VGBench: Evaluating Large Language Models on Vector Graphics Understanding and Generation

Bocheng Zou* **Mu Cai*** **Jianrui Zhang** **Yong Jae Lee**
Department of Computer Science
University of Wisconsin-Madison
{bochengz,mucai,harrisz,yongjaelee}@cs.wisc.edu

https://vgbench.github.io/

Abstract

In the realm of vision models, the primary mode of representation is using pixels to rasterize the visual world. Yet this is not always the best or unique way to represent visual content, especially for designers and artists who depict the world using geometry primitives such as polygons. Vector graphics (VG), on the other hand, offer a textual representation of visual content, which can be more concise and powerful for content like cartoons, sketches and scientific figures. Recent studies have shown promising results on processing vector graphics with capable Large Language Models (LLMs). However, such works focus solely on qualitative results, understanding, or a specific type of vector graphics. We propose *VGBench*, a comprehensive benchmark for LLMs on handling vector graphics through diverse aspects, including (a) both visual understanding and generation, (b) evaluation of various vector graphics formats, (c) diverse question types, (d) wide range of prompting techniques, (e) under multiple LLMs and (f) comparison with VLMs on rasterized representations. Evaluating on our collected 4279 understanding and 5845 generation samples, we find that LLMs show strong capability on both aspects while exhibiting less desirable performance on low-level formats (SVG). Both data and evaluation pipeline will be open-sourced at `https://vgbench.github.io`.

1 Introduction

Current vision models are mostly built on pixels, rasterizing the visual world into a matrix representation. Such rasterized represents diverse visual content with equally sized elements. But pixels are not the only way to represent the visual world. For contents such as cartoons, sketches or scientific figures, a different representation using explicit geometry primitives can be more concise and beneficial. Vector graphics offer such a textual representation for visual content via geometry primitives, e.g., circles and polygons, as shown in Figure 1 (a). Vector graphics have been critical for designers and artists since the geometry primitives can be easily manipulated. Vector representations include Scalable Vector Graphics (SVG), TikZ, Graphviz, etc.

Vector Graphics vector representations make it possible to conduct visual understanding and generation with LLMs such as GPT-4 (OpenAI, 2023b). Recent studies (Bubeck et al., 2023; Cai et al., 2023; Rodriguez et al., 2023) showcase LLMs' superior capability across different perspectives. However, those works either (1) only show qualitative results (Bubeck et al., 2023), (2) only study vector graphics understanding (Wang et al., 2024) and not generation, or (3) only study one specific type of vector graphics such as SVG (Cai et al., 2023; Wang et al., 2024; Rodriguez et al., 2023) or TikZ (Belouadi et al., 2024). Therefore, the community lacks a comprehensive LLM benchmark for vector graphics.

In this paper, we propose *VGBench* to comprehensively evaluate LLMs' vector graphics processing capabilities via different aspects: *VGBench* (1) includes both visual understanding (*VGQA*) and generation (*VGen*); (2) evaluates diverse vector graphics formats such as SVG, TikZ, and Graphviz; (3) covers a set of taxonomies from low-level vision to high-level semantics, from color, shape, to category and advanced reasoning questions such as usage and the relation between objects; (4) adopts a variety of prompting techniques, such as zero-shot prediction, chain-of-thought reasoning, in-context learning, *etc*.; (5) evaluates diverse LLMs including GPT-4 (OpenAI, 2023b), GPT-3.5 (OpenAI, 2023a), Llama-3-8B-Instruct, Llama-3-70B-Instruct (Meta, 2024), Qwen2-7B-Instruct, Qwen2-72B-Instruct (qwe, 2024), Phi-3-mini-128k-instruct, Phi-3-medium-128k-instruct (Abdin et al., 2024), gemini-1.5-

*Equal Contribution.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 3647–3659
November 12-16, 2024 ©2024 Association for Computational Linguistics

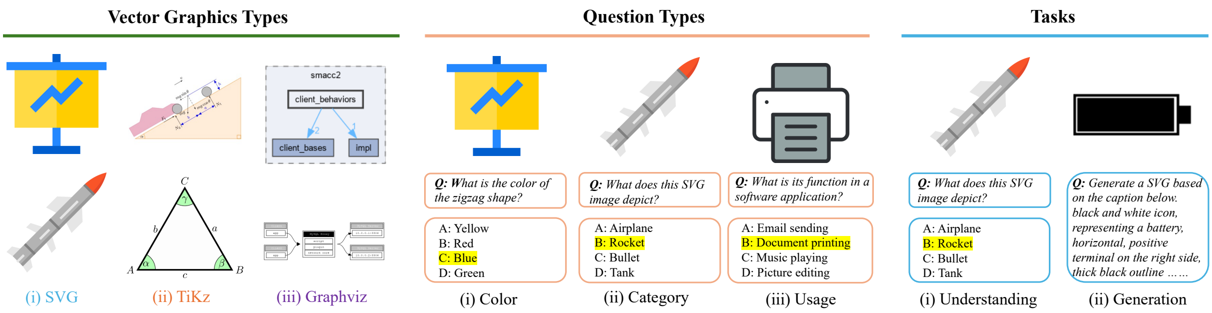


Figure 1: *VGBench* is the **first comprehensive** vector graphics (VG) *understanding* and *generation* benchmark across diverse vector graphics types, question types, and prompting techniques on a rich set of SoTA LLMs. Our large scale benchmark consists of 4279 multi-choice question-answer pairs and 5845 VG-caption pairs.

pro (Reid et al., 2024); and (6) evaluates the VLM LLaVA-1.5-13b (Liu et al., 2024) over rasterized representations of images in our benchmark.

We collect 4279 high-quality visual question-answer (QA) pairs for vector graphics (VG) understanding and 5845 VG-caption pairs for vector graphics generation. The vector graphics code is collected from existing datasets and the Internet. For visual question answering, we use a semi-automated pipeline to curate the questions. Specifically, we prompt GPT-4V(ision) to generate question-answer pairs given the provided in-context examples. Human annotators then filter the generated QA pairs to get the final high-quality vector graphics QA dataset. We use the gathered questions to evaluate if an LLM can understand vector graphics correctly. For text-to-vector-graphic generation (T2VG), we utilize GPT-4V to generate the captions and then use CLIP Score (Hessel et al., 2021) and Fréchet Inception Distance (FID) (Heusel et al., 2017) to evaluate the quality of the LLM generated vector graphics code.

