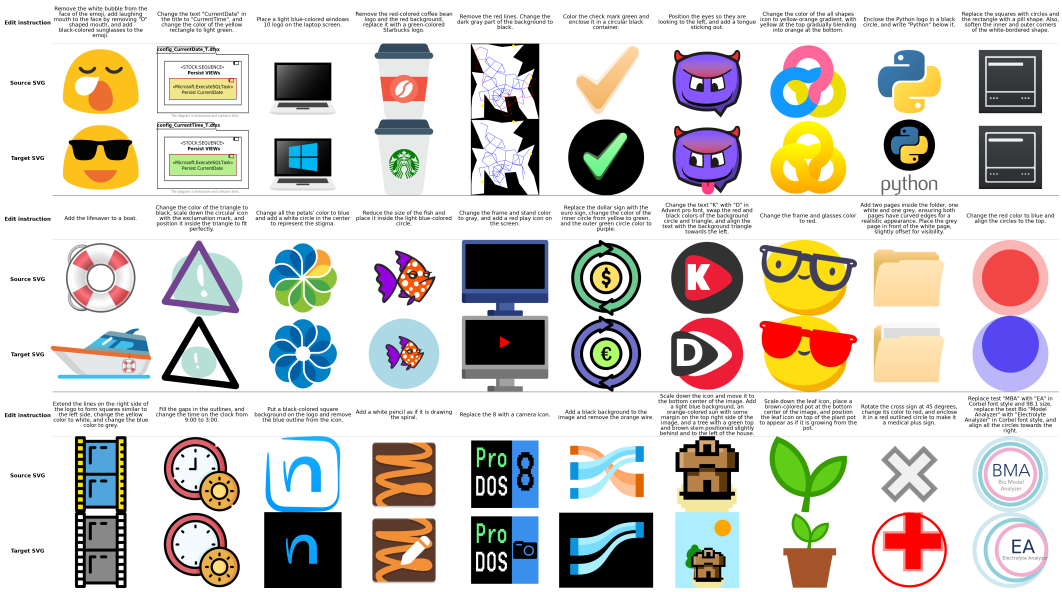


Figure 3: **Visualization of VG-Edit Test Examples.** We randomly sample 21 examples, and show the editing instruction to perform, along with the source and target vectors.

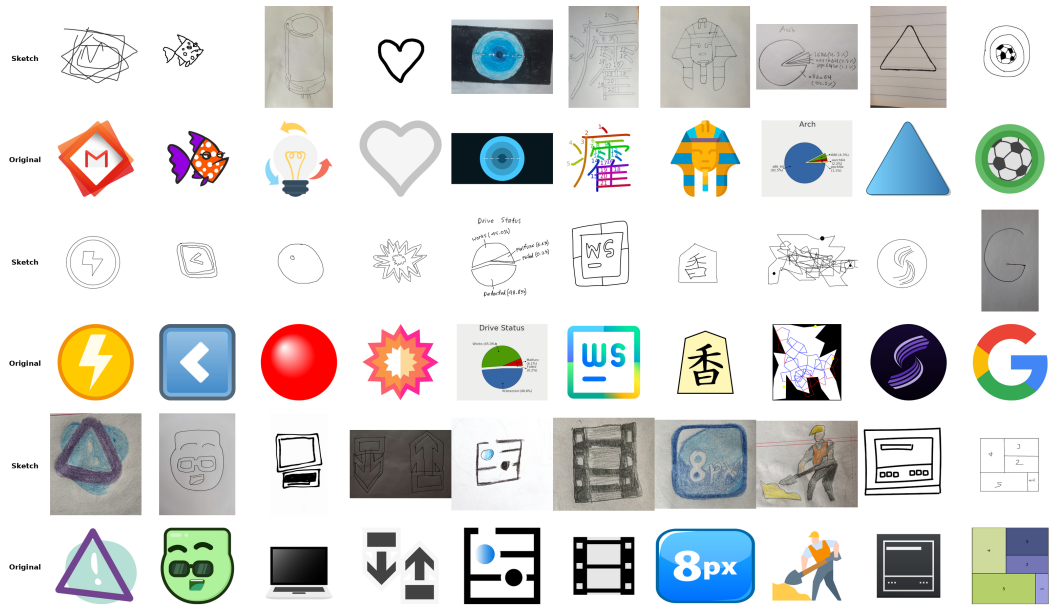


Figure 4: **Visualization of VG-Sketch Test Examples.** We randomly sample 30 examples, and show the sketch and the target vector.

