Plot2Code: A Comprehensive Benchmark for Evaluating Multi-modal Large Language Models in Code Generation from Scientific Plots

Chengyue Wu[1,2] **Zhixuan Liang**[1] **Yixiao Ge**[2*]
Qiushan Guo[1] **Zeyu Lu**[3] **Jiahao Wang**[1] **Ying Shan**[2] **Ping Luo**[1]

[1]The University of Hong Kong [2]ARC Lab, Tencent PCG
[3]Shanghai Jiao Tong University

https://github.com/TencentARC/Plot2Code

Abstract

Multi-modal Large Language Models have shown remarkable progress in visual contexts, yet their ability to convert visual figures into executable code remains underexplored. To address this, we introduce **Plot2Code**, a comprehensive benchmark designed to assess MLLMs' visual coding capabilities. Plot2Code includes 132 high-quality matplotlib plots across six plot types, as well as an additional 150 and 86 plots from Python's and R's plotly libraries respectively, totaling 368 plots. Each plot is paired with its source code and a descriptive instruction generated by GPT-4, enabling thorough evaluation across diverse inputs. Furthermore, we propose three automatic evaluation metrics—code pass rate, text-match ratio, and GPT-4V rating judgement—to assess the quality of generated code and rendered images. Notably, the GPT-4V rating demonstrates strong reliability, as it correlates well with human evaluations, particularly for datasets of a certain size. Cross-validation across MLLMs (GPT-4V, Gemini-1.5-Pro, and Claude-3-Opus) also shows high consistency in ratings, which likely stems from the fact that ratings are based on rendered images rather than direct MLLM outputs, indicating minimal bias for this metric. Our evaluation of 14 MLLMs, including both proprietary, and open-source models, highlights significant challenges in visual coding, particularly for text-dense plots, where MLLMs heavily rely on textual instructions. We believe these findings will advance future development of MLLMs.

1 Introduction

In the wake of significant advancements in big data and computational power, Large Language Models (LLMs) (Touvron et al., 2023; Brown et al., 2020; Hoffmann et al., 2022; Kaplan et al., 2020), such as ChatGPT (OpenAI, 2023a) and GPT-4 (OpenAI, 2023b), have become focal points of interest in both academic and commercial spheres. To extend their versatility across various contexts, Multi-modal Large Language Models (MLLMs) (Ge et al., 2024; Lu et al., 2024; OpenAI, 2023c) have rapidly evolved, as exemplified by the latest models such as GPT-4V (OpenAI, 2023c), Gemini (Gemini Team, 2023), Claude-3 (Anthropic, 2024), and the open-source models LLaVA (Liu et al., 2024a,b), Mini-GPT (Zhu et al., 2023; Chen et al., 2023a) and so on (Ge et al., 2024; Chen et al., 2023b). Concurrently, a diverse array of evaluation benchmarks (Li et al., 2023b,a; Yue et al., 2023; Ying et al., 2024) are curated to assess their visual comprehension performance across different domains. However, there remains a notable gap towards diagrams within text-dense images, which are crucial for assessing the multi-modal reasoning proficiency of MLLMs (Masry et al., 2022; Mathew et al., 2021). While Masry et al. (2024) proposes the benchmark evaluating chart understanding capabilities, it does not assess the ability of MLLMs to generate code that renders a provided plot, which is crucial for a full understanding of chart comprehension skills.

In line with Richard Feynman's philosophy, "What I cannot create, I do not understand," evaluating the capability of MLLMs to generate code that renders a provided chart effectively further showcases their multi-modal understanding and reasoning prowess. This particular challenge demands MLLMs to accurately interpret the visual elements present in input diagrams, correlate them with textual context provided, and finally derive executable code to generate the plots. Although the development of code generation from uni-modal natural language has experienced rapid progress in recent years (Roziere et al., 2023; Guo et al., 2024; Wu et al., 2024), the exploration of code generation using multi-modal inputs remains an active area of research. Previous efforts, *e.g.* HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), have concentrated on uni-modal code

*Correspondence to yixiaoge@tencent.com.

Findings of the Association for Computational Linguistics: NAACL 2025, pages 3006–3028

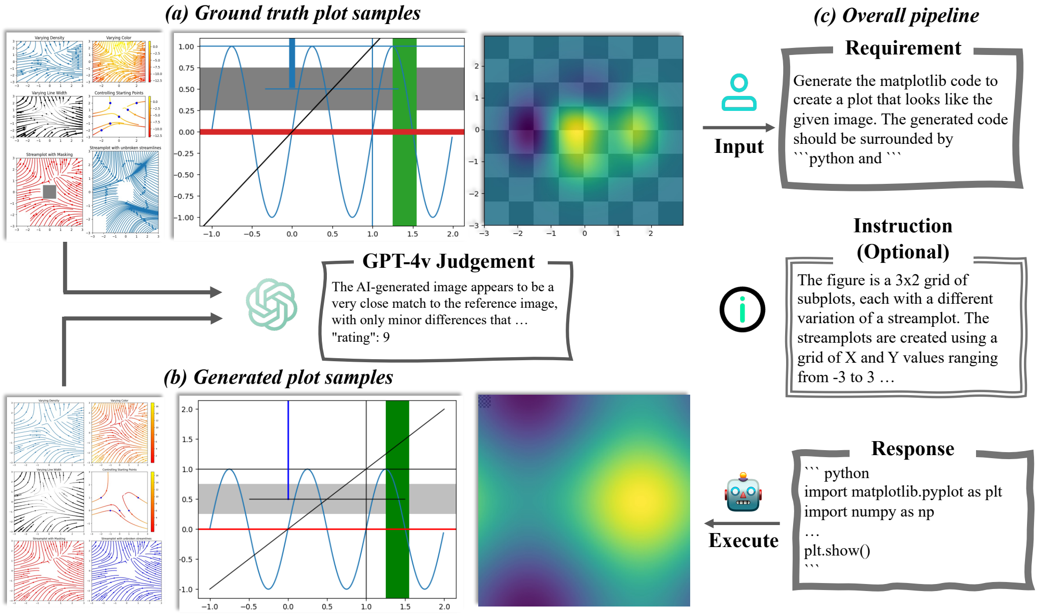


Figure 1: **Overview of Plot2Code.** ***Left:*** (a) Representative samples from the ground truth plots in our Plot2Code dataset. (b) Plot samples generated by multi-modal LLMs using the reference image. ***Right:*** (c) The comprehensive pipeline employed to assess the code generation ability of multi-modal LLMs. We consider two distinct settings: **Direct Asking** and **Conditional Asking**.

problems, while the more recent Design2Code (Si et al., 2024) has expanded the scope to include user-interface (UI) design, particularly HTML files, for evaluating MLLMs. However, these studies focus on unimodal scenarios (*e.g.*, text-only (Chen et al., 2021; Austin et al., 2021) or image-only (Si et al., 2024) inputs) and have limited capabilities when evaluating models with multimodal inputs.

To this end, our work underscores the critical motivation by addressing these three key challenges in evaluating MLLMs' coding capabilities: **i) Do the evaluation settings accommodate all modalities, including text and images, for both input and output?** This fundamental question pertains to the scope of visual coding. By employing extensive evaluation settings, we can conduct thorough ablation analyses of MLLMs' performance across various input modalities and their combinations, while also assessing outputs across different modalities. **ii) Are the evaluation metrics accurate, straightforward, and comprehensive?** Most existing code benchmarks rely on unit tests to obtain binary evaluation results. While this approach may suffice for uni-modal code tasks, it falls short for visual coding tasks that require not only the code pass rates but also the assessments of generated image fidelity. **iii) Are the evaluations for visual coding tasks relevant to real-world applications?** It is imperative that benchmarks align with real-world uses and applications, particularly in coding tasks. Employing the commonly used multiple-choice format for evaluating code tasks would be inadequate and incongruous.

Hence, in response to the aforementioned challenges, we present Plot2Code, a comprehensive and specialized multi-modal code benchmark crafted to evaluate the multi-modal understanding, reasoning, and coding capabilities of MLLMs. This benchmark comprises of 132 high-quality matplotlib plots across six plot types, alongside an additional 150 and 86 plots from Python's and R's plotly libraries, respectively, totaling 368 plots.[1] Each plot is paired with its corresponding code and a detailed description generated by GPT-4. To cater to diverse input and output formats, Plot2Code includes two evaluation settings, **Direct Asking** and **Conditional Asking**, supporting automatic metric-based evaluations for both text and image outputs. MLLMs can be evaluated using text, images, and a blend of both as inputs, while the text and image outputs can be assessed based on the code pass rate, text-match ratio and GPT-4V rating judgement, which consistently aligns with human evaluations and shows minimal bias across different advanced MLLMs.

We evaluate 14 publicly accessible MLLMs across various evaluation settings. Our findings

[1]The matplotlib and plotly libraries are released under BSD-compatible license (Hunter, 2007; Inc., 2015). We acknowledge the use and adhere to the licensing terms provided.

underscore the significant challenges posed by Plot2Code, with GPT-4V achieving an overall score of 7.68/10, indicating considerable room for enhancement in visual coding tasks. The contributions of this study can be summarized as follows:

- We construct a novel evaluation benchmark, tailored for multi-modal code tasks, enabling the assessment of advancements in multi-modal understanding and reasoning.
- Development of a diverse array of evaluation settings for Plot2Code, accommodating varied modalities for input and output through image-code pairs and automatic evaluation metrics.
- Evaluations of various publicly available MLLMs on Plot2Code, revealing that current MLLMs like GPT-4V, Gemini-Pro, and Claude-3, demonstrate modest performance in visual coding tasks.