Our key findings are as follows:

- LLMs show much better vector graphic understanding capability in TikZ and Graphviz than SVGs. TikZ and Graphviz include more high-level semantics compared to SVG, which is composed of low-level geometry primitives. This demonstrates that LLMs are more capable in understanding vector graphics code with high-level semantics.

- Advanced prompting techniques such as in-context learning or chain-of-thought prompting can bring significant performance boost for SVG, a low-level VG format.

- LLMs show strong vector graphics generation ability on TikZ and Graphviz format compared to SVG format, hinting that TikZ or Graphviz might be a better medium for LLMs to manipulate vector graphics.

- In both understanding and generation, GPT-4 shows the strongest performance, yet open-source models such as Llama-3-70b shows competitive performance in understanding tasks.

We hope that our work can serve as a foundation for LLM vector graphics understanding and generation benchmarking, and motivate further work to improve such capabilities. Our benchmark dataset and evaluation pipeline will be released.

2 Related Work

2.1 Vector Graphics

Vector graphics represent images using basic geometric elements like points, lines, and curves, rather than pixels. This method offers an alternative to raster graphics, providing advantages such as infinite scalability without losing detail and easy human manipulation.

There are a variety of vector graphics formats, such as SVG (Quint, 2003), TikZ (Mertz and Slough, 2007) and Graphviz (Gansner, 2009). SVG format defines 14 functional areas or feature sets and represents graphics by recording basic information associated to these primitives, such as their coordination and scales, in an XML file. TikZ format defines some commands to build basic geometric elements and is mainly used with LaTeX. In practice, third-party packages are also commonly used with TikZ to build more diverse images. Graphviz (Gansner, 2009) is a vector graphics format that focuses on representing different kinds of graphs. In this paper, we explore the said three kinds of vector graphics to provide a thorough and comprehensive analysis regarding the reasoning capabilities of LLMs on vector graphics.

2.2 Evaluation for Image Understanding and Generation

Works on Image Understanding are mainly based on raster images. VQA (Antol et al., 2015) first introduced the task of free-form and open-ended Visual Question Answering and evalauted existing LSTM-CNN based methods. CLIP (Radford et al., 2021) introduces two encoders for both texts and images to achieve an aligned representation to serve as a baseline for many image understanding tasks. LLaVA (Liu et al., 2023) and LLaMA-Adapter (Zhang et al., 2023) propose approaches to solve general-purpose visual and language understanding problems based on large language models.

While vector graphics can usually be converted to a raster image easily (Gharachorloo et al., 1989), there are few works that try to directly understand the vector graphics format. (Jiang et al., 2021) explores such a way using graph neural networks. (Wang et al., 2024) utilizes large language models to understand vector graphics. In our work, we utilize multiple prompting methods, to be mentioned in the following section, to evaluate different LLMs' vector graphics understanding capabilities by prompting them with the vector graphics code directly.

Most machine learning based image generation models aim to generate raster images (Kingma and Welling, 2013; Goodfellow et al., 2020; Ho et al., 2020; Ramesh et al., 2021). Some research focus on generating vector graphics in text format. Many works generate vector graphics from a raster image (Diebel, 2008; Xia et al., 2009; Ha and Eck, 2017; Ma et al., 2022). Leveraging language models, some try to generate text representing vector graphics directly (Carlier et al., 2020; Wu et al., 2023; Rodriguez et al., 2023). In our work, we provide a different approach to evaluate vector graphics generation via leveraging competent multimodal models such as GPT4-V (OpenAI, 2023b) to generate a detailed caption from a rasterized image of a vector graphics object, based on which other LLMs will be generating vector graphics code for the same object during evaluation. We argue that models like GPT4-V can provide high-quality captions for us to automate part of the evaluation process.

2.3 Prompting Techniques for Large Language Models

A variety of prompting strategies have been proven capable of boosting the performance of LLMs, such as GPT4 (Achiam et al., 2023). Few-shot learning (Brown et al., 2020b) requires the user to give a few examples of the task to the LLM, while Chain of Thought (Wei et al., 2022) instructs the LLM to think step by step to achieve higher performance. In-context learning (Brown et al., 2020a) provides few-shot examples at inference time, and shows strong performance boost without updating the model's parameters. In this paper, we broadly evaluate LLMs' vector graphic understanding capability by employing the aforementioned prompting techniques.

3 Tasks and Experiments

We first introduce the source of our vector graphics images in Sec. 3.1, and then describe the experiment settings in Sec. 3.2. After that, we detail our tasks, benchmark creation, evaluation pipeline and results for vector graphics understanding and generation in Sec. 3.3 and Sec. 3.4, respectively. Finally, we provide in-depth analyses on the performance under different LLMs, different sequence lengths, and reasoning processes in Sec. 3.5.

3.1 Vector Graphics Data Collection

We collect vector graphics samples for both understanding tasks and generation tasks from a variety of sources. For samples in SVG format, we collect them from a large-scale SVG repository.[1] We sample the TikZ format vector graphics code from the DaTikZ dataset (Belouadi et al., 2024). We sample the Graphviz code used to build our dataset by crawling GitHub.[2]

3.2 Experiment Settings

Vector Graphics Types Here we consider three major types of vector graphics: Scalable Vector Graphics (SVG), TikZ, and Graphviz. SVG is exceptionally versatile and suitable for web applications, allowing for detailed graphical representations that scale infinitely without loss of quality. This enables SVGs to theoretically represent any visual content including complex animations and

[1] https://www.kaggle.com/datasets/victorcondino/svgicons

[2] https://github.com/

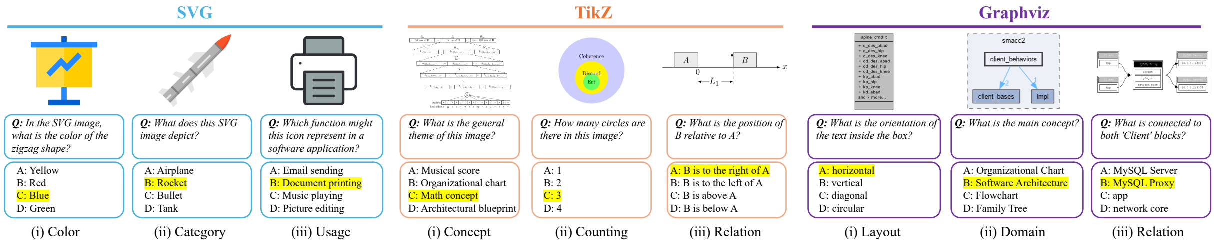


Figure 2: Examples of the vector graphics QAs for diverse formats including SVG, TikZ, and Graphviz in *VGQA*.