A.1.4 Task-Specific Annotation Procedures

Sketch2SVG Generation: Annotators were provided with SVG images and asked to create corresponding sketches in two variants:

- **Hand-drawn:** Using pen or pencil on paper, photographed with standardized lighting and resolution
- **Digital:** Created using drawing tablets and stylus input for consistent digital sketches

Both variants included colored and black-and-white versions to test model robustness across different input modalities.

SVG Editing - Ensuring Complexity: We implemented strict complexity requirements to avoid trivial edits that could be synthetically generated:

Prohibited Simple Edits: Rotation, color changes, scaling, basic shape removal - operations easily automated by current LLMs.

Required Complex Edits: Path modifications, primitive additions, parameter adjustments, conceptual additions requiring semantic understanding. For example:

- Adding elements from other SVGs in the database (e.g., incorporating a needle shape into a hammer SVG)
- Modifying facial expressions in character illustrations
- Converting chart types (pie to bar charts)
- Structural modifications requiring new geometric primitives

Caption Generation: We implemented a comprehensive multi-stage process for generating high-quality text descriptions:

1. **Detailed Visual Description:** Annotators created comprehensive descriptions of vector graphics, with particular emphasis on color specification. To ensure color accuracy, annotators were required to include hexadecimal color codes in parentheses alongside natural language color descriptions (e.g., "red (#FF0000)").
2. **Cross-validation with VLM:** All human-generated descriptions were processed and cross-validated using Qwen2-VL-32B to ensure consistency and completeness of visual descriptions.
3. **Instruction Reformatting:** Captions were systematically reformatted from descriptive statements into instruction-style prompts suitable for the Text2SVG generation task. This process generated two distinct variants:
   - **Hexadecimal Color Version:** Instructions containing precise hexadecimal color specifications, which empirically demonstrate superior SVG generation accuracy
   - **Natural Language Color Version:** Instructions using standard color names for broader accessibility
4. **Quality Validation:** Final consistency checks and inter-annotator agreement measurement across all caption variants

Quality Assurance: All annotations underwent rigorous quality control including automated SVG syntax validation, human verification of task requirements, and consistency checks across related task pairs.

B Additional Qualitative Results

We provide additional figures (Figures 5–9) showing qualitative results of the models on the presented tasks.

Figure 5: Visualization of test performance on the Sketch2SVG task. When the input sketch lacks color, models tend not to introduce new colors. In contrast, when color is present in the sketch, models successfully reproduce it in the generated SVG.