We anticipate that Plot2Code will stimulate the research community to further explore and advance the realm of MLLMs, propelling us towards the realization of truly intelligent multi-modal systems.

2 Related Work

2.1 Advancements in Multi-modal Large Language Models

With the rapid progress of Large Language Models (LLMs) (OpenAI, 2023a; Touvron et al., 2023; OpenAI, 2023b), integrating multi-modal input into LLMs has gained significant interest (Liu et al., 2024a; Gemini Team, 2023; Lu et al., 2024; OpenAI, 2023c; Mu et al., 2024). Research focuses on developing encoders for processing multi-modal inputs via LLMs. Some studies (Haoran et al., 2023; Lu et al., 2024; Gemini Team, 2023) target text-dense images like documents and charts using high-resolution vision encoders. Our aim is to evaluate MLLMs' ability to generate code from reference plots, showcasing their visual coding skills.

2.2 Multi-modal Code Benchmark

Specialized models called Code LLMs (Roziere et al., 2023; Li et al., 2023c; Guo et al., 2024) focus on tasks like code completion and infilling, demonstrating reasoning abilities. Uni-modal benchmarks like HumanEval and MBPP (Chen et al., 2021; Austin et al., 2021) use unit tests and Pass@k metrics. Recent evaluations include multi-turn interactive settings (Wang et al., 2023c; Yang et al., 2024). MMCode (Li et al., 2024) integrates images into code tasks, while Design2Code (Si et al., 2024) uses CLIP scores for HTML generation. Work like (Rodriguez et al., 2023) explores extracting SVG code from images. Plot2Code offers diverse evaluation scenarios with uni-modal and multi-modal inputs, using metrics like code pass rate and plot similarity to assess MLLMs' reasoning capabilities. See Table 2 for details.

3 Dataset Collection

In this section, we outline the process of curating and processing our benchmark data. We began by crawling every website link listed in the gallery of Python's matplotlib, Python's plotly and R's plotly, and extracting the code block.

3.1 Test Set Curation

Our goal was to acquire plot-code pairs that effectively evaluate MLLM's code generation capabilities. Since the initial Python code may not always generate high-quality plots, we used both automatic processing and manual filtering.

Generation Filtering. We found some HTML files contained multiple code segments focusing on imports and initializations, which didn't produce plots. Thus, we extracted single code block from each HTML file that's directly for rendering plots.

Type Filtering. We assumed plots were simple and static figures rendered by the matplotlib engine, excluding animations and interactive plots. We filtered out plots tagged with animation, widget, and event handling.

Manual Curation. After processing, we manually curated examples based on these criteria: (1) Plots are free of external file dependencies and can be directly rendered. (2) Plots exhibit diversity in size, text, colors, and types, providing a comprehensive evaluation benchmark. (3) Plots are distributed across various difficulty levels, from beginner to specialized. This stringent manual filtering resulted in 368 high-quality test examples for our benchmark.

3.2 Evaluation Setting

We assess the test set under two distinct evaluation scenarios: direct asking and conditional asking. To facilitate convenient extraction of code from the MLLM-generated responses, we request the code to be enclosed between specific markers, enabling the use of regular expressions for extraction.

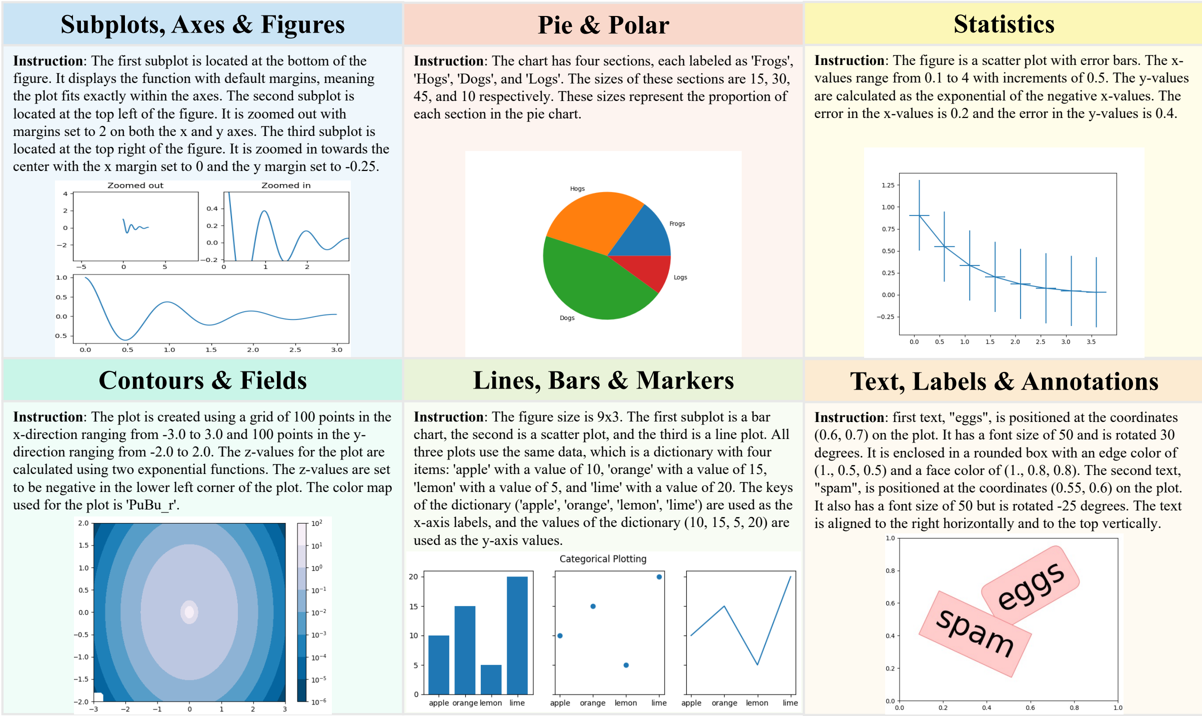


Figure 2: **Examples of Plot2Code benchmark.** We show different-type plots with instructions.

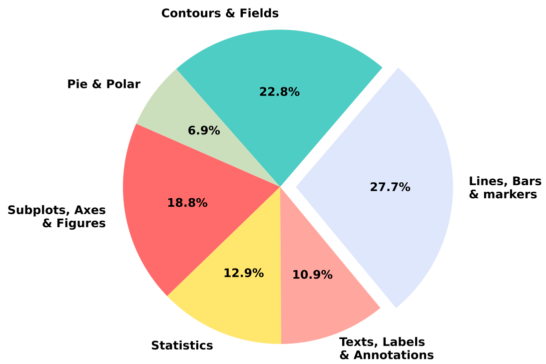


Figure 3: **Type Distribution of Plot2Code.**

Direct Asking. This setting means giving a MLLM an image as input and requiring it to generate executable code that produces a diagram closely resembling the input image. The specific prompt can be found in Appendix A.1. Figure 10 illustrates an example in this case.

Conditional Asking. For MLLMs, this setup involves receiving an image and text instructions to generate executable code that meets the specified conditions. For LLMs, only the text instructions are provided. GPT-4 extracts these instructions from the ground truth code, retaining essential information for reproduction without revealing the code itself. The prompt used to construct these instructions can be found in Appendix A.2. Figure 11 illustrates an example in this case.

3.3 Data Statistics

Key Statistics. Table 1 presents key statistics to gauge the difficulty, where we use `matplotlib.pyplot` subset for illustration. Our 132 test samples contain a total of 293 subplots (min=1, max=18). Tokenizing the scraped code with the LLaMA-2 tokenizer (Touvron et al., 2023), we found an average of 409 tokens per code file (standard deviation = 291), and 242 tokens per instruction (standard deviation = 58). Using PaddleOCR[2], we calculated an average of 23 text elements per plot (standard deviation = 13), highlighting the complexity of our benchmark.

Type Distribution. Figure 3 shows the type distribution of plots, classified by tags from the matplotlib gallery. The most common types are lines, bars, and markers, while others include contours, fields, pie charts, polar plots, subplots axes, statistical representations, and text annotations.

3.4 Evaluation Metrics

Unlike uni-modal code generation tasks assessed using straightforward metrics like unit tests and code pass rate, multi-modal tasks require more precise and comprehensive evaluation methods. We propose code pass rate, text match ratio, and GPT-4V judgement score for this purpose.

[2] https://github.com/PaddlePaddle/PaddleOCR

Statistic	Number
Total Samples	132
- Contours & Fields	30 (22.7%)
- Lines, Bars & Markers	37 (28.0%)
- Texts, Labels & Annotations	14 (10.6%)
- Statistics	17 (12.9%)
- Subplots, Axes & Figures	25 (18.9%)
- Pie & Polar	9 (6.8%)
Total Subplot Count	293
Code Length (tokens)	401 ± 281
- Minimum Length	60
- Maximum Length	1823
Instruction Length (tokens)	279 ± 115
- Minimum Length	72
- Maximum Length	628
Text Count	23 ± 13

Table 1: **Key Statistics of Plot2Code.** Tokens are counted by LLaMA-2 tokenizer.