SVG				TikZ				Graphviz			
Category	Color	Usage	Overall	Concept	Counting	Relation	Overall	Layout	Domain	Relation	Overall
869	671	688	2228	580	239	320	1139	319	418	175	912

Table 1: Statistics of *VGQA*. We collect a large set of QAs for each vector graphics format under diverse tasks, resulting in 4279 QAs in total.

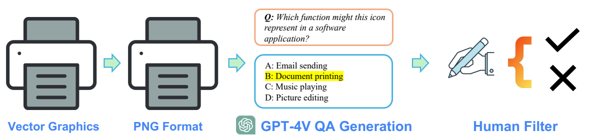


Figure 3: The semi-automatic curation pipeline in *VGQA*. Vector graphics are converted into PNG format, then GPT-4V is utilized to generate the questions and answers (QA) candidates. Finally, human annotators filter the QA pairs to obtain the high-quality QA dataset.

interactive elements. TikZ, in contrast, is specifically tailored for creating high-precision scientific illustrations within LaTeX documents, offering a comprehensive suite of tools for detailed diagrammatic representations; it encompasses a broad spectrum of high-level semantics such as "circuit diagrams, complex mathematical illustrations, and structured diagrams". Graphviz, on the other hand, belongs to the family of automated graph drawing tools, which are optimized for generating diagrams from abstract descriptions and data structures, making it ideal for visualizing hierarchical information, such as state machines, organizational charts, and network infrastructures.

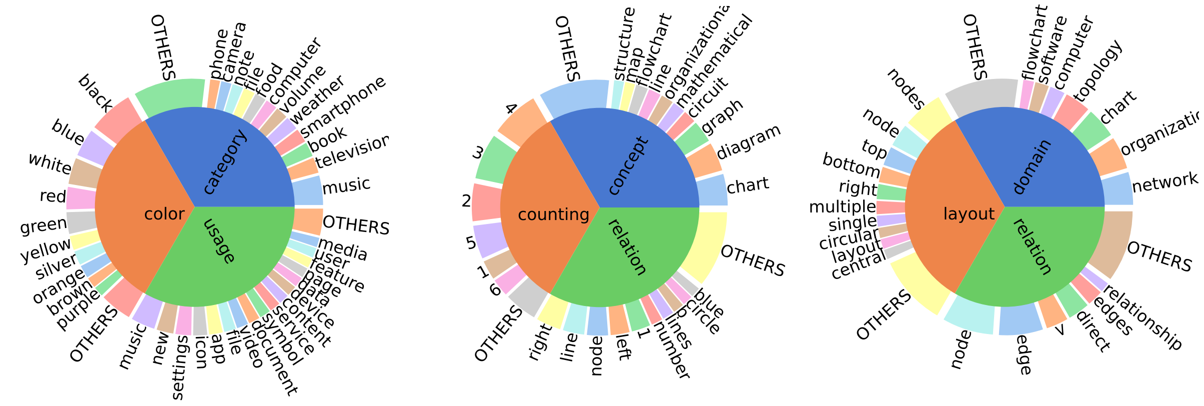
Language Models We primarily use GPT-4 (1106 version) (OpenAI, 2023b) as the medium for vector graphics understanding and generation. This is because GPT-4 shows superior language reasoning and generation capabilities, as previously mentioned in Section 2.3. We also evaluated other proprietary models such as GPT-4o, GPT-3.5 (OpenAI, 2023a) and gemini-1.5-pro (Reid et al., 2024), along with many other open-source LLMs that are highly capable, including Llama-3-70B-Instruct (Meta, 2024), Llama-3-8B-Instruct, Qwen2-7B-Instruct, Qwen2-72B-Instruct (qwe, 2024), Phi-3-mini-128k-instruct and Phi-3-medium-128k-instruct (Abdin et al., 2024).

Tasks We consider two major tasks in computer vision: (1) visual understanding, and (2) visual generation. We design multiple choice questions to evaluate vector graphics understanding while using image generation metrics including Fréchet Inception Distance (FID) (Heusel et al., 2017) and CLIP score (Hessel et al., 2021) to measure the quality and correctness of generated vector graphics.

Prompting Techniques We adopt three widely used prompting techniques: zero-shot, chain-of-thought (CoT) prompting, and in-context learning (few-shot prompting). For CoT, we instruct the LLM to think step by step by appending "Please think step by step" to the initial question, using multi-round dialog to let the LLM consider each option separately before figuring out the answer. For in-context learning, we provide 3 examples of the same question type.

3.3 *VGQA*: Vector Graphics Understanding Benchmark

Tasks *VGQA* is designed to evaluate models' vector graphics understanding capability. We systematically design a range of tasks based on the nature of each vector graphics category, aiming at a comprehensive evaluation across different semantic levels. For SVG, we design three types of questions: color, category, and usage; for TikZ, we use concept, counting, and relations as types of questions;



(a) SVG (b) TikZ (c) Graphviz

Figure 4: Word distribution based on question categories for each vector graphic type. The top 20 words are sampled from the answers to each type of question. Words with a frequency of less than 4% are represented as "OTHERS".

SVG				TikZ				Graphviz			
Category	Color	Usage	Average	Concept	Counting	Relation	Average	Layout	Domain	Relation	Average
86.9	67.1	68.8	74.3	58.0	47.8	32.0	45.6	58.0	83.6	31.8	57.0

Table 2: Human filtering passing rate of *VGQA*. TikZ and Graphviz show lower and less than half passing rate than SVG, indicating that even SoTA models exhibit poor vector graphic understanding capabilities in certain areas.