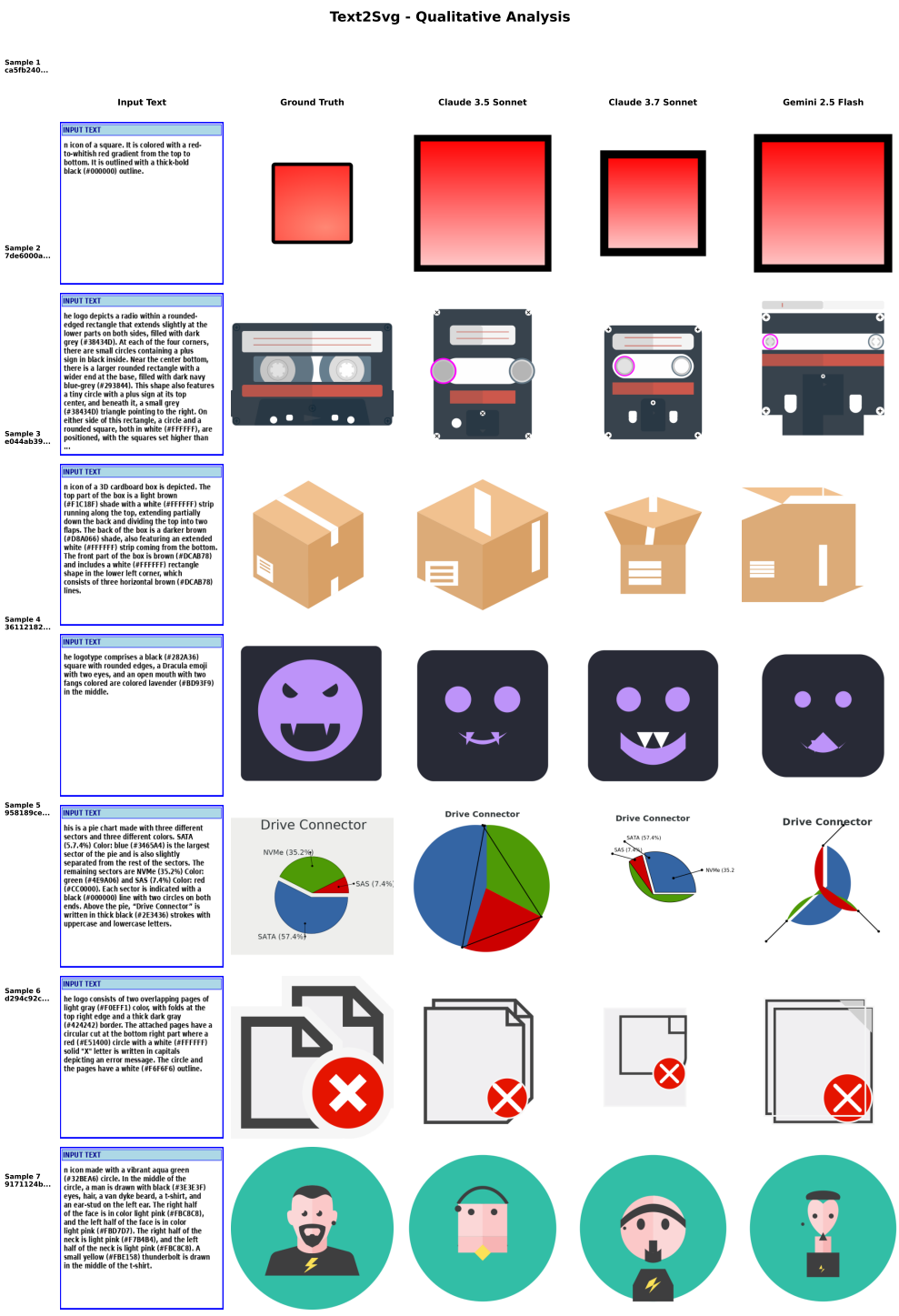


Figure 6: Qualitative analysis of Text2SVG generation results. The figure shows examples of text2SVG generation across different model performances. Examples demonstrate successful generations with accurate semantic understanding and geometric representation, as well as common failure modes including incorrect primitive usage, semantic misunderstanding, and incomplete shape representations.