Code Pass Rate. This metric checks if the MLLM-generated code can render an image using the specified plotting library, *i.e.*, Python's `matplotlib`, Python's `plotly` and R's `plotly`, ensuring the code is executable.

GPT-4V Rating Judgement. We use state-of-the-art MLLM *i.e.* GPT-4V to assess the high-level similarity between generated plots and ground truth plots. This evaluation involves providing both images to the models with a detailed prompt asking them to consider factors including overall appearance, colors, shapes, positions, and other visual elements. The models then rate the similarity on a scale of 1 to 10. The specific prompt used is shown in Appendix A.3. As illustrated in Sec. 5, the GPT-4V overall rating shows a strong correlation with human ratings on datasets exceeding 60 samples. Besides, cross-validation among multiple MLLMs (GPT-4V, Gemini-1.5-Pro, and Claude-3-Opus) also shows high consistency in ratings, indicating minimal bias of this metric. This comprehensive approach ensures a detailed and objective assessment of visual similarity between plots.

Text-Match Ratio. While high-level evaluation provided by the GPT-4V rating is valuable, it misses detailed plot components like text. To address this, we introduce text-match ratio to measure the fine-grained similarity between text elements in generated and reference plots by considering both textual content and spatial positions. Concretely, we first align the size and positions between two diagrams. Then, text elements are extracted by OCR. After that, we compute the absolute distance between positions of texts and convert it to similarity. Finally, texts are considered matched if the content is matched and the similarity exceeds 0.8. Details are shown in Algorithm 1. This ratio indicates the accuracy of text reproduction in generated plots.

3.5 Comparison with Other Datasets

As depicted in Table 2, our dataset encompasses the most extensive range of evaluation settings and metrics compared to all other uni-modal and multi-modal code benchmarks.

4 Experiments

In this section, we evaluate a variety of multi-modal large language models and methods on our Plot2Code benchmark to compare their performance, including both closed-source commercial models and state-of-the-art open-source models.

4.1 Evaluation Details

Evaluated (M)LLMs. To ensure a comprehensive evaluation, we assess 14 representative closed-source and open-source (M)LLMs that vary in parameters, resolution settings, and backbone LLMs, such as GPT (OpenAI, 2023a), DeepSeek (Bi et al., 2024), Mistral (Jiang et al., 2023), Mixtral (Jiang et al., 2024), and Yi (Young et al., 2024). The quantitative evaluation is provided in Sec. 4.2. We also explore different prompt strategies, including Chain-of-Thought (Wei et al., 2022a) and Plan-and-Solve (Wang et al., 2023a). We investigate the influence of different designs of MLLMs on the performance of our benchmark in Sec. 4.3.

Evaluation Methods. As mentioned in Sec. 3.2, we employ two distinct evaluation settings: Direct Asking and Conditional Asking. For LLMs lacking vision capabilities, we evaluate them solely in the Conditional Asking setting with instruction input. Furthermore, we extend the GPT-4V judgement setting to conduct pairwise evaluations between two (M)LLMs and perform a correlative analysis between GPT-4V judgement and human evaluation. More details are provided in Sec. 4.4.

4.2 Overall Evaluation

We showcase the quantitative results of (M)LLMs on our Plot2Code benchmark here. The code pass rate, text-match ratio, and GPT-4V overall rating for both direct asking and conditional asking scenarios are reported in Table 3.

The Comprehensive Challenge of Plot2Code. The benchmark poses considerable challenges,

Dataset	Task Type	Input Format		Output Eval Format			Evaluation Metric		
		T	I+T	I	T	I+T	Pass Rate	Component Match	Rating
HumanEval (Chen et al., 2021)	Programming	✔	✘	✘	✔	✘	✔	✘	✘
SVGEditBench (Nishina and Matsui, 2024)	SVG	✔	✘	✔	✘	✘	✘	✘	✘
MMcode (Li et al., 2024)	Algorithm	✔	✔	✘	✔	✘	✔	✘	✘
Design2Code (Si et al., 2024)	Websites	✘	✔	✔	✘	✘	✘	✔	✔
Plot2Code	Plots	✔	✔	✔	✔	✔	✔	✔	✔

Table 2: **Comparison with other uni-modal and multi-modal code benchmarks.**"I" represents images,"T" represents text, and "I+T" stands for the multi-modal information with images and text.

Model	Backbone LLM	Direct Asking			Conditional Asking		
		Pass Rate	Text-Match	Rating	Pass Rate	Text-Match	Rating
LLMs							
ChatGPT (OpenAI, 2023a)	ChatGPT (OpenAI, 2023a)	-	-	-	80.3	56.7	6.59
GPT-4 (OpenAI, 2023b)	GPT-4 (OpenAI, 2023b)	-	-	-	80.3	68.0	7.36
GPT-4 (CoT) (OpenAI, 2023b)	GPT-4 (OpenAI, 2023b)	-	-	-	78.8	66.0	7.09
GPT-4 (PS+) (OpenAI, 2023b)	GPT-4 (OpenAI, 2023b)	-	-	-	77.3	66.8	7.26
Closed-source MLLMs							
Claude-3-Opus (Anthropic, 2024)	Claude-3 (Anthropic, 2024)	84.1	57.5	4.37	78.0	69.7	7.68
Claude-3-Sonnet (Anthropic, 2024)	Claude-3 (Anthropic, 2024)	75.8	46.7	5.38	65.9	57.0	7.20
Gemini-Pro (Gemini Team, 2023)	Gemini (Gemini Team, 2023)	68.2	53.6	5.06	55.3	66.9	7.10
GPT-4V (OpenAI, 2023c)	GPT-4 (OpenAI, 2023b)	84.1	57.7	**6.48**	81.8	70.7	7.68
GPT-4V (CoT) (OpenAI, 2023c)	GPT-4 (OpenAI, 2023b)	89.4	56.3	6.30	81.8	69.7	7.75
GPT-4V (PS+) (OpenAI, 2023c)	GPT-4 (OpenAI, 2023b)	86.4	55.3	6.25	85.6	71.4	**7.83**
Open-source MLLMs (Low resolution setting)							
Mini-Gemini-2B (Li et al., 2023d)	Gemma-2B (Team et al., 2024)	39.4	21.4	1.96	22.7	31.8	2.80
Mini-Gemini-8x7B (Li et al., 2023d)	Mixtral-8x7B (Jiang et al., 2024)	75.8	33.9	3.76	62.1	52.3	5.74
Mini-Gemini-34B (Li et al., 2023d)	Yi-34B (Young et al., 2024)	67.4	30.5	2.78	50.0	51.2	4.79
Open-source MLLMs (High resolution setting)							
DeepSeek-VL-7B (Lu et al., 2024)	DeepSeek-7B (Bi et al., 2024)	72.0	38.7	3.69	56.8	50.1	5.19
LLaVA-1.6-Mistral-7B (Liu et al., 2024a)	Mistral-7B (Jiang et al., 2023)	64.4	32.6	3.06	42.4	45.1	4.48
LLaVA-1.6-34B (Liu et al., 2024a)	Yi-34B (Young et al., 2024)	72.0	34.6	3.18	53.0	50.7	5.60
Mini-Gemini-8x7B-HD (Li et al., 2023d)	Mixtral-8x7B (Jiang et al., 2024)	73.5	40.7	3.87	58.4	53.7	6.08
Mini-Gemini-34B-HD (Li et al., 2023d)	Yi-34B (Young et al., 2024)	55.8	34.0	3.06	43.4	46.1	5.35

Table 3: **Quantitative results for 14 MLLMs across two settings, Direct Asking and Conditional Asking.** The maximum value of GPT-4V overall rating is **bolded**.

as even advanced models like Claude-3-Opus, Gemini-Pro, and GPT-4V achieve only 7.68, 7.10, and 7.68, respectively, in the overall assessment for the conditional asking scenario, indicating substantial room for improvement. In addition to the overall rating, the pass rate also presents challenges for MLLMs, particularly when instructions are added. For example, Gemini-Pro's pass rate decreases from 68.2% to 55.3% after incorporating the instruction, as the added requirements will make it harder to generate the corresponding code. In contrast to widely used benchmarks like MT-bench and HumanEval, where recent advanced models attain ratings above 9.00 and code pass rates exceeding 80%, Plot2Code necessitates both visual understanding and reasoning abilities to analyze the plot, generate executable code, and create a plot resembling the reference plot. This heightened challenge for (M)LLMs serves as a rigorous examination of visual reasoning and coding capabilities.

Gap between Closed-source and Open-source Models. Open-source models significantly lag behind closed-source models. We evaluated advanced open-source MLLMs, such as DeepSeek-VL (Lu et al., 2024), Mini-Gemini (Gemini Team, 2023), and LLaVA-Next (Liu et al., 2024a). The best-performing open-source model, Mini-Gemini-8x7B-HD, achieved a 6.08 GPT-4V judgement score and a 58.4% code pass rate. However, this still falls short compared to commercial closed-source MLLMs. There is a need for the open-source community to develop more powerful models to compete with proprietary ones.

4.3 Influence of Different Settings

We analyze the results from various perspectives, encompassing prompt strategies, backbone LLMs, and the resolution settings. The key findings are summarized as follows.