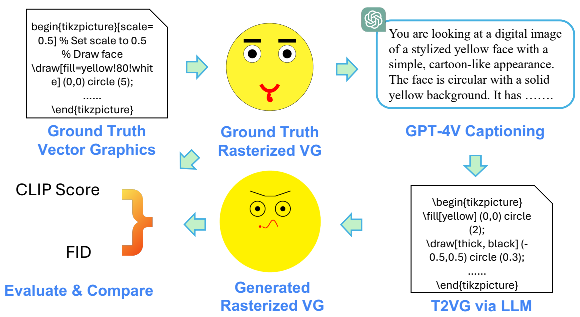


Figure 5: The automatic generation pipeline in *VGen*. The vector graphics collected from the Internet is first rendered into the ground truth image then captioned by GPT-4V. The caption is fed into the target LLM to generate new vector graphics, which will be compared with the caption using CLIP Score and FID for a similarity score. The score is then compared with the similarity score between the ground truth and the same caption as the upper bound.

while for Graphviz, we design layout, domain, and relations. Examples are shown in Figure 2.

Benchmark Creation and Evaluation We employ a semi-automatic benchmark curation pipeline for *VGQA*, as shown in Figure 3. Specifically, we render code representing vector graphics into a PNG image before leveraging GPT-4V (OpenAI, 2023b) to generate the 4-choice question-answer candidates. Then, human annotators with rich vision-linguistic knowledge make binary annotations to mark whether both the question and the answer of a candidate are rational, correct and belong to that specific type. Our approach brings several benefits: (i) annotation cost is greatly reduced due to GPT-4V's low API cost; (ii) GPT-4V is one of the most competitive LMMs that can provide high quality candidates; (iii) the human filtering process ensures the correctness of the final vector graphics understanding benchmark.

Finally, we collect 4279 samples in total, as shown in Table 1 and 2. The word distribution of answers in the *VGQA* dataset is illustrated in Figure 4. Specifically, we have 2228, 1139, and 912 samples for SVG, TikZ, and Graphviz, respectively. After an LLM makes responses to the vector graphics questions, we compare the final responses with the ground-truth answers to compute accuracy. For LLMs with weaker instruction-following capabilities in producing easily parsable outputs, we use GPT-4 to determine their chosen option and then assess their accuracy.

Results Evaluation results of *VGQA* under GPT-4 (OpenAI, 2023b) are shown in Table 3. Several interesting findings arise from the results:

GPT-4 generally shows strong vector graphics understanding capability. In the zero-shot setting, GPT-4 shows non-trivial accuracy far beyond ran-

Prompting	SVG				TikZ				Graphviz			
	Category	Color	Usage	Avg	Concept	Counting	Relation	Avg	Domain	Layout	Relation	Avg
Zero-Shot	41.2	72.8	50.6	54.9	89.4	77.5	76.0	81.0	84.6	82.3	86.6	84.5
In-Context Learning	49.4	74.1	61.4	61.6	89.4	75.0	77.0	80.5	86.0	82.3	87.8	85.4
Chain of Thought	49.2	77.5	53.4	60.0	89.3	77.3	78.4	81.7	86.3	82.9	86.0	85.1

Table 3: Evaluation of *VGQA* across diverse vector graphics formats for GPT-4. It can be seen that GPT-4 performs better on higher-level semantics with TikZ and Graphviz than on lower-level SVGs. It can also be seen that using specific prompting techniques improves performance, especially with Chain of Thought prompting.

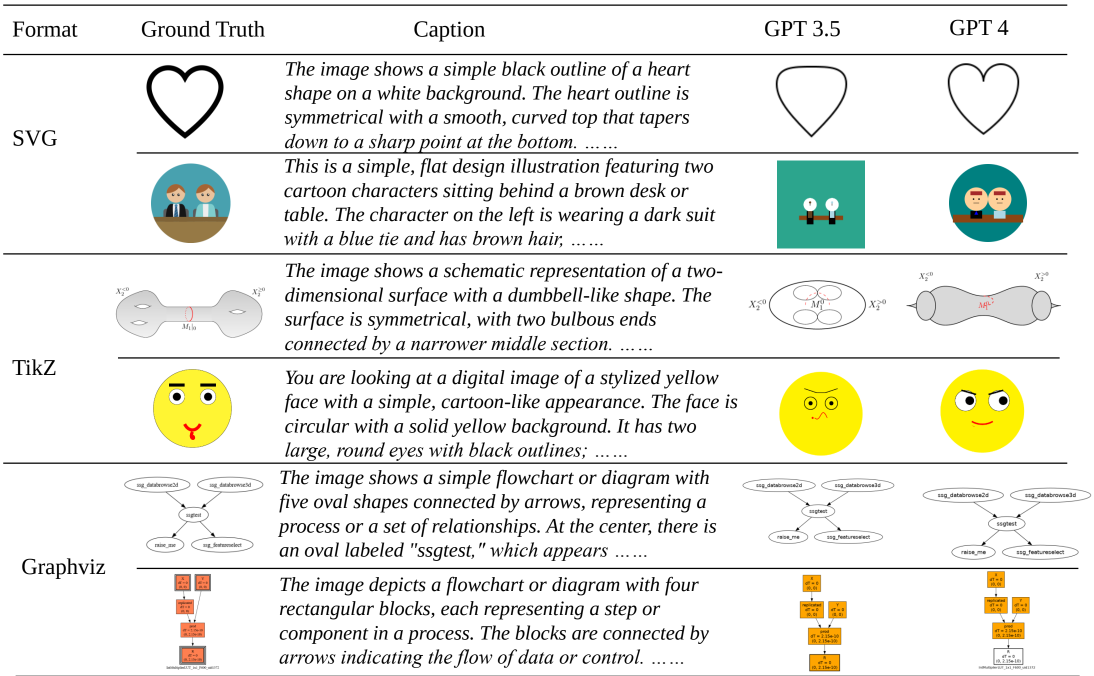


Figure 6: Examples of generated vector graphics. The ground truth images are rendered by vector graphics code directly from Internet. The captions are generated by GPT-4V, The images on the right side are rendered by vector graphics code generated by GPT-3.5 or GPT-4.

dom accuracy (25%) across all categories. Specifically, GPT-4 shows strong performance in TikZ, with an average accuracy of 78%.

GPT-4 shows stronger performance in high-level vector graphics language (e.g., TikZ, Graphviz) compared to low-level vector graphics language SVG. In either zero-shot, few-shot, or Chain-of-Thought settings, TikZ and Graphviz show at least 17% better performance than SVG. As a reminder, TikZ and Graphviz are fundamentally different from SVG in terms of the semantic levels, as SVG is composed of geometry primitives while TikZ and Graphviz contain high-level semantics such as "above", "below", explicit representation of nodes and edges, *etc*.

Chain of Thought (CoT) and In-Context Learning (ICL) show some performance improvements for some tasks, but not significant. CoT and ICL show ∼7% performance boost for SVG which owns lowest performance among three formats. Yet CoT and ICL show no benefits for TikZ and limited improvements for Graphviz, where GPT-4 already obtains ∼83% accuracy under TikZ and Graphviz.

Different vector-graphic formats show diverse behaviors upon question types. For SVG, GPT-4 struggles at high-level questions and receives ∼50% accuracy on category and reasoning types, while in TikZ and Graphviz, GPT-4 shows decent performance across all types of questions. This again demonstrates that GPT-4 shows inferior performance in low-level vector graphics tasks, especially on tasks related to reasoning.

3.4 *VGen*: Vector Graphics Generation Benchmark

Tasks We introduce *VGen*, a benchmark evaluating LLMs' vector-graphics generation capability. We use text to vector-graphics (T2VG) generation to test an LLM's ability to generate vector graphics code conditioned on a text prompt.

Benchmark Creation and Evaluation Again we evaluate on three vector graphics formats: SVG, TikZ and Graphviz. First, we obtain captions for each vector graphics image by leveraging GPT-4V (OpenAI, 2023b) over its rasterized image. Then we prompt the LLM to generate the vector graphics code corresponding to the caption.

	SVG	TikZ	Graphviz
# of VG-captions pairs	2000	2000	1845

Table 4: Statistics of *VGen* on three VG formats.

Finally, we map the generated vector graphics into rasterzied images, then use CLIP Score and Fréchet Inception Distance (FID) Score to evaluate the quality of the generated vector graphics.

We use CLIP Score to measure the similarity between each generated vector graphics and its associated caption. We utilize Long-CLIP (Zhang et al., 2024) instead of the vanilla CLIP (Radford et al., 2021) since our detailed captions are often longer than CLIP's maximum context length of 77. FID is utilized to evaluate the distribution gap between the original vector graphics and generated ones. For both metrics, we use the score of our ground truths as the upper bound to reflect the quality of the generated images. The overall pipeline is shown in Figure 5.

LLM	SVG	TikZ	Graphviz
Ground-Truth	25.61	24.63	23.67
GPT-4	23.97	24.42	24.50
GPT-3.5-Turbo	22.88	24.21	23.88

Table 5: CLIP score between captions and rasterized images from the generated vector graphics.

LLM	SVG	TikZ	Graphviz
GPT-4	44.81	39.38	77.03
GPT-3.5-Turbo	60.67	17.49	88.20

Table 6: The FID score between the ground truth images and the generated images. Lower is better.

Results CLIP and FID score under each VG type is shown in Table 5 and Table 6, respectively. Qualitative examples are shown in Figure 6.

Both GPT-3.5 and GPT-4 show strong vector graphics generation capability. Both LLMs show similar CLIP score as the ground truth. Results on FID score also support this claim. GPT-4 shows better performance than GPT-3.5 on CLIP score. Qualitative examples including the heart shape and flowchart generation also demonstrate the promising capability of VG generation using LLMs.

3.5 In Depth Analysis

Impact of Different LLMs We next perform experiments over a variety of large language models, including GPT-4, GPT-3.5, Llama-3-70B-Instruct (Meta, 2024) and Llama-3-8B-Instruct. Results are shown in Table 7. The results show that GPT-4 has the best VG understanding ability over vector graphics among those models while Llama-3-70B shows better performance than GPT-3.5.

Comparison between LLMs and MLLMs on Image Understanding We find that VLMs such as LLaVA (Liu et al., 2023) show interesting behavior compared with LLMs on vector graphics. To evaluate the performance of VLMs on *VGQA*, we render each visual content in our benchmark into the PNG format, and then feed the same question to VLMs. Specifically, we evaluate LLaVA-1.5-13b (Liu et al., 2024), as shown in Table 9. LLaVA-1.5 shows stronger performance in SVG format compared to TikZ and Graphviz. The strong performance gain that LLMs obtained on high-level vector graphics languages such as TikZ and Graphviz shows that those kinds of vector graphics are more aligned with LLMs' training data, natural languages, which is a highly compressed representation of the world. Low-level vector graphics languages such as SVG cover more low-level visual signals that can be better handled by VLMs using their rasterized representation.

LLM	SVG Avg	TikZ Avg	Graphviz Avg
GPT-4	54.9	**81.0**	**84.5**
GPT-4o(Text)	64.4	79.8	80.8
LLaVA-1.5-13b	**84.1**	47.8	50.1

Table 9: The comparison on image understanding ability between LLMs and VLMs, reflected by the average accuracy (%) on *VGQA*. We feed the rasterialized vector graphics in PNG format along with the same question in text format to LLaVA-1.5-13b to evaluate its performance on *VGQA*

Impact of Vector Graphics Sequence Length We next study the influence of the length of the vector graphics on vector graphics understanding.

Model	SVG				TikZ				Graphviz			
	Category	Color	Usage	Avg	Concept	Counting	Relation	Avg	Domain	Layout	Relation	Avg
Proprietary Large Language Models (LLMs): Vector Graphics as Input												
GPT-4o	52.5	80.4	60.3	64.4	87.0	75.0	77.3	79.8	83.6	75.0	83.7	80.8
GPT-4	41.2	72.8	50.6	54.9	89.4	77.5	76.0	81.0	84.6	82.3	86.6	84.5
GPT-3.5-Turbo	33.4	50.5	47.1	43.7	76.7	56.8	54.4	62.6	83.6	62.5	63.5	69.9
Gemini-1.5-Pro	39.2	73.2	47.9	53.4	86.7	74.9	71.8	77.8	79.5	66.8	86.0	77.4
Open-sourced Large Language Models (LLMs): Vector Graphics as Input												
Llama-3-8B	32.3	39.8	48.0	40.0	64.6	53.0	45.9	54.5	68.0	52.5	55.8	58.8
Llama-3-70B	46.3	58.7	55.3	53.4	78.5	68.2	66.7	71.1	72.8	61.4	74.4	69.5
Qwen2-7B	33.3	48.7	46.3	42.8	79.4	64.7	58.3	67.5	81.8	57.3	68.6	69.2
Qwen2-72B	43.4	62.4	55.9	53.9	88.6	74.6	72.5	78.6	86.5	71.5	80.8	79.6
Phi-3-Mini-128K	34.1	29.8	49.7	37.9	70.6	52.5	50.7	57.9	74.7	58.9	68.6	67.4
Phi-3-Medium-128k	43.6	44.7	60.6	49.6	80.4	59.7	62.8	67.6	81.4	66.5	72.7	73.5
Large Multimodal Models (LMMs): Rasterized Image as Input												
LLaVA-1.5-13b	83.0	85.2	84.0	84.1	64.3	34.3	44.8	47.8	46.7	53.9	49.7	50.1

Table 7: The evaluation of *VGQA* across diverse vector graphics formats for different LLMs and the evaluation of rasterized representation of *VGQA* in VLMs in the zero-shot setting.

Length	SVG				TikZ				Graphviz			
	Category	Color	Usage	Avg	Concept	Counting	Relation	Avg	Domain	Layout	Relation	Avg
1-1000	59.3	72.2	65.5	65.7	83.2	79.8	69.9	77.6	78.4	84.1	90.4	84.3
1000-2000	47.0	75.3	60.8	61.0	88.0	83.8	79.7	83.8	90.4	77.6	87.9	85.3
2000-3000	46.8	76.4	50.6	57.9	85.5	66.7	80.0	77.4	96.2	80.0	77.8	84.7
3000-4000	51.5	64.1	54.1	56.6	89.7	69.2	70.0	76.3	90.5	82.4	75.0	82.6
> 4000	48.9	70.1	52.5	57.2	95.8	55.0	81.2	77.3	87.0	82.5	72.2	80.6

Table 8: *VGQA* performance under different lengths of vector graphics for GPT-4 with zero-shot prompting. GPT-4 performs better on some lengths than others. For instance, in the Graphviz Domain question type, GPT-4 performs at an outstanding 96% accuracy on the 2k-3k range while showing most subpar performance on the <1k range.

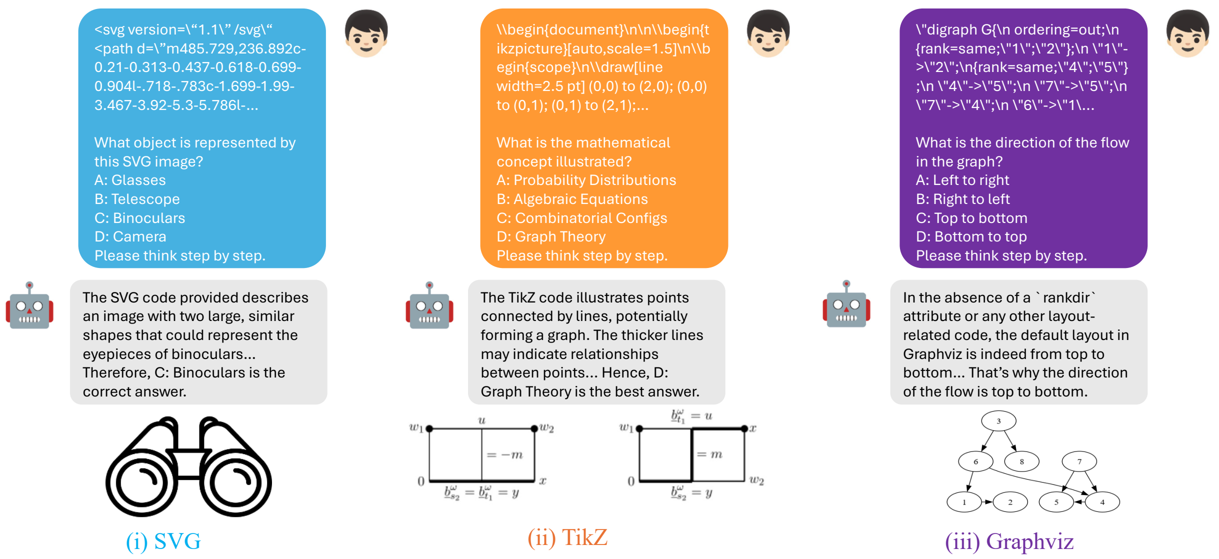


Figure 7: Examples of prompting GPT-4 using Chain-of-Thought with different types of vector graphics in *VGQA*. We only show GPT-4 the vector graphics code, but we include the rasterized images here for the sake of the readers.

Results for GPT-4 are shown in Table 8, where GPT-4 shows consistent performance across different length groups. Specifically, low-level vector graphics format such as SVG is most sensitive to the length. When the length increases, the understanding performance on SVG decreases steadily, while the understanding performance on other high level format remains stable. Another noticeable finding is that questions requiring complex reasoning, such as Usage in SVG or Relation in Graphviz, suffer more from the increasing sequence length.

Can LLMs Reason over Vector Graphics? The reasoning process of GPT-4 under the CoT setting is shown in Figure 7. Results show that GPT-4 can detect the key information over those samples, such as *"two large, similar shapes that could represent the eyepieces ..."*, for correct reasoning. We include the full conversation in Appendix 6.3.

4 Conclusion

Our study unveils new insights into the capabilities of LLMs in understanding and generating vector graphics. We discovered that LLMs demonstrate decent vector graphics understanding in TikZ, Graphviz, and SVGs, with a particular strength in understanding vector graphics code with higher-level semantics. We also found that LLMs often exhibit strong vector graphics generation capabilities. Interestingly, advanced prompting techniques can significantly improve performance for low-level formats such as SVG, and while GPT-4 had the strongest performance, open-source models like Llama-3-70B and Qwen2-72B show competitive performance. Our work lays a groundwork for future studies into LLMs' vector graphics understanding and generation benchmarking, and we hope it will inspire further efforts to enhance these capabilities. We will release our benchmark dataset and evaluation pipeline.