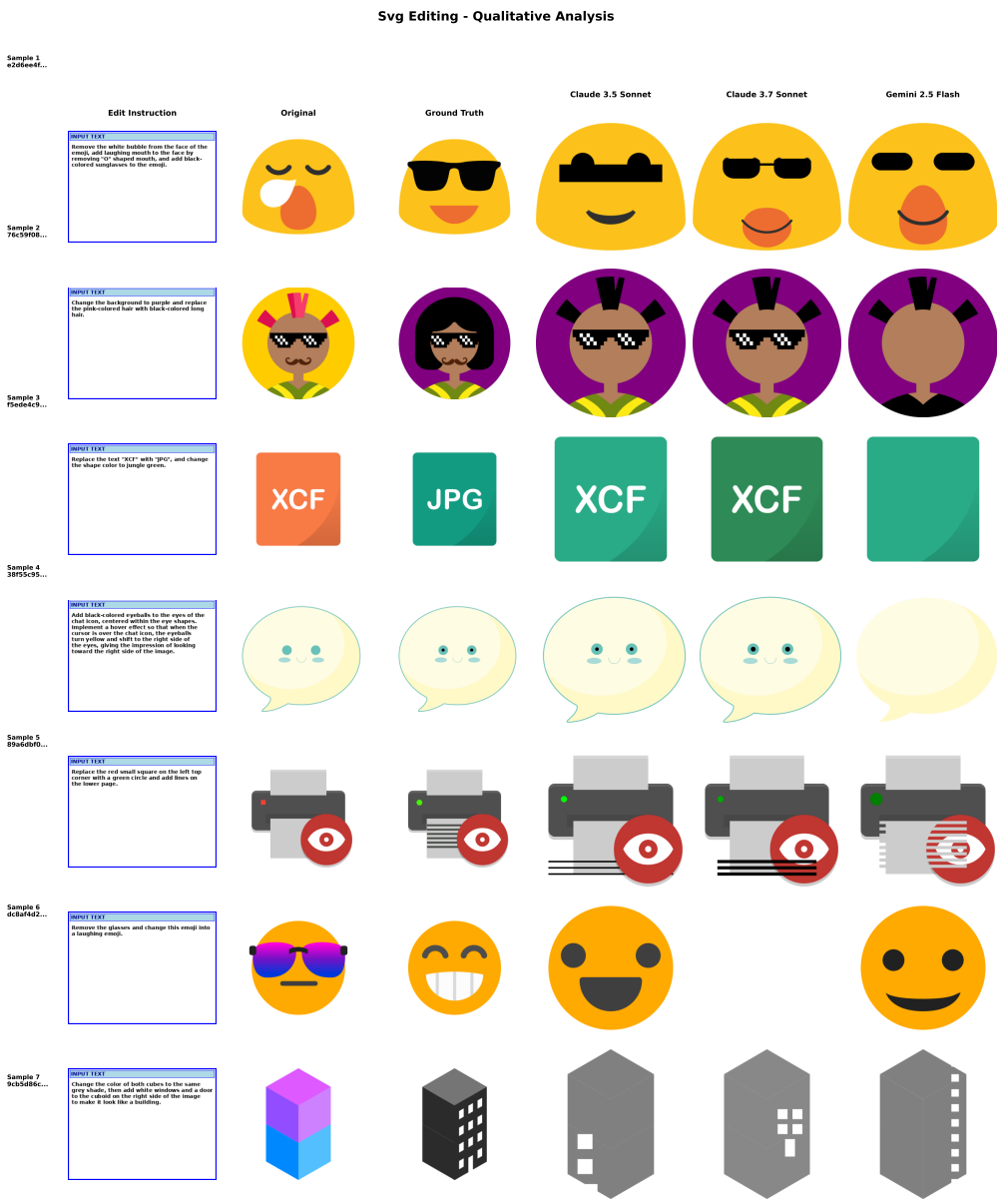


Figure 7: **Qualitative analysis of SVG editing with natural language instructions**. The figure demonstrates model performance on various editing tasks including color changes, geometric transformations, and structural modifications. Examples show input SVG (left), editing instruction (center), and generated output (right). Successful cases highlight accurate instruction parsing and precise SVG manipulation, while failure cases reveal challenges in understanding complex instructions and maintaining visual coherence.