The Influence of LLMs. As depicted in Tab. 3, there is a strong correlation between model performance and the backbone LLM used, evident in both Mini-Gemini and LLaVA. This suggests that the Plot2Code task may require powerful backbone LLMs to facilitate the reasoning process and generate executable code.

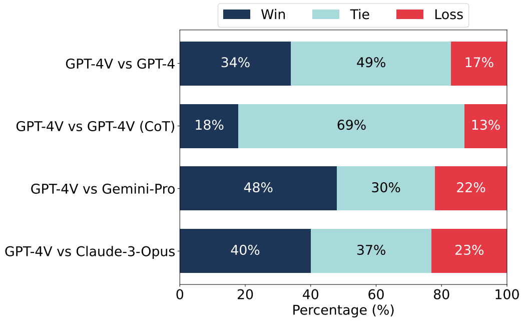


Figure 4: **Pair evaluation results in the conditional asking setting.** We use GPT-4V without prompt strategies as the baseline (this method is not shown in the table as it serves as the basis for pairwise comparison).

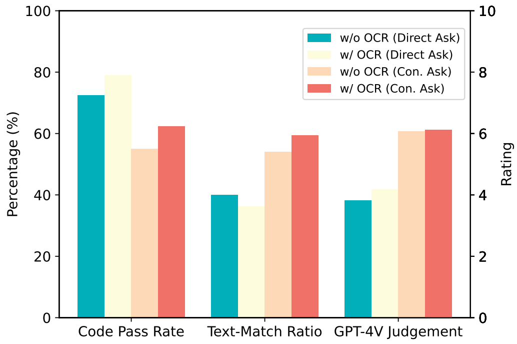


Figure 5: **Ablation experiments involving the addition of OCR tokens or not.** The base model is Mini-Gemini-8x7B-HD. OCR tokens are extracted using PaddleOCR, supported by Mini-Gemini's official codebase.

Evaluation Settings. As discussed in Sec. 3.2, we have two evaluation settings. Table 3 shows that in the conditional asking setting, MLLMs generally achieve a lower pass rate but higher similarity than in the direct asking setting. This is likely because the added instruction imposes stricter requirements, making it harder to generate executable code, but enhancing image similarity to the reference.

Apply Prompt Techniques. We also explored different prompt strategies like Chain-of-Thought (Wei et al., 2022b) and Plan-and-Solve (Wang et al., 2023b) in Fig. 4, finding no clear advantage over default prompt, indicating ongoing exploration in multi-modal reasoning prompts.

Nuance Evaluation across Plot Types. We rate GPT-4V, Gemini-1.5-Pro, and Claude-3-Opus, for different plot types. The results are shown in Figure 6. We find that the model is relatively balanced in all types. The stronger model generally performs better across all types.

Image Resolution Settings. We examined vision encoder settings focusing on image resolution. Higher resolution encoders provide more detailed

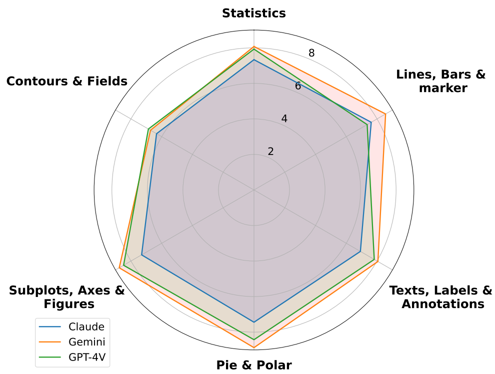


Figure 6: Radar chart comparing the average scores of different chart types for three models: Claude, Gemini, and GPT-4V. Each axis represents a different chart type.

information from input images, and we found that MLLMs with higher resolution consistently performed better—similar to trends seen in ChartQA (Masry et al., 2022) and DocQA (Mathew et al., 2021). Additionally, adding OCR tokens, as shown in Figure 5, improved performance similar to high-resolution settings, suggesting that current MLLMs may need more powerful vision encoders to capture detailed image information.

4.4 Pairwise Model Comparison

In accordance with the conventional practice of pairwise model evaluation (Zheng et al., 2024; Zhou et al., 2024), we expand the GPT-4V judgement setting to perform pairwise evaluations between two (M)LLMs. The detailed prompt employed for pairwise model comparison can be found in the Appendix A.3. For each reference sample, we request GPT-4V to determine which generated image is more similar when comparing a pair of MLLMs. To mitigate the influence of differing positions, we swap the two generated images for an additional evaluation. A model is considered victorious only if it wins both rounds; otherwise, the result is deemed a tie. We utilize GPT-4V as the baseline for comparison. The results are illustrated in Figure 4, from which we can infer that: (i) Compared to GPT-4, the inclusion of image input for GPT-4V is beneficial in generating higher quality plots. (ii) The commonly used prompt strategy, Chain-of-Thought, does not yield additional advantages in our benchmark.

5 Statistical Analysis

In this section, we perform statistical analyses to justify our benchmark design. We first investigate the effectiveness of our proposed metrics in indicat-

Indicators	t-statistic	p-value	Reject H_0
MSE	-1.24	0.22	✗
SSIM	1.28	0.21	✗
CLIP-Score	4.23	1.24×10^{-4}	✔
Text-Match Ratio	5.69	9.62×10^{-7}	✔
GPT-4V Judgement	**9.07**	1.22×10^{-11}	✔

Table 4: **Comparison of indicators in distinguishing image groups w/ and w/o significant differences.**

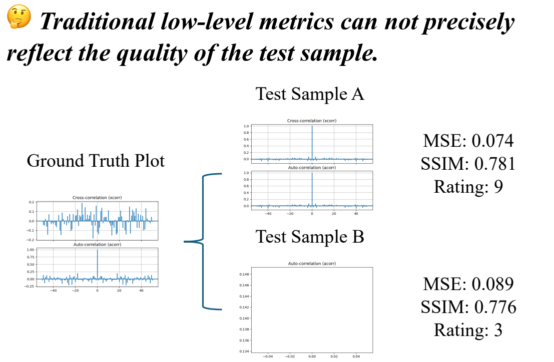


Figure 7: **Inaccuracy of traditional low-level metrics.**

ing image similarity. Additionally, we analyze the correlation between GPT-4V judgement and human evaluation to substantiate our metric's validity.

Hypothesis Tests for Image Similarity Metrics. To evaluate indicators' ability to distinguish images with different visual similarities, we conducted hypothesis tests comparing two groups: one with high similarity (generated by GPT-4V) and another with lower similarity (from Mini-Gemini-2B). Using a two-sample t-test, we calculate:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

where $\bar{X}_1$, $\bar{X}_2$ are sample means, s_1^2, s_2^2 are variances, and n_1, n_2 are sample sizes. With a p-value less than 0.05, we reject the null hypothesis (H_0) in favor of a significant difference (alternative hypothesis, H_1). Results showed MSE and SSIM p-values of 0.22 and 0.21, respectively, not rejecting H_0. However, GPT-4V Judgement and Text-Match Ratio had p-values of 1.22×10^{-11} and 9.62×10^{-7}, indicating significant differences, outperforming the CLIP-score with a p-value of 1.24×10^{-4}.

Correlative Analysis between GPT-4V Judgement and Human Evaluation. To investigate the similarity between the GPT-4V judgement and human evaluation, a correlative analysis was performed using different correlation coefficients, including Kendall's Tau, Pearson correlation coefficient, and Spearman's rank correlation coefficient. Details can be found in the Appendix C. As shown in Table 5, All three correlation coefficients indicated a moderate positive relationship between the GPT-4V judgement and human evaluation. Moreover, the p-values were all smaller than the significance level of 0.05, suggesting that the correlations were statistically significant. These findings imply that the GPT-4V judgement is in general agreement with human evaluation, demonstrating its effectiveness in assessing the similarity.

Robustness Analysis upon Sample Size. We randomly sample entries to evaluate the relationship between the score variance and the dataset size. In Figure 8, our experiments show that the score does not vary much once the dataset size exceeds 60, which demonstrates that our dataset is enough to assess the ability to generate plots reliably. Additionally, the absolute ratings from humans and GPT-4V are highly correlated when the dataset size exceeds 60, with average ratings of 7.17 and 7.71 for humans and GPT-4V, respectively.

Potential Bias with AI Rating. To investigate potential biases in each business MLLM, we conducted a cross-validation using GPT-4V, Gemini-1.5-Pro, and Claude-3-Opus to generate plots and rate each other. The results, presented in the Figure 9, indicating high consistency among the MLLM judges. The Cronbach's Alpha values for GPT-4V, Gemini, and Claude samples are 0.77, 0.84, and 0.82, respectively, suggesting minimal bias when MLLMs evaluate their own generated samples.

6 Conclusion

In this study, we introduced Plot2Code, a benchmark for assessing multi-modal language models' code generation capabilities. We proposed evaluation metrics like code pass rate, text-match ratio, and GPT-4V rating judgement to provide a holistic assessment of model performance. The study found notable performance differences among models, highlighting challenges in reproducing text elements and details. Plot2Code aims to advance multi-modal reasoning, text-dense image understanding, and code generation in MLLMs. Future research may explore multi-modal prompts and vision encoder design to bridge the gap between open-source MLLMs and commercial APIs.

Limitations: This benchmark suffers from potential contamination, as pre-trained models might use the data as part of its training data. Additionally, while the dataset is comprehensive on code generation for plotting, it may not fully capture the complexity and variety of other types of programming tasks beyond chart generation.