5 Limitations

We acknowledge that one cannot systematically evaluate the behavior of the closed-source models we employed, namely GPT-4, GPT-35-Turbo, and GPT-4V. Besides, more evaluations on recent LLMs can be conducted, which can provide more supporting experiments on LLMs' behavior on vector graphics understanding and generation.

Furthermore, recent works propose more prompting techniques such as Tree of Thoughts (ToT) (Yao et al., 2024) and Everything of Thoughts (XoT) (Ding et al., 2024). Incorporating these prompting techniques could further enhance our study.

Acknowledgements

This work was supported in part by IIS2404180, Microsoft Accelerate Foundation Models Research Program, and Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. 2022-0-00871, Development of AI Autonomy and Knowledge Enhancement for AI Agent Collaboration) and (No. RS-2022-00187238, Development of Large Korean Language Model Technology for Efficient Pre-training).

References

2024. Qwen2 technical report.

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Jonas Belouadi, Anne Lauscher, and Steffen Eger. 2024. Automatikz: Text-guided synthesis of scientific vector graphics with tikz. In *The Twelfth International Conference on Learning Representations*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020a. Language models are few-shot learners. In *NeurIPS*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020b. Language models are few-shot learners. *NeurIPS*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

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

Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. 2020. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems*, 33:16351–16361.

James Richard Diebel. 2008. *Bayesian Image Vectorization: the probabilistic inversion of vector image rasterization*. Stanford University.

Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2024. Everything of thoughts: Defying the law of penrose triangle for thought generation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 1638–1662, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Emden R Gansner. 2009. Drawing graphs with graphviz. *Technical report, AT&T Bell Laboratories, Murray, Tech. Rep, Tech. Rep.*

Nader Gharachorloo, Satish Gupta, Robert F Sproull, and Ivan E Sutherland. 1989. A characterization of ten rasterization techniques. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 355–368.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

David Ha and Douglas Eck. 2017. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. 2021. CLIPScore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7514–7528, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Xinyang Jiang, Lu Liu, Caihua Shan, Yifei Shen, Xuanyi Dong, and Dongsheng Li. 2021. Recognizing vector graphics without rasterization. *Advances in Neural Information Processing Systems*, 34:24569–24580.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pages 34892–34916. Curran Associates, Inc.

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

Andrew Mertz and William Slough. 2007. Graphics with tikz. *The PracTEX Journal*, 1:1–22.

Meta. 2024. Llama-3. `https://ai.meta.com/blog/meta-llama-3/`.

OpenAI. 2023a. Chatgpt. `https://openai.com/blog/chatgpt/`.

OpenAI. 2023b. Gpt-4 technical report.

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

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

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Juan A Rodriguez, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, David Vazquez, Christopher Pal, and Marco Pedersoli. 2023. Starvector: Generating scalable vector graphics code from images. *arXiv preprint arXiv:2312.11556*.

Zhenhailong Wang, Joy Hsu, Xingyao Wang, Kuan-Hao Huang, Manling Li, Jiajun Wu, and Heng Ji. 2024. Text-based reasoning about vector graphics. *Preprint*, arXiv:2404.06479.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

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

Tian Xia, Binbin Liao, and Yizhou Yu. 2009. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Transactions on Graphics (TOG)*, 28(5):1–10.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Beichen Zhang, Pan Zhang, Xiaoyi Dong, Yuhang Zang, and Jiaqi Wang. 2024. Long-clip: Unlocking the long-text capability of clip. *arXiv preprint arXiv:2403.15378*.

Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. 2023. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*.

6 Appendix

6.1 The specific prompt we used

6.1.1 Prompts used to build the dataset

Question Generation System prompt: The system prompts used to generate questions are different for different types of vector graphics and different types of questions. See the code in supplemental material for details.

User prompt: *The caption of this image is {caption}, generate the json according to the instruction. <IMAGE>*

Caption Generation System prompt: *Generate a detailed caption for the given image. The reader of your caption should be able to replicate this picture.*

User prompt: *<IMAGE>*

6.1.2 Prompts used to evaluate the models's understanding ability

Zero-shot System prompt: *I will present a {format} code. Please answer my questions only based on code. Answer and only answer the letter corresponding to the correct option. Do not add any additional comment in your response*

User prompt: *"{code}". Given this image, answer {question}. Options are {options}*

Few-shot System prompt: *I will present a {format} code. Please answer my questions only based on code. Answer and only answer the letter corresponding to the correct option. Do not add any additional comment in your response. For your reference, I will give you some examples.*

User prompt: *This is an example, the code is: {code}*

User prompt: *Given this image, answer {few_shot_sample_question}. Options are {few_shot_sample_options}*

Simulated assistant prompt: *{few_shot_sample_answer}*

Repeat the last three prompts for three times, each time pass a different samples.

User prompt: *"{code}". Given this image, answer {question}. Options are {options}*

Zero-shot-cot System prompt: *I will present a {format} code. Please answer my questions only based on code. Please consider the question step by step.*

User prompt: *{code}*

User prompt: *Given this image, the question is {question}. Options are {options}. Do not answer directly, consider each option individually.*

User prompt: *Carefully consider if the option A is correct*

Wait for the large language model's reponse and add its response to the context.

User prompt: *Carefully consider if the option B is correct*

Wait for the large language model's reponse and add its response to the context.

User prompt: *Carefully consider if the option C is correct*

Wait for the large language model's reponse and add its response to the context.

User prompt: *Carefully consider if the option D is correct*

Wait for the large language model's reponse and add its response to the context.