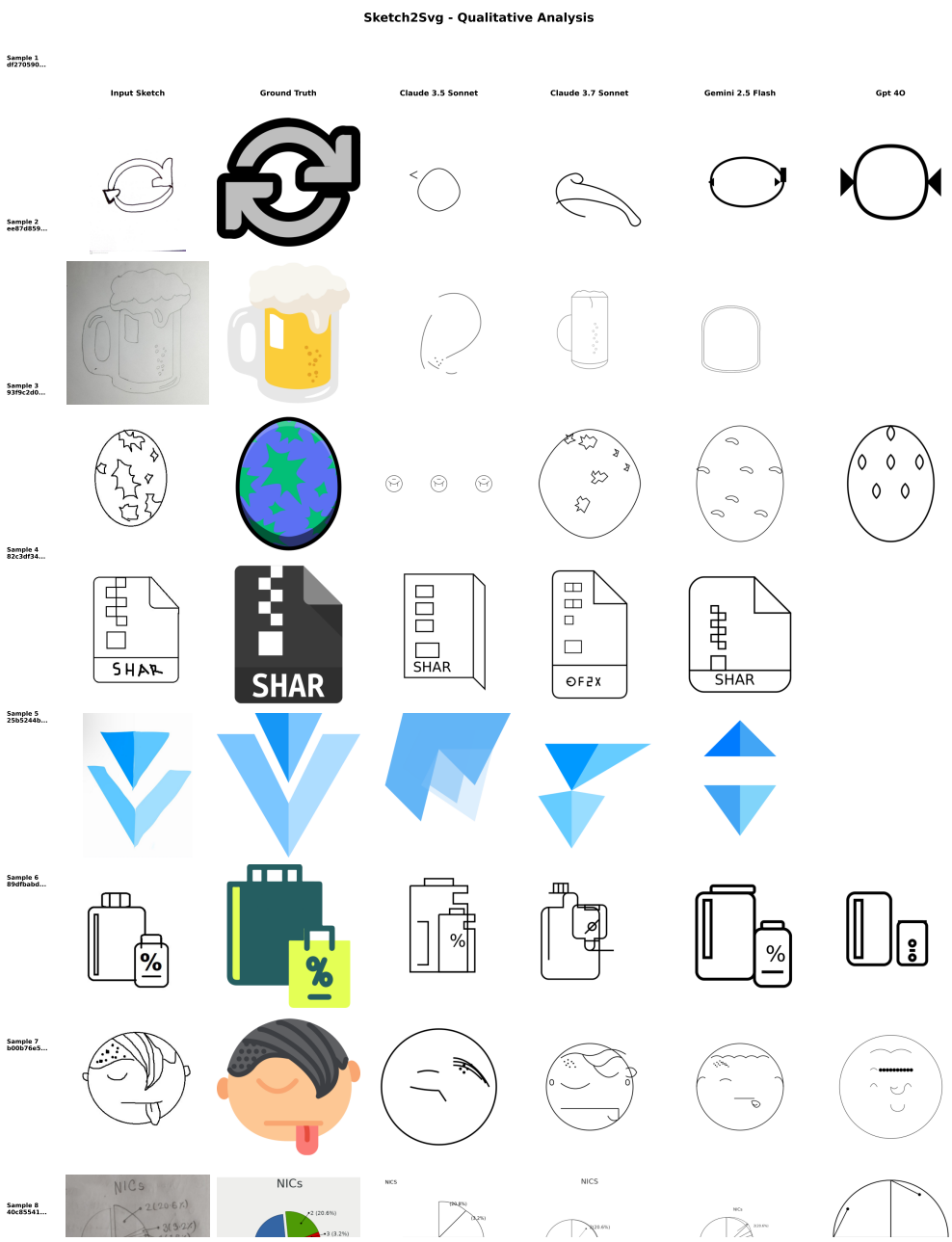


Figure 8: **Qualitative analysis of Sketch2SVG generation results**. The figure illustrates model performance in converting hand-drawn sketches to clean SVG code. Examples display input sketches (left), ground truth SVG (second column), and model-generated SVGs (rest of the columns).

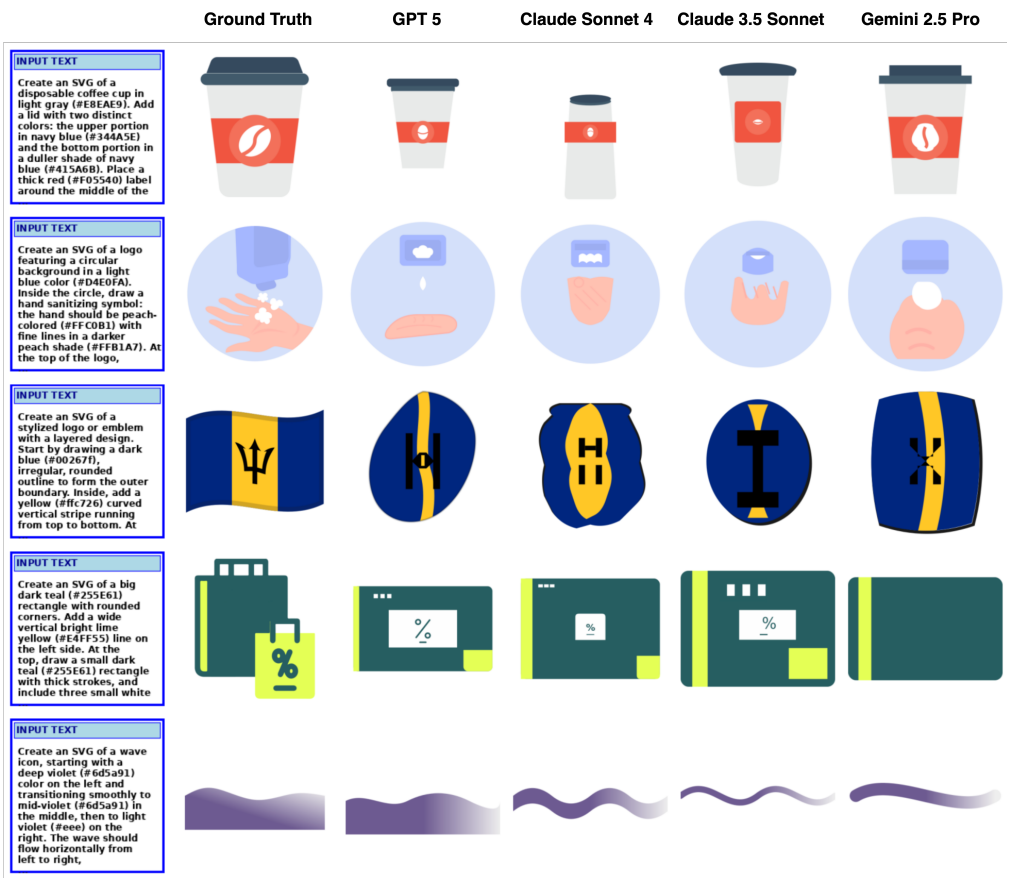


Figure 9: **Qualitative analysis of Text2SVG generation results**.

C VLM AS A JUDGE

We present the exact prompts used in our experiments—both for VLM-as-a-judge and for generation, color-coded by task.

Prompt 1: Used for VLM-as-a-Judge Score (Text2Svg)

You are an expert evaluator for text-to-SVG generation tasks. Your job is to assess how well a generated image matches a given text description.

Evaluation Instructions: Please evaluate how well the generated image matches the text description using the following criteria:

1. **Visual Accuracy:** Does the image accurately represent what is described in the text?
2. **Completeness:** Are all elements mentioned in the text present in the image?
3. **Quality:** Is the image clear, visually appealing, and well-formed?

Text Description: {caption}

Scoring Rubric (1--10):

- **1--3 (Poor):** Mostly unrelated to the text, major inaccuracies or missing elements, poor quality.
- **4--6 (Fair):** Partially related to the text, some missing or incomplete elements, moderate quality.

Table 6: Comparison of SVG datasets and benchmarks across scale, task coverage, and annotation quality. VectorGym uniquely combines multi-task evaluation with human-verified quality across diverse SVG content types. Existing datasets either focus on single tasks, rely on synthetic annotations, or lack comprehensive evaluation frameworks.

Dataset	Year	Size	Content types	Tasks	Annotation
VG-Sketch (ours)	2025	6,500	icons, fonts, diagrams, emojis	Sketch–to–SVG	human
VG-Text2SVG (ours)	2025	6,500	icons, diagrams, emojis, fonts	Text–to–SVG	human
VG-Edit (ours)	2025	6,500	diverse	SVG editing	human
SVG–Stack	2025	2,283,875	diverse (icons, logos, diagrams)	SVG corpus	unlabeled
Text2SVG–Stack	2025	2,180,000	diverse (paired text→SVG)	Text–to–SVG	synthetic captions
SVG–Fonts	2025	1,928,271	fonts, glyphs	SVG corpus	unlabeled
SVG–Icons	2025	89,376	icons	SVG corpus	unlabeled
SVG–Emoji	2025	10,043	emojis	SVG corpus	unlabeled
MMSVG–2M	2025	2,000,000	icons, illustrations, characters	Image–to–SVG, Text–to–SVG,	mixed (web + synthesized)
UniSVG	2025	525,000	unified multi–domain	Image–to–SVG, Text–to–SVG, Understanding	mixed
SVGX–SFT–1M	2025	1,000,000+	diverse (instruction↔SVG)	instruction–following	synthetic (LLM)
SVG–1M (SVGen)	2025	1,000,000	icons	Image–to–SVG, Text–to–SVG,	synthetic (LLM)
FIGR–SVG (from FIGR–8)	2025	1,330,000	icons	Text/Image–to–SVG	converted + synthetic text
DeepSVG dataset (SVG–Icons8)	2020	100,000	icons	SVG generation	curated
SVGenius	2025	2,377	diverse	understanding; editing	human–verified
VGBench	2024	4,279 + 5,845	multi–format (SVG, TikZ, Graphviz)	understanding; generation	synthetic + verified
SVGEditBench v2	2025	1,683 triplets	emojis, icons	SVG editing	synthetic prompts
VectorEdits	2025	270,000+	diverse	SVG editing (instr.–guided)	synthetic (VLM)
Quick Draw!	2017	50,000,000+	sketches	sketch recognition	human
IconDesc (UI icons)	2024	˜1,400	UI icons	captioning (alt–text)	human

- **7--9 (Good):** Mostly accurate and complete, minor issues in detail or quality, clear and visually appealing.
- **10 (Excellent):** Perfect match to the text description, all elements accurate, excellent quality.

Output format:

```
score: <number 1-10>
explanation: <brief explanation>
```

Prompt 2: Used for VLM-as-a-Judge Score (Sketch2Svg)

You are an expert evaluator for sketch-to-SVG generation tasks. Your job is to assess how well a generated image matches an input sketch.
Evaluation Instructions: Please evaluate how well the generated image matches the input sketch using the following criteria:

1. **Shape Fidelity:** Do the shapes in the generated image match the sketch?

2. **Structure Preservation:** Is the overall structure and layout maintained?
3. **Detail Accuracy:** Are important details from the sketch captured?
4. **Quality:** Is the generated image clear, visually appealing, and well-formed?

Scoring Rubric (1--10):

- **1--3 (Poor):** Significant deviations in shape or structure, poor quality.
- **4--6 (Fair):** Partial match with noticeable inaccuracies, moderate quality.
- **7--9 (Good):** Closely matches the sketch with minor issues, clear and appealing.
- **10 (Excellent):** Perfectly matches the sketch, all details accurate, excellent quality.

Output format:

```
score: <number 1-10>
explanation: <brief explanation>
```

Prompt 3: Used for VLM-as-a-Judge Score (Svg-Editing)

You are an expert evaluator for SVG editing tasks. Your job is to assess how well a generated image follows editing instructions applied to an original image.
Evaluation Instructions: Please evaluate how well the generated image follows the editing instructions applied to the original image using the following criteria:

1. **Instruction Compliance:** Does the generated image follow the editing instructions?
2. **Original Preservation:** Are unmodified parts of the original image preserved?
3. **Coherence:** Does the edited image look natural and coherent?
4. **Quality:** Is the edited image clear, visually appealing, and well-formed?

Editing Instructions: {caption}
Scoring Rubric (1--10):

- **1--3 (Poor):** Instructions not followed, major quality or coherence issues.
- **4--6 (Fair):** Partial compliance, noticeable issues in preservation or quality.
- **7--9 (Good):** Mostly compliant with minor issues, clear and appealing.
- **10 (Excellent):** Perfectly follows instructions, preserves original, excellent quality.

Output format:

```
score: <number 1-10>
explanation: <brief explanation>
```

Prompt 4: Unified VLM-as-a-Judge (All Tasks)

You are an expert evaluator for SVG-related tasks (Text-to-SVG, Sketch-to-SVG, and SVG Editing). Assess the quality of a candidate

```
SVG output given the corresponding input (text prompt, sketch image
description, or original SVG plus edit instruction).
Please score on a 0--10 scale and provide a brief explanation.  Use
the following criteria where applicable:
1) Visual Accuracy and Fidelity:  How well does the generated SVG
match the intended content and structure?  2) Semantic Alignment:
Does it satisfy the input requirements (text description, sketch
content, or editing instruction)?  3) Code Quality and Efficiency:
Is the SVG valid, reasonably compact, and using appropriate
primitives and attributes?  4) Aesthetics and Clarity:  Is the
result clean, legible, and stylistically coherent?
Output format:  score:  <0-10> explanation:  <one or two sentences>
```