Ethical considerations: Reliance on automated code generation may erode human coding skills, and increased dependence on AI reduce oversight, leading to errors in critical systems.

References

Anthropic. 2024. Claude 3 haiku: our fastest model yet. Available at: https://www.anthropic.com/news/claude-3-haiku.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. arXiv preprint arXiv:2108.07732.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. arXiv preprint arXiv:2401.02954.

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

Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. 2023a. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. arXiv preprint arXiv:2310.09478.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Zhong Muyan, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. 2023b. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. arXiv preprint arXiv:2312.14238.

Yuying Ge, Sijie Zhao, Jinguo Zhu, Yixiao Ge, Kun Yi, Lin Song, Chen Li, Xiaohan Ding, and Ying Shan. 2024. Seed-x: Multimodal models with unified multi-granularity comprehension and generation. arXiv preprint arXiv:2404.14396.

Google Gemini Team. 2023. Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y Wu, YK Li, et al. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. arXiv preprint arXiv:2401.14196.

Wei Haoran, Kong Lingyu, Chen Jinyue, Zhao Liang, Ge Zheng, Yang Jinrong, Sun Jianjian, Han Chunrui, and Zhang Xiangyu. 2023. Vary: Scaling up the vision vocabulary for large vision-language models. arXiv preprint arXiv:2312.06109.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. arXiv preprint arXiv:2203.15556.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. Computing in Science & Engineering, 9(3):90–95.

Plotly Technologies Inc. 2015. Collaborative data science.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. arXiv preprint arXiv:2310.06825.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. arXiv preprint arXiv:2401.04088.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361.

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. 2023a. Seed-bench-2: Benchmarking multimodal large language models. arXiv preprint arXiv:2311.17092.

Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. 2023b. Seed-bench: Benchmarking multimodal llms with generative comprehension. arXiv preprint arXiv:2307.16125.

Kaixin Li, Yuchen Tian, Qisheng Hu, Ziyang Luo, and Jing Ma. 2024. Mmcode: Evaluating multi-modal code large language models with visually rich programming problems. arXiv preprint arXiv:2404.09486.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023c. Starcoder: may the source be with you! arXiv preprint arXiv:2305.06161.

Yanwei Li, Yuechen Zhang, Chengyao Wang, Zhisheng Zhong, Yixin Chen, Ruihang Chu, Shaoteng Liu, and Jiaya Jia. 2023d. Mini-gemini: Mining the potential of multi-modality vision language models. arXiv:2403.18814.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024a. Llava-next: Improved reasoning, ocr, and world knowledge.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024b. Visual instruction tuning. Advances in neural information processing systems, 36.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, Yaofeng Sun, Chengqi Deng, Hanwei Xu, Zhenda Xie, and Chong Ruan. 2024. Deepseek-vl: Towards real-world vision-language understanding. Preprint, arXiv:2403.05525.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. arXiv preprint arXiv:2203.10244.

Ahmed Masry, Mehrad Shahmohammadi, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. 2024. Chartinstruct: Instruction tuning for chart comprehension and reasoning. arXiv preprint arXiv:2403.09028.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In Proceedings of the IEEE/CVF winter conference on applications of computer vision, pages 2200–2209.

Yao Mu, Junting Chen, Qing-Long Zhang, Shoufa Chen, Qiaojun Yu, GE Chongjian, Runjian Chen, Zhixuan Liang, Mengkang Hu, Chaofan Tao, et al. 2024. Robocodex: Multimodal code generation for robotic behavior synthesis. In Forty-first International Conference on Machine Learning.

Kunato Nishina and Yusuke Matsui. 2024. Svgeditbench: A benchmark dataset for quantitative assessment of llm's svg editing capabilities. arXiv preprint arXiv:2404.13710.

OpenAI. 2023a. Chatgpt. https://chat.openai.com.

OpenAI. 2023b. Gpt-4 technical report. ArXiv, abs/2303.08774.

OpenAI. 2023c. GPT-4V(ision) system card.

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.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. arXiv preprint arXiv:2308.12950.

Chenglei Si, Yanzhe Zhang, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. 2024. Design2code: How far are we from automating front-end engineering? arXiv preprint arXiv:2403.03163.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. arXiv preprint arXiv:2403.08295.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. arXiv preprint arXiv:2305.04091.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023b. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. arXiv preprint arXiv:2305.04091.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2023c. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. arXiv preprint arXiv:2309.10691.

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

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

Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ping Luo, and Ying Shan. 2024. Llama pro: Progressive llama with block expansion. arXiv preprint arXiv:2401.02415.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. 2024. Intercode: Standardizing and benchmarking interactive coding with execution feedback. Advances in Neural Information Processing Systems, 36.

Kaining Ying, Fanqing Meng, Jin Wang, Zhiqian Li, Han Lin, Yue Yang, Hao Zhang, Wenbo Zhang, Yuqi Lin, Shuo Liu, et al. 2024. Mmt-bench: A comprehensive multimodal benchmark for evaluating large vision-language models towards multitask agi. arXiv preprint arXiv:2404.16006.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. arXiv preprint arXiv:2403.04652.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. 2023. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. arXiv preprint arXiv:2311.16502.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. Advances in Neural Information Processing Systems, 36.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024. Lima: Less is more for alignment. Advances in Neural Information Processing Systems, 36.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. arXiv preprint arXiv:2304.10592.

A Prompt Template

In this section, we introduce the prompt template used for the experiment.

A.1 Prompt for Code Generation

We use the following for the direct asking setting.

> You are a helpful assistant that can generate Python code using matplotlib. Generate the matplotlib code to create a plot that looks like the given image, as similar as possible. The generated code should be surrounded by ```python and ```
>
> *<image_token><image_token><image_token>...*

For the conditional asking setting, we add the instruction of the reference plot at the front of the direct asking prompt.

> *<instruction>*
>
> You are a helpful assistant that can generate Python code using matplotlib. Generate the matplotlib code to create a plot that looks like the given image, as similar as possible. The generated code should be surrounded by ```python and ```
>
> *<image_token><image_token><image_token>...*

A.2 Prompt for Instruction Generation

Here is the prompt for generating each plot's corresponding instruction. We requite the GPT-4 to examine the code for each plot and summarize the key information in it without any implementation details.

> Please review the Python code provided below, which uses matplotlib.pyplot to generate figures. Your job is to identify key details, like type, texts, etc., required to recreate a figure from the given code: *<code>*
>
> Remember, your response should not include any code and avoid implementation details. Do not describe any detailed variables or functions in the code. Instead, use everyday language to describe the necessary information. If the code uses random seed, you should extract it for reproduction. Reveal the data used in the figure for recreation. Strictly follow the rule that do not expose any variables or functions used in the code. Summarize the crucial information as follows:

A.3 Prompt for Evaluation

We use the following prompt for GPT-4V overall rating. We will provide both the ground truth image and the test image generated by the MLLM assistant for GPT-4V to rate the similarity.

You are a helpful assistant. Please evaluate the similarity between a reference image created using matplotlib and an image generated by code provided by an AI assistant. Consider factors such as the overall appearance, colors, shapes, positions, and other visual elements of the images. Begin your evaluation by providing a short explanation. Be as objective as possible. After providing your explanation, you must rate the response on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]",

<gt_image_token><gt_image_token><gt_image_token>...

<test_image_token><test_image_token><test_image_token>...

In the pair evaluation, we utilize the following prompt to determine which generated image, either from Assistant A or Assistant B, is more similar to the ground truth image.

You are a helpful assistant. Please act as an impartial judge and evaluate the quality of the generated images provided by two AI assistants given the ground truth image displayed below. You should choose the assistant that generate the more similar image. Your evaluation should consider factors such as the overall appearance, colors, shapes, positions, and other visual elements of the images.

Here is the ground truth image.

<gt_image_token><gt_image_token><gt_image_token>...

Here is the image generated by the assistant A.

<test_image_A_token><test_image_A_token><test_image_A_token>...

Here is the image generated by the assistant B.

<test_image_B_token><test_image_B_token><test_image_B_token>...

Begin your evaluation by comparing the two responses and provide a short explanation. Avoid any biases and ensure that the order in which the responses were presented does not influence your decision.

A.4 Prompt Strategy

Additionally, we experiment with alternative prompt strategies that promote more reasoning by MLLM assistants, such as Chain-of-Thought (CoT) (Wei et al., 2022a) and Plan-and-Solve (PS) (Wang et al., 2023a).

The Chain-of-Thought (CoT) prompt is demonstrated below, wherein a specific sentence is added at the commencement of the assistant's response.

<USER>

You are a helpful assistant that can generate Python code using matplotlib ...

<Assistant>

Let us think step by step. ...

Kendall's Tau		Pearson		Spearman	
Coefficient	p-value	Coefficient	p-value	Coefficient	p-value
0.437	8.68×10^{-49}	0.479	6.89×10^{-54}	0.469	1.57×10^{-51}

Table 5: **Correlation coefficient comparison between GPT-4V evaluations and human evaluations.**

We modify Plan-and-Solve (PS) strategy to make it compatible with our visual coding task and call it as PS+ in Table 3. It first encourage MLLM assistants to make a detailed plan.

<USER>

You are a helpful assistant that can generate Python code using matplotlib ...