User prompt: *Which option is the best? Answer and only answer the letter corresponding to the correct option. Do not add any additional comment in your response*

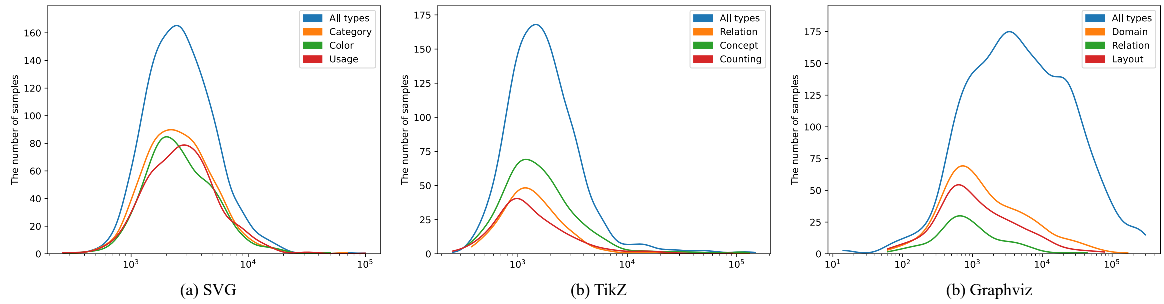


Figure 8: The distribution of vector graphics length in *VGBench*. X-axis denotes the string length of vector graphics files in each vector grapics format.

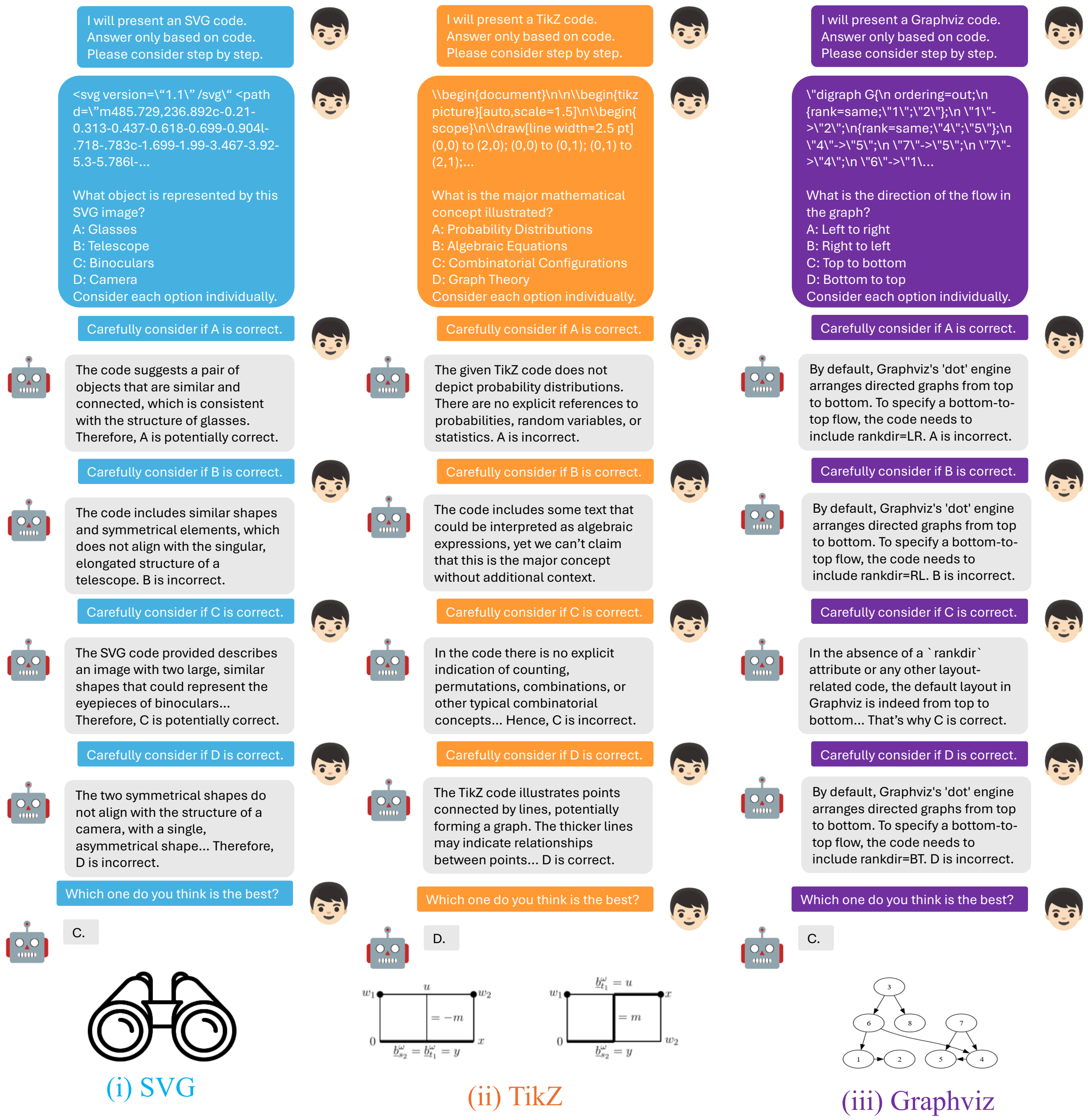


Figure 9: We include the full conversation with GPT-4 as indicated in Figure 7. We ask the model to consider if each option is correct individually, then ask another GPT-4 model to judge if the reasoning matches the correct answer.

6.1.3 Prompts used to evaluate models' generation ability

System prompt: *Generate a {format} based on the caption below. You should output the compilable code without any additional information.*

User prompt: *{caption}*

6.2 Data distribution

We include the distribution of *VGQA* grouped by each vector graphic category in Figure 8, each in itself grouped by the specific question categories we assigned.

6.3 Detailed examples for reasoning

We include the full version of the three example conversations previously put in Figure 7 now in Figure 9. The three conversations show how we only input the vector graphics code, exhibit the question, ask the model to consider each question carefully, and finally make its best choice.

6.4 Llama variants used in this paper

We evaluated Llama's variants, Llama-3-8B-Instruct-262k[3] and Llama-3-70B-Instruct-Gradient-262k[4] in this paper because they have extended context length.

6.5 Human filtering

The authors of this study, proficient in English with extensive research experience in vision-language learning, perform the vector graphics QA filtering.

6.6 Programs and Data Release

Our code and data is included in the supplementary materials.

[3]https://huggingface.co/gradientai/Llama-3-8B-Instruct-262k

[4]https://huggingface.co/gradientai/Llama-3-70B-Instruct-Gradient-262k