Prompt 5: Used for Text2SVG Generation

```
You are an expert in generating SVG representations of textual
descriptions.  Follow these steps carefully:
1.  Analyze the given text input and identify the key visual
elements it describes.
2.  Convert the description into a minimal and clear SVG
representation using basic SVG shapes such as <rect>, <circle>,
<line>, and <path>.
3.  Ensure the SVG design is simple, scalable, and directly
represents the input text.
4.  Do not include any additional text, explanations, comments, or
formatting|only output valid SVG code.
5.  The output must be a complete SVG document, starting with
'<svg>' and ending with '</svg>'.
```

Prompt 6: Used for Sketch2SVG Generation

```
You are an expert in converting images into vector-based SVG
representations.  Follow these steps carefully:
1.  Analyze the input image and extract its key shapes, contours,
and structural features.
2.  Convert these features into an SVG format using appropriate
primitives such as <path>, <circle>, <rect>, and <line>.
3.  Optimize the SVG output by simplifying paths and reducing
unnecessary complexity while maintaining accuracy.
4.  Do not include any additional text, explanations, comments, or
formatting|only output valid SVG code.
5.  The output must be a complete SVG document, starting with
'<svg>' and ending with '</svg>'.
```

Prompt 7: Used for SVG Editing Generation

```
You are an expert in editing SVG images based on text instructions.
Follow these steps carefully:
1.  Analyze the original SVG and the editing instruction.
2.  Apply the requested modifications while preserving the overall
structure.
3.  Ensure the edited SVG is valid and well-formed.
4.  Do not include any additional text, explanations, comments, or
formatting|only output valid SVG code.
5.  The output must be a complete SVG document, starting with
'<svg>' and ending with '</svg>'.
```

Prompt 8: Used for SVG Captioning Rubric

```
You are a concise evaluator of caption similarity.
Compare a PREDICTION caption to a GROUND-TRUTH caption (no image).
Judge semantic meaning, not exact wording.

Rules:
- Accept paraphrases and synonyms.
- Treat numbers, counts, colors, attributes, relations, and negation as strict (penalize mi
- Penalize unsupported or contradictory details (hallucinations) more than omissions.
- Ignore casing and punctuation (except negation words like "no/not/without").
- Do not use world knowledge; compare only what the texts state.

Scoring (return a single integer 0-5):
5 = Semantically equivalent or near-paraphrase; all key facts align; no contradictions.
4 = Very close; only a minor detail missing/different; no contradictions.
3 = Partially correct; several core elements match but some important detail is missing or
2 = Weak overlap; topic similar but multiple important errors or added unsupported specific
1 = Minimal overlap; only a very generic element matches.
0 = Unrelated or contradicts core facts (e.g., negation flip, wrong main objects/actions).

Output ONLY the integer score (0-5). No words, no JSON, no explanations.

GROUND-TRUTH: {ground_truth_caption}
PREDICTION: {prediction_caption}
Score 0-5.
```

D CAPTIONING METRICS

We compute captioning metrics pairwise over aligned (reference, prediction) captions and average across the corpus.

- **BLEU (corpus BLEU)**: n-gram precision with brevity penalty; 0–100 (higher is better).
- **chrF++ (CHRF)**: Character n-gram F-score (word order=2); 0–100 (higher is better).
- **ROUGE-L (F1)**: Longest common subsequence overlap (F1); 0–100 (higher is better).
- **BERTScore (F1)**: Semantic similarity via contextual embeddings; 0–100 (higher is better). `rescale_with_baseline=False`.
- **BGE-M3 Similarity**: Average cosine similarity of `BAAI/bge-m3` sentence embeddings; 0–100 (higher is better).
- **GPT-5 Rubric Similarity**: LLM-judged semantic agreement on a 0–5 rubric mapped to 0–100; higher is better.