<Assistant>

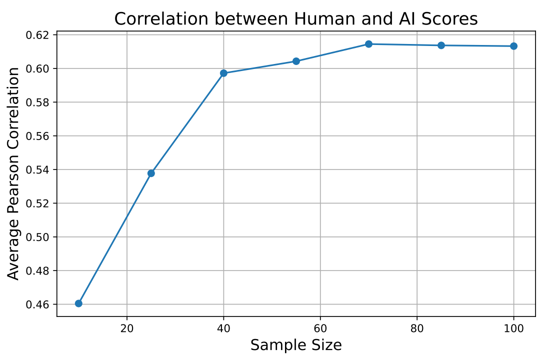
Let us first describe the plot and make a detailed plan step by step ...

If the assistant outputs the code during the first step, the strategy will terminate. Otherwise, the second step will be employed, prompting the assistant to produce the final answer based on the plan described in the first stage.

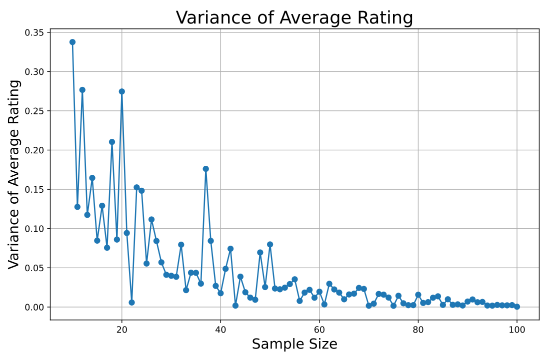
Previous messages...

<Assistant>

Based on the above description, now we are prepared to generate the code. The generated code is surrounded by ```python and ```to make it easier to be extracted by regular expressions. Therefore, the code is:



(a)

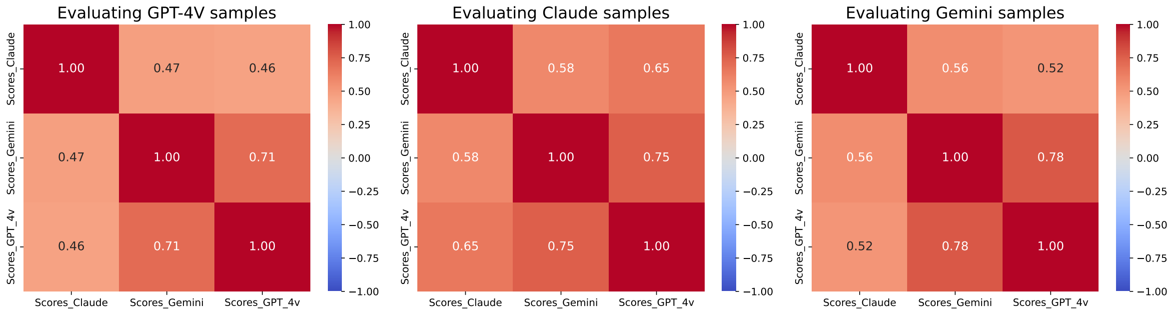


(b)

Figure 8: (a) Relationship between sample size and the average Pearson correlation coefficient between human evaluations and AI evaluations of image similarity. (b) Variance of the average rating as a function of sample size.

B Case Study

In this section, we present several examples using GPT-4V as the model under evaluation. We showcase cases from the direct asking setting (Figure 10), the conditional asking setting (Figure 11), and the pair-evaluation setting compared to Gemini-Pro (Figure 12), respectively. All the samples are drawn with the default prompt strategy.



(a) (b) (c)

Figure 9: Correlation analysis between AI tools (Claude, Gemini, GPT-4V) when evaluating images generated by different AI tools. The results show the consistency of ratings among the different AI tools. (a), (b), and (c) present the evaluated results from GPT-4V, Claude, and Gemini's generated images, with Cronbach's Alpha 0.77, 0.84, and 0.82 respectively.

C Correlation Analysis

In this section, we discuss the details of the correlation analysis.

C.1 Pair-wise Correlation Analysis

We select 20 pair evaluation samples with GPT-4V as the baseline in the conditional asking setting (10 compared to Gemini-Pro, 10 compared to Claude-3-Opus). Subsequently, we use these 20 samples to create an online questionnaire and invite colleagues from the lab, who hold at least a bachelor's degree, to participate. Each question in the questionnaire presents the ground truth image, the generated image from Assistant A, and the generated image from Assistant B. Participants are asked to choose one of the following three options:

- Assistant A's generated image is more similar to the ground truth image
- Assistant B's generated image is more similar to the ground truth image
- the level of similarity is close

In the end, we receive 46 completed questionnaires, resulting in 46 x 20 = 920 samples for conducting the correlation analysis.

C.2 Absolute Rating Correlation Analysis

We conduct a human rating to assess 100 GPT-4V generated samples with 100 people. The results are shown in Figure 8. The x-axis represents the sample size, ranging from 10 to 100, and the y-axis represents the variance of the average rating obtained from multiple iterations (5 iterations for each sample size). As the sample size increases, the correlation between human and AI scores generally rises, suggesting that larger sample sizes lead to more robust and consistent correlation estimates. The plot demonstrates that the correlation stabilizes around a value of approximately 0.62 as the sample size reaches 60 and beyond.

C.3 Correlation Analysis between AI Rating

In this section, we present a correlation analysis of AI tool ratings on images generated by various AI systems, focusing on the consistency among evaluations given by Claude, Gemini, and GPT-4V. To quantify this consistency, we utilize Cronbach's Alpha, a statistic commonly used to measure the internal consistency or reliability of a set of ratings.

Cronbach's Alpha is defined as:

$$\alpha = \frac{N \cdot \bar{c}}{\bar{v} + (N-1) \cdot \bar{c}}$$

where N is the number of items (in this case, AI tools), $\bar{c}$ is the average of all covariances between tool ratings, and $\bar{v}$ is the average variance of each individual tool rating.

The analysis of GPT-4V ratings (Figure 9a) resulted in a Cronbach's Alpha of 0.77, indicating a substantial level of agreement and confirming moderate reliability among the AI tool ratings when assessing images generated by GPT-4V. Similarly, the Cronbach's Alpha for the ratings provided by Claude (Figure 9b) was 0.84, suggesting strong reliability and showcasing a high degree of agreement among the tool ratings for images generated by Claude. In the case of images generated by Gemini (Figure 9c), the evaluation resulted in a Cronbach's Alpha of 0.82, indicating strong consistency among the tool ratings, comparable to those observed in Claude's analysis.

Overall, across all evaluations, the Cronbach's Alpha values exceed the threshold of 0.7, indicating good internal consistency and validating the reliability of the ratings among different AI tools. These results affirm that the AI tools consistently rate images, reinforcing the reliability of evaluations within our proposed benchmark.

D Evaluation Metric Details

Algorithm 1 Text-Match Ratio Calculation

Input: Reference and Generated Plots
Output: Text Match Score
Extract text elements and positions using OCR
Calculate size ratio: $s_r \leftarrow r_1/r_2$
for each position p_2 in generated text **do**
    Adjust positions $p_{2,adj} \leftarrow p_2 \times s_r$
end for
for each pair (t_1, t_2) **in** (ref, gen) **do**
    Compute distance $d \leftarrow |p_1 - p_{2,adj}|$
    $d_s \leftarrow \exp(-d/100)$
    if t_1 == t_2 **and** $d_s > 0.8$ **then**
        Mark as matched
    else
        Mark as unmatched
    end if
end for
Calculate pairs:
    $total \leftarrow matched + unmatched$
$m_s \leftarrow matched/total$
return m_s

E Datasheets

E.1 Motivation

For what purpose was the dataset created? Was there a specific task in mind? Was there a specific gap that needed to be filled? Please provide a description.

Plot2Code is a comprehensive and novel benchmark tailored for the specific multi-modal code tasks, enabling the assessment of advancements in multi-modal understanding and reasoning. We carefully collect 132 manually selected high-quality matplotlib plots across six plot types from publicly available matplotlib galleries. For each plot, we carefully offer its source code, and an descriptive instruction summarized by GPT-4. This approach enables Plot2Code to extensively evaluate MLLMs' code capabilities across various input modalities. We anticipate that Plot2Code will stimulate the research community to further explore and advance the realm of MLLMs, propelling us towards the realization of truly intelligent multi-modal systems.

E.2 Composition

What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)? Are there multiple types of instances (e.g., movies, users, and ratings; people and interactions between them; nodes and edges)? Please provide a description. How many instances are there in total (of each type, if appropriate)?

This benchmark comprises a carefully curated dataset comprising 132 matplotlib plots across 6 plot types, incorporating a total of 293 subplots sourced from matplotlib galleries, as shown in Table 1. And each plot is paired with its corresponding code and a detailed description generated by GPT-4, as shown in Figure 2. In addition to the matlotlib, we also provide other plotting library data: 150 plots from Python's plotly and 86 plots R's plotly with the corresponding code and GPT-4 summarized descriptions.

Does the dataset contain all possible instances or is it a sample (not necessarily random) of in- stances from a larger set? If the dataset is a sample, then what is the larger set? Is the sample representative of the larger set (e.g., geographic coverage)? If so, please describe how this representativeness was validated/verified. If it is not representative of the larger set, please describe why not (e.g., to cover a more diverse range of instances, because instances were withheld or unavailable).

The dataset contains a small, representative sample of chart data from various plotting libraries, rather than being derived from any single larger dataset. To some extent, it can represent plots generated by programming languages (Python, R), as it comes from examples in these classic plotting libraries. However, it cannot represent all types of plots generated by programming languages.

What data does each instance consist of? "Raw" data (e.g., unprocessed text or images) or features? In either case, please provide a description.

Every instance contains the following components: 1. Images: Plots from from various plotting libraries examples. 2. Codes: Corresponding codes of the plots. 3. Instructions: GPT-4 summarized descriptions of the plots.

Is there a label or target associated with each instance? If so, please provide a description.

The task is for the MLLM to write the code that generates the corresponding plot based on the given instruction and plot.

Is any information missing from individual instances? If so, please provide a description, explaining why this information is missing (e.g., because it was unavailable). This does not include intentionally removed information, but might include, e.g., redacted text.

All instances contain the complete information (plot, code and instruction).

Are relationships between individual instances made explicit (e.g., users' movie ratings, social network links)? If so, please describe how these relationships are made explicit.

Yes, the instances are explicitly grouped by the base type they were sampled from. The grouping is reflected in the directory structure.

Are there recommended data splits (e.g., training, development/validation, testing)? If so, please provide a description of these splits, explaining the rationale behind them.

We split the dataset according to different libraries (matplotlib, Python's plotly, and R's plotly). Each chart corresponds to code and its instruction. All data is used for evaluating the MLLM, not for training. This split is designed to assess the MLLM's capabilities across different libraries. However, dataset users can freely design other splits according to their task requirements.

Are there any errors, sources of noise, or redundancies in the dataset? If so, please provide a description.

There are no errors, sources of noise, or redundancies in the dataset.

Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)? If it links to or re- lies on external resources, a) are there guarantees that they will exist, and remain constant, over time; b) are there official archival versions of the complete dataset (i.e., including the external resources as they existed at the time the dataset was created); c) are there any restrictions (e.g., licenses, fees) associated with any of the external resources that might apply to a future user? Please provide descriptions of all external resources and any restrictions associated with them, as well as links or other access points, as appropriate.

The dataset is self-contained. In this study, we crawled every website link listed in the Matplotlib gallery and Plotly documentation to collect data for our analysis. Both Matplotlib and Plotly libraries are distributed under permissive open-source licenses. We have taken the following steps to ensure compliance with the respective license terms:

- **Acknowledgment of Licenses:** We acknowledge that the Matplotlib library and its gallery are distributed under the BSD 3-Clause License, and the Plotly library and its documentation are distributed under the MIT License.

- **Retention of Copyright Notices:** We have retained all copyright notices and license information from the original Matplotlib gallery content and Plotly documentation, as required by their respective licenses.

- **Usage and Distribution:** Our use of the Matplotlib gallery and Plotly documentation content is solely for academic and research purposes. We have not modified the original content from the Matplotlib gallery or Plotly documentation, and any distribution of our work will include proper attribution to the Matplotlib and Plotly projects.

By adhering to these guidelines, we ensure that our use of the Matplotlib and Plotly content is fully compliant with their respective licenses.

Does the dataset contain data that might be considered confidential (e.g., data that is protected by legal privilege or by doctor-patient confidentiality, data that includes the content of individuals non-public communications)? If so, please provide a description.

The dataset does not contain data that might be considered confidential.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety? If so, please describe why.

The dataset does not contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety.

Does the dataset relate to people? If not, you may skip the remaining questions in this section.

The dataset does not relate to any people.

E.3 Collection Process

How was the data associated with each instance acquired? Was the data directly observable (e.g., raw text, movie ratings), reported by subjects (e.g., survey responses), or indirectly inferred/derived from other data (e.g., part-of-speech tags, model-based guesses for age or language)? If data was reported by subjects or indirectly inferred/derived from other data, was the data validated/verified? If so, please describe how.

The instruction data for the plots was generated by GPT-4. The authors reviewed all the instructions to ensure their accuracy. The plots and codes were crawled from the corresponding plotting package webpages, and they are all validated by the authors.

If the dataset is a sample from a larger set, what was the sampling strategy (e.g., deterministic, probabilistic with specific sampling probabilities)?

The dataset does not come from a larger dataset.

Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and how were they compensated (e.g., how much were crowdworkers paid)?

The data collection process was automatic crawled and generated for the most part. All the programs for collecting and generating data are written by the authors.

Over what timeframe was the data collected? Does this timeframe match the creation timeframe of the data associated with the instances (e.g., recent crawl of old news articles)? If not, please describe the timeframe in which the data associated with the instances was created.

The bulk of the data was collected and generated in March, 2024-May 2024. We crawled plots and codes from the latest plot package website.

Were any ethical review processes conducted (e.g., by an institutional review board)? If so, please provide a description of these review processes, including the outcomes, as well as a link or other access point to any supporting documentation.

No, there was no need for ethical review as the dataset is fully from open public code package.

E.4 Preprocessing/cleaning/labeling

Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature ex- traction, removal of instances, processing of missing values)? If so, please provide a description. If not, you may skip the remainder of the questions in this section.

Our data process for acquiring high-quality plot-code pairs to evaluate MLLM code generation capabilities involved three main steps: 1. Generation Filtering: Code was extracted from HTML files containing a single code block, resulting in 529 plot-code pairs. 2.Type Filtering: Only simple, static matplotlib figures were kept, excluding animations and interactive plots. 3.Manual Curation: Examples were manually selected based on criteria such as lack of external dependencies, diversity in plot characteristics, and varied difficulty levels.

E.5 Uses

Has the dataset been used for any tasks already? If so, please provide a description.

No.

Is there a repository that links to any or all papers or systems that use the dataset? If so, please provide a link or other access point.

No.

What (other) tasks could the dataset be used for?

You can use this data for the ChartQA benchmark, as it includes plots and corresponding code. You can construct QA data based on the data in the code.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses? For example, is there anything that a future user might need to know to avoid uses that could result in unfair treatment of individuals or groups (e.g., stereo- typing, quality of service issues) or other undesirable harms (e.g., financial harms, legal risks) If so, please provide a description. Is there anything a future user could do to mitigate these undesirable harms?

No.

Are there tasks for which the dataset should not be used? If so, please provide a description.

No.

E.6 Distribution

Will the dataset be distributed to third parties out- side of the entity (e.g., company, institution, organization) on behalf of which the dataset was created? If so, please provide a description.

Yes, the dataset is available publicly for anyone interested to use.

How will the dataset will be distributed (e.g., tar- ball on website, API, GitHub) Does the dataset have a digital object identifier (DOI)?

The dataset is distributed through Hugging Face, which will ensure the long term data availability, in https://huggingface.co/datasets/TencentARC/Plot2Code.

When will the dataset be distributed?

The dataset has been released now.

Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)? If so, please describe this license and/or ToU, and provide a link or other access point to, or otherwise reproduce, any relevant licensing terms or ToU, as well as any fees associated with these restrictions.

This dataset is open-sourced under the Apache-2.0.

Have any third parties imposed IP-based or other restrictions on the data associated with the instances? If so, please describe these restrictions, and provide a link or other access point to, or otherwise re- produce, any relevant licensing terms, as well as any fees associated with these restrictions.

This dataset is open-sourced under the Apache-2.0. These evaluation code and datasets are fully open for academic research and can be used for commercial purposes with official written permission.

Do any export controls or other regulatory restrictions apply to the dataset or to individual instances? If so, please describe these restrictions, and provide a link or other access point to, or otherwise reproduce, any supporting documentation.

No.

E.7 Maintenance

Who will be supporting/hosting/maintaining the dataset?

Support and management will be provided by the dataset authors.

How can the owner/curator/manager of the dataset be contacted (e.g., email address)?

Main contact: Chengyue Wu (hillwu@connect.hku.hk)

Additional contact: Yixiao Ge (yixiaoge@tencent.com)

Is there an erratum? If so, please provide a link or other access point.

No.

Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)? If so, please describe how often, by whom, and how updates will be communicated to users (e.g., mailing list, GitHub)?

Yes, the development of the dataset is planned to continue, and contributions from users are also welcomed.

If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (e.g., were individuals in question told that their data would be retained for a fixed period of time and then deleted)? If so, please describe these limits and explain how they will be enforced.

The dataset does not relate to people.

Will older versions of the dataset continue to be supported/hosted/maintained? If so, please describe how. If not, please describe how its obsolescence will be communicated to users.

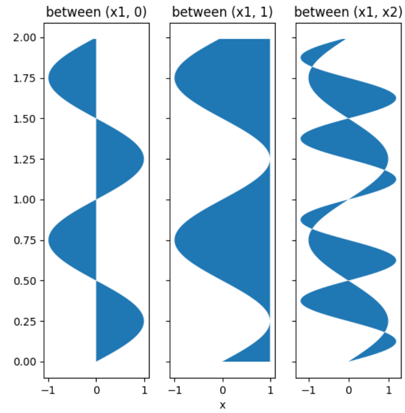
Yes, we plan to support versioning of the dataset so that all the versions are available to potential users. Hugging Face platform will maintain the history of version.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so? If so, please provide a description. Will these contributions be validated/verified? If so, please describe how. If not, why not? Is there a process for communicating/distributing these contributions to other users? If so, please provide a description.

Everyone can contribute through the Hugging Face platform. We will carefully review the contributions to assess their value before merging them into our dataset.

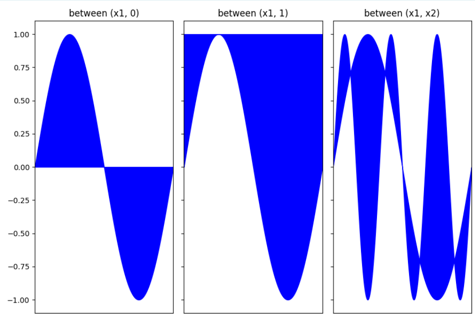
Direct Asking

<USER>
Direct Asking Prompt (illustrated in Appendix A.1)



<GPT-4V>

```
import matplotlib.pyplot as plt
import numpy as np
# Data for plotting
x = np.linspace(0, 2 * np.pi, 500)
y1 = np.sin(x)
y2 = np.sin(3 * x)
fig, axs = plt.subplots(1, 3, figsize=(9, 6), sharey=True)
# Plot with fill between y1 and 0
axs[0].fill_between(x, y1, 0, color='blue')
axs[0].set_title('between (x1, 0)')
# Plot with fill between y1 and 1
axs[1].fill_between(x, y1, 1, color='blue')
axs[1].set_title('between (x1, 1)')
# Plot with fill between y1 and y2
axs[2].fill_between(x, y1, y2, color='blue')
axs[2].set_title('between (x1, x2)')
# Set the x label only on the last subplot
for ax in axs:
    ax.label_outer()
    ax.set_xlim([0, 2*np.pi])
    ax.set_xticks([])  # Remove x-axis labels
plt.tight_layout()
plt.show()
```



<Evaluation>
To evaluate the similarity between the reference image and the generated image, we will consider several factors:

1. Overall Appearance: The layout of both images includes three separate plots aligned horizontally. However, the reference image has a square aspect ratio for each plot, while the generated image has a rectangular aspect ratio, making the plots wider.
2. Colors: Both images use a similar shade of blue for the filled areas within the plots.
3. Shapes: The shapes within the plots are significantly different. The reference image contains smooth, wave-like shapes that are filled between certain bounds. The generated image also has wave-like shapes, but they are more sinusoidal and regular, with sharper peaks and troughs.
4. Positions: The positioning of the plots relative to each other is consistent in both images, with each plot directly adjacent to the next.
5. Visual Elements: The axes, labels, and titles are present in both images, but the font size and the exact positioning may vary slightly due to the different aspect ratios of the plots.

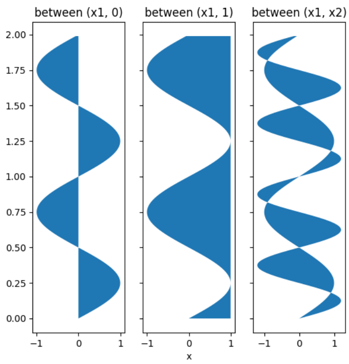
Considering these factors, the images share some similarities in layout and color but differ significantly in the shapes and aspect ratios of the plots. The overall concept seems to be similar, but the execution and the details of the plots are quite different.
Rating: [[3]]

Figure 10: A case of Direct Asking, showcasing the generated code, plot, and evaluation result.

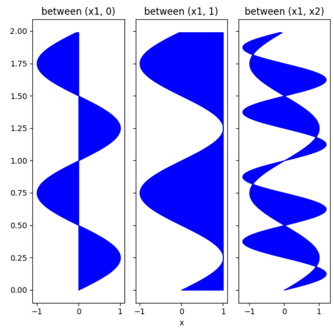
Conditional Asking

<USER>
The figure created by this code consists of three subplots arranged horizontally, all sharing the same y-axis. The figure size is 6 by 6. The first subplot is a filled area plot between the y-axis and a curve defined by the sine of 2 times pi times y. The title of this subplot is 'between (x1, 0)'. The second subplot is a filled area plot between a vertical line at x=1 and the same sine curve as in the first subplot. The title of this subplot is 'between (x1, 1)', and it also has an x-axis label 'x'. The third subplot is a filled area plot between the sine curve and another curve defined by 1.2 times the sine of 4 times pi times y. The title of this subplot is 'between (x1, x2)'. The y values range from 0 to 2 with a step of 0.01. The x values for the first curve are the sine of 2 times pi times y, and for the second curve are 1.2 times the sine of 4 times pi times y.

Direct Asking Prompt (illustrated in Appendix A.1)



<GPT-4V>
```
import matplotlib.pyplot as plt
import numpy as np
# Define the y values
y = np.arange(0, 2, 0.01)
# Define the x values for the curves
x1 = np.sin(2 * np.pi * y)
x2 = 1.2 * np.sin(4 * np.pi * y)
# Create the figure and subplots
fig, axs = plt.subplots(1, 3, figsize=(6, 6), sharey=True)
# First subplot
axs[0].fill_betweenx(y, x1, 0, color='blue')
axs[0].set_title('between (x1, 0)')
# Second subplot
axs[1].fill_betweenx(y, x1, 1, color='blue')
axs[1].set_title('between (x1, 1)')
axs[1].set_xlabel('x')
# Third subplot
axs[2].fill_betweenx(y, x1, x2, color='blue')
axs[2].set_title('between (x1, x2)')
# Display the plot
plt.tight_layout()
plt.show()
```



<Evaluation>
To evaluate the similarity between the reference image and the AI-generated image, we will compare the following elements:
1. Overall Appearance: We will look at the layout and composition of the images.
2. Colors: We will compare the color schemes used in both images.
3. Shapes: We will assess the shapes present in the images and their accuracy.
4. Positions: We will examine the positioning of the elements within the images.
5. Other Visual Elements: We will consider any additional visual elements that contribute to the likeness or differences between the images.

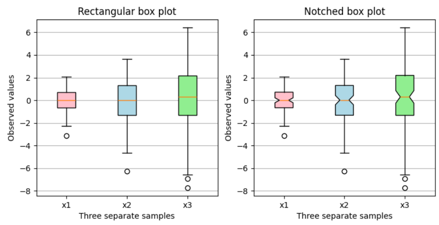
Upon comparing the two images:
- The overall appearance of both images is quite similar, with three panels displaying what appears to be filled curves.
- The colors used in both images are shades of blue, although the reference image has a lighter shade compared to the AI-generated image's deeper blue.
- The shapes of the filled curves are very similar, with slight variations in the curvature that could be attributed to the randomness in the data generation or rendering process.
- The positions of the curves within the panels are consistent, with each panel showing a progression of the shape's complexity or the interaction between two variables.
- Other visual elements such as the axes, labels, and titles appear to be consistent across both images.

Considering these factors, the images are quite similar, with only minor differences in the shade of blue and slight variations in the shapes' curvature.
Rating: [[9]]

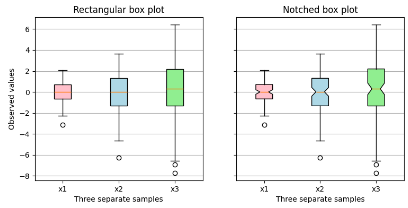
Figure 11: A case of Conditional Asking, showcasing the generated code, plot, and evaluation result.

Pair Evaluation

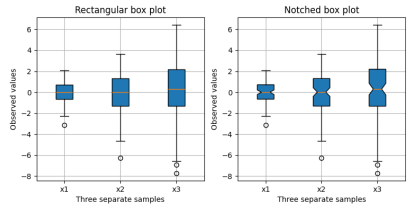
Ground Truth Image



Assistant A



Assistant B



GPT-4V Judgment (when A is placed in the first position):

Upon comparing the two responses to the ground truth image, it is clear that the image generated by Assistant A is more similar to the ground truth image than the image generated by Assistant B.

Assistant A's image closely matches the ground truth in terms of overall appearance, colors, shapes, and positions. The colors of the box plots (red, blue, green) are identical to the ground truth, and the layout of the plots, including the titles and axis labels, are accurately reproduced.

Assistant B's image, while maintaining the correct structure of a box plot, does not replicate the colors of the box plots accurately. All the boxes are blue, which differs from the ground truth image that has distinct colors for each box plot. Additionally, the titles and axis labels are missing, which are present in the ground truth.

Therefore, based on the evaluation criteria of overall appearance, colors, shapes, positions, and other visual elements, Assistant A has generated an image that is more similar to the ground truth image.

GPT-4V Judgment (when B is placed in the first position):

Upon comparing the images generated by Assistant A and Assistant B with the ground truth image, it is clear that Assistant B has produced an image that is more similar to the ground truth.

Assistant A's image differs significantly in color, using a uniform blue for all box plots, whereas the ground truth image uses distinct colors for each box plot (red, blue, green). Additionally, Assistant A's image does not replicate the notched feature in the right-hand side box plots, which is a key characteristic of the ground truth image.

Assistant B's image, on the other hand, closely matches the ground truth in terms of color, with each box plot accurately reflecting the colors in the ground truth (red, blue, green). Furthermore, Assistant B has successfully replicated the notched feature in the box plots on the right-hand side, just as in the ground truth image.

Therefore, based on the evaluation criteria of overall appearance, colors, shapes, and positions, Assistant B's image is more similar to the ground truth image.

Figure 12: A case of pair evaluation. We interchange the order of responses from the two assistants and conduct the evaluation twice. Both results are presented here.