Distill Visual Chart Reasoning Ability from LLMs to MLLMs

Anonymous ACL submission

Abstract

Solving complex chart Q&A tasks requires advanced visual reasoning abilities in multimodal large language models (MLLMs), including recognizing key information from visual inputs and conducting reasoning over it. While fine-tuning MLLMs for reasoning is critical, collecting and annotating charts and questions is expensive, hard to scale, and often results in low-quality annotations. To address this, we propose *Code-as-Intermediary Translation* (CIT), a cost-effective, efficient and scalable data synthesis method for distilling visual reasoning abilities **from LLMs to MLLMs**. The code serves as an intermediary that translates visual chart representations into textual representations, enabling language models to understand cross-modal information and generate reasoning chains accordingly. In this way, we can employ text-based synthesizing techniques to expand chart-plotting code and generate high-quality Q&A pairs for training models. This produces **REACHQA**, a dataset containing 3k reasoning-intensive charts and 20k Q&A pairs to enhance both recognition and reasoning abilities of MLLMs. Experiments show that models fine-tuned with REACHQA not only perform well on chart-related tasks but also show performance gains on general reasoning benchmarks.

1 Introduction

Multimodal large language models (MLLMs) have achieved notable progress, particularly in visual recognition tasks (OpenAI, 2024a; Anthropic, 2024). However, their ability to comprehend complex images like charts in real-world contexts and to address reasoning-intensive questions remains limited compared to humans (Masry et al., 2022; Huang et al., 2024; Wang et al., 2024c). Our analysis of the error distribution in ChartQA (Figure 1) also reveals two main failure modes in current MLLMs: while most errors originate from visual misrecognition, a substantial portion arises from flawed reasoning even when visual elements are correctly identified. This contrasts sharply with human performance (Wang et al., 2024a,c), since we can purposefully identify task-critical information from images and engage in step-by-step reasoning processes. These observations motivate our investigation into bridging this capability gap through the acquisition of human-like reasoning patterns.

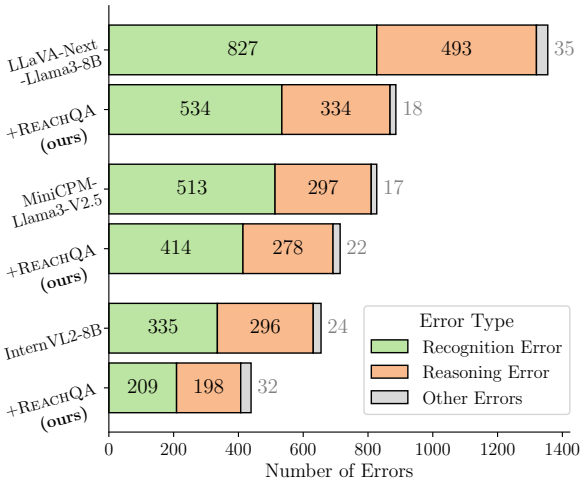


Figure 1: Error distribution of three baseline models vs. our REACHQA-trained versions on ChartQA test set (Masry et al., 2022), as judged by GPT-4o. Error types are categorized into Recognition Error, Reasoning Error, and Other Errors (question misinterpretation, factual inconsistency or hallucination, and response refusal).

While distilling expert rationales from humans or stronger models presents a promising pathway for improving reasoning abilities (Han et al., 2023; Meng et al., 2024; Masry et al., 2024a,b), constructing high-quality training data for chart-related tasks is expensive and hard to scale. Early approaches typically rely on manual chart collection from online sources, meticulous data filtering and annotation (Masry et al., 2022; Wang et al., 2024c). Recent attempts to automate Q&A generation through LLMs often use data tables as inputs (Han et al., 2023; Masry et al., 2024a), which ne-

glect the visual-semantic features of charts. Even with the use of MLLMs (Masry et al., 2024b), our preliminary study (§ 2.2) shows they also struggle to produce accurate and challenging data for advanced reasoning skill acquisition. In comparison, we find that when LLMs process charts in a better textual format—**code**, they can generate Q&A pairs at lower costs and with higher quality.

Inspired by the concept of intermediary translation (Zarechnak, 1986; Léon, 2007), which refers to using a bridge language to improve translation quality across diverse languages in literary studies, we introduce **Code-as-Intermediary Translation (CIT)**. In this method, the code acts as an intermediary, converting chart images into textual representations by faithfully encoding visual-semantic features within itself. This process enables LLMs to understand cross-modal information more accurately, thereby generating visually complex Q&A pairs with high-quality reasoning rationales. Furthermore, it facilitates the adoption of text-based instruction augmentation strategies, such as Self-Instruct (Wang et al., 2023) and Evol-Instruct (Xu et al., 2024), to expand the quantity and enhance the complexity of the synthetic charts. Starting with 33 seed codes collected from the Matplotlib gallery, we synthesize more chart-plotting codes covering diverse types and topics, and then complicate them to create richer ones. Finally, using the synthetic codes as a bridge, we generate charts (via Python) and instructions (via LLMs) in a bi-directional process, ensuring the alignment between modalities.

With the CIT method, we construct **REACHQA**, a multimodal instruction dataset containing $3,249$ **rea**soning-intensive **ch**arts and $19,963$ Q&A pairs, all at a remarkably low cost of just $\$300$. The dataset comprises questions focused on both visual recognition and reasoning, designed to address the dual challenges of current MLLMs. Additionally, we create a manually verified test set to assess models' recognition and reasoning abilities independently. Experiments demonstrate that REACHQA-trained models achieve substantial performance gains across benchmarks, with LLaVA-Next-Llama3-8B (Li et al., 2024) improving by over 30% on average, while both types of errors are significantly reduced (Figure 1). Notably, these improvements generalize beyond chart-specific tasks to broader multimodal reasoning tasks like MathVista and MATH-Vision—an outcome previously unattainable with existing chart-focused datasets. Finally, we explore REACHQA's working mechanism and more features, providing actionable guidelines for building performant multimodal datasets.

Our contributions are summarized as follows:

1. We propose Code-as-Intermediary Translation (CIT), a cost-effective and efficient method for synthesizing multimodal instruction data with code as a bridge between the two modalities.

2. Through CIT, we construct REACHQA, the first fully LLM-synthesized reasoning-intensive chart Q&A dataset, focusing on both visual recognition and reasoning abilities.

3. We conduct extensive experiments and analyses to demonstrate REACHQA's effectiveness for MLLMs, along with its strong generalization to broader multimodal reasoning tasks.

2 Background

2.1 Deficiencies in Existing Chart Datasets

Existing chart-related datasets are either collected from online data sources or generated by models, sometimes requiring manual annotation or automated question generation. Most of them focus on visual recognition tasks. While some recent works target advanced reasoning, they often struggle with scalability or other shortcomings. Table 1 summarizes these datasets, with further details below.

Chart Properties. The *visual diversity* is shaped by the variety of chart types and topics (Wang et al., 2024c). Early datasets like ChartQA and OpenCQA, sourced from limited websites, featured uniform styles with minimal diversity. To address this, recent works like ChartAst synthesize charts with randomized attributes (e.g., color, fonts) using LLMs. However, beyond the superficial variations in chart appearance, many of them overlook the *visual complexity* (Zeng et al., 2024). As models evolve, simple style changes no longer pose challenges. Datasets like CharXiv and MMC, which include complex scientific charts from arXiv papers, naturally exhibit greater complexity in recognition. Additionally, the textual format of charts is critical, enabling dataset expansion via language models.

Q&A Properties. Some benchmarks like PlotQA and ChartBench use predefined templates to generate Q&A pairs, resulting in monotonous and simplistic questions. Other datasets, such as ChartQA and CharXiv, required manual annotation, which improved quality but increased costs and

Datasets	Chart Properties				Q&A Properties			Dataset Properties		
	# Chart Type	# Chart Topic	Textual Format	Vis. Comp.	Temp. Free	Vis. Refer.	Rat. Annot.	Train Set	Test Set	Scal.
PlotQA (Methani et al., 2020)	3	-	Table	✗	✗	✓	✗	✓	✓	✗
ChartQA (Masry et al., 2022)	3	15	Table	✗	✓	✓	✗	✓	✓	✗
OpenCQA (Kantharaj et al., 2022)	5	10	Caption	✗	✓	✗	✓	✗	✓	✗
MathVista (Lu et al., 2024)	-	-	-	✗	✓	✗	✗	✗	✓	✗
CharXiv (Wang et al., 2024c)	-	-	-	✓	✓/✗	✓	✗	✗	✓	✗
ChartBench (Xu et al., 2023)	**9 / 42**	-	Table	✗	✗	✗	✗	✓	✓	✓
ChartX (Xia et al., 2024)	18	22	Code*	✗	✓	✗	✗	✗	✓	✓
MMC (Liu et al., 2024a)	6	5	Caption	✓	✓	✗	✓	✓	✓	✓/✗
ChartLlama (Han et al., 2023)	10	-	Table	✗	✓	✗	✓	✓	✓	✓
ChartAst (Meng et al., 2024)	9	-	Table	✗	✗	✗	✓	✓	✗	✓/✗
ChartInstruct (Masry et al., 2024a)	-	-	Table	✗	✓	✗	✓	✓	✗	✓/✗
ChartGemma (Masry et al., 2024b)	-	-	-	✗	✓	✓	✓	✓	✗	✓/✗
REACHQA (ours)	**10 / 32**	∞	**Code**	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of existing chart-related datasets. Only the chart Q&A task is considered, though some datasets include multiple tasks. Abbreviations: Vis.=visual, Comp.=complexity, Temp.=template, Refer.=Reference, Rat.=rationale, Annot.=annotation and Scal.=scalable. Cells marked with "✓/✗" indicate mixed attributes (e.g., partially template-based; scalable Q&A but non-scalable chart data.). "/" means the dataset includes multiple chart type granularity. "*" indicates while chart-plotting codes are public, their Q&A synthesis still relies on data tables.

hindered scalability. With the advent of LLMs, works like ChartLlama and ChartInstruct use them to generate diverse questions from data tables while also providing rationale annotations for training. However, these methods fail to capture visual elements like color, layout, and structure because they rely on only the data table. Thus, the generated Q&A pairs lack *visual references*, undermining the inherently multimodal nature of this task. To address this, ChartGemma uses MLLMs to generate Q&A directly from charts.

Dataset Properties. While manually annotated datasets like MathVista and CharXiv provide high-quality data, their development is resource-intensive, typically resulting in datasets of only a few thousand samples. In the era of LLMs, such methods are impractical for scaling to the size needed to train larger models. Recent efforts, such as ChartAst, ChartInstruct, and ChartGemma, have explored Q&A generation for dataset expansion, but they remain limited by the difficulty of collecting a large set of charts. A more scalable approach is to leverage the generative capabilities of LLMs to synthesize charts like ChartBench and ChartX.

2.2 Can LLMs Understand Charts without Visual Input?

To explore whether there is a more effective textual format for representing visual information than data tables, we propose using **code**. By precisely encoding chart structures and details, the code may serve as an ideal bridge between modalities. We design an experiment to test this hypothesis. We first collect 25 complex charts, along with their corresponding data tables and code, from authentic research papers. These charts often feature multiple or overlay plots and dense data groups, with the code averaging over 100 lines. For each sample, GPT-4o receives three types of input—table, code, and chart images—to generate a challenging Q&A pair. In total, 75 pairs are created, randomly shuffled, and then presented to annotators for blind evaluation. The annotators are asked to rate each pair on accuracy, reasoning complexity, and visual reference, using a scale of 1 (low) to 3 (high).

Input	Acc.	Reas. Comp.	Vis. Refer.	Cost ($)
Table	**2.72**	2.51	1.19	**0.047**
Code	2.60	**2.56**	2.15	0.092
Chart	1.91	1.53	**2.36**	0.107

Table 2: Rating results for three input types in our study.

The results in Table 2 indicate that both text-based inputs outperform visual chart input in the first two aspects, with code scoring 2.60 in accuracy (vs. 1.91) and 2.56 in reasoning complexity (vs. 1.53). As expected, table input has the lowest visual reference score (1.19), while chart input scores highest in this (2.36), confirming the ability of MLLMs to directly interpret visual information. Surprisingly, despite the absence of visual input to

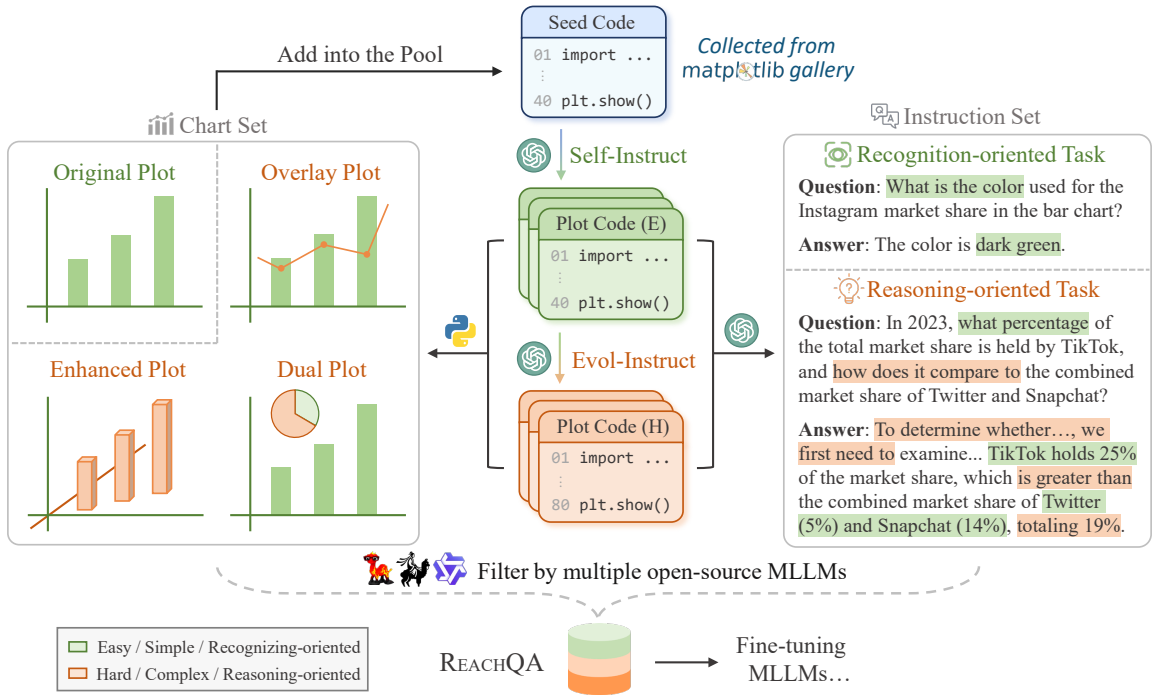


Figure 2: Overview of the Code-as-Intermediary Translation (CIT) method for synthesizing multimodal instruction data. The process starts with 33 seed codes, generating plot codes across various chart types, topics, and complexity levels via Self-Instruct and Evol-Instruct. The chart and instruction sets are constructed bi-directionally, and the final filtering yields ReachQA, a dataset for distilling visual chart reasoning abilities from LLMs to MLLMs.

the model, the code achieves a relatively high visual reference score (2.15), highlighting its potential to translate chart images into textual representations.

3 Methodology

Building on the findings above, we propose Code-as-Intermediary Translation (CIT), a data synthesis method for distilling visual reasoning abilities from LLMs to MLLMs, as illustrated in Figure 2. In the following sections, we describe how we synthesize intermediate codes (§ 3.1), generate paired charts and instructions (§ 3.2), ensure data quality (§ 3.3), and ultimately construct our dataset, ReachQA.

3.1 Intermediary Code Synthesis

Seed Code Collection. We start by collecting a small set of 33 seed code samples, which we refer to as C_{seed}. These samples are sourced directly from the official Matplotlib gallery[1] to ensure quality and minimize manual effort. Collectively, these code samples cover a diverse range of chart types, including common types like bar, line, and scatter charts, as well as more specialized charts such as bubble, contour, and donut charts. All samples are verified for executability to guarantee the reliability of the subsequent code synthesis process.

[1]https://matplotlib.org/stable/gallery/index.html

Self-Instruct for Diverse Code Generation. To expand the diversity and coverage of the chart set, we apply the Self-Instruct method (Wang et al., 2023), which synthesizes instruction data by prompting LLMs with existing ones as few-shot examples (Brown et al., 2020). In our approach, we provide 3 randomly selected code snippets as examples in each generation, guiding the model to synthesize chart-plotting code of the same kind.

To diversify chart generation, a chart type is randomly chosen from 10 major and 32 minor categories for the model to generate. For chart content, we provide two topic options, allowing the model to freely combine or expand on these themes based on its knowledge, leading to varied topics and data. A chain-of-thought (CoT) process (Wei et al., 2022) is used for code generation, starting with the chart's background and data, followed by the final executable code. This step-by-step approach ensures logical coherence and code functionality. The generated codes are referred to as C_{easy} for use in subsequent phases of the construction. The chart types and topics are detailed in Appendix A.1.

Evol-Instruct for Complex Code Generation. To enhance the visual complexity of the synthetic charts, we adopt the Evol-Instruct method (Xu et al., 2024), which leverages LLMs to evolve sim-

ple chart-plotting code into more complex versions by presenting existing code alongside an evolution strategy as context. It addresses a key limitation in prior work that emphasizes the quantity of charts while often neglecting the difficulty of chart interpretation. Starting with code samples from C_{easy}, we apply one of the following predefined evolution directions: (1) expanding the data size or number of data groups; (2) adding or modifying visual elements to enhance presentation; (3) overlaying a different type of chart on the original plot; (4) introducing an additional subplot beside the original plot. These strategies ensure that the resulting charts demand more nuanced visual interpretation and in-depth reasoning. We follow a CoT process like previous steps, where the model first analyzes the existing code and then generates the evolved one. The evolved codes, referred to as C_{hard}, are also added to the code pool for subsequent use.

3.2 Bi-directional Translation

Chart Generation through Code Execution and Self-Repair. We generate charts by executing all the Python plotting code. However, during the generation and evolution process, program errors are inevitable. To ensure correctness, we will validate the code before adding it to the pool. When errors occur, the code is not immediately discarded; instead, we apply a Self-Repair method (Chen et al., 2024a), feeding the code and execution results into the LLMs for correction. This process repeats until the code is fixed or reaches an iteration limit, after which the code is discarded if it remains faulty. On average, this approach fixes about 15% of the code generated by GPT-4o, with 5% remaining unrepairable and filtered out, yielding C_{final}.

Instruction Generation through Guided Prompting. After verifying executability, we use C_{final} to create instruction sets in the form of Q&A pairs. Building on prior work of in-context Q&A generation (Chen et al., 2023; He et al., 2024), we guide the model in two steps: first generating a batch of questions, then producing corresponding answers. To ensure high-quality answers, we also employ a step-by-step approach where the model first provides detailed calculations and analyses, which are then refined into concise, educational answers optimized for learning (Gunasekar et al., 2023). The model generates two types of instructions: *recognition-oriented*, focused on visual information retrieval, and *reasoning-oriented*, requiring both recognition and multi-step reasoning. With minimal constraints on content, the model is encouraged to explore creative and diverse instructions. Multiple questions can be generated for each chart, and redundant ones are filtered using ROUGE-L overlap, following Wang et al. (2023).

3.3 Quality Assurance

Multimodal Validation for Enhanced Data Quality. Although our dataset is fully synthesized using LLMs, we acknowledge the importance of integrating visual information to enhance data quality (Masry et al., 2024b; Zeng et al., 2024). Thus we introduce a multimodal validation step, using MLLMs to verify both generated charts and their corresponding instructions. Since models differ in architecture, visual encoders, and training recipes, they may focus on varying aspects of the images. Taking this into account, we adopt a "majority voting" approach by ensembling multiple smaller, locally hosted models. This ensures reliable visual validation while remaining cost-effective. For chart validation, each model rates charts on a scale of 1 to 5, and those below a threshold are filtered out. For instructions, both Q&A pairs and corresponding charts are fed into the models and verified, with multiple negative votes leading to sample rejection.

Testing Set Construction and Annotation Refinement. For the REACHQA testing set, we follow a similar process as in previous data generation but apply stricter filtering criteria to ensure higher quality. Additional annotators are recruited for manual review and refinement. For the charts, they first check the images to identify any potential visual errors. For the Q&A pairs, they ensure the questions are relevant to the chart and answerable, then correct any hallucinations or logical inconsistencies in the answers. Afterwards, two rounds of review are conducted to confirm the questions meet the multimodal recognition or reasoning standards in our settings. Only samples with agreement from at least two annotators are included. The inter-annotator agreement, with a kappa coefficient of 0.82, indicates strong consistency (Landis, 1977). Table 3 presents the final dataset statistics.

The total cost of data construction, excluding open-source model usage and annotation labor for the testing set, was about $300. The detailed expense breakdown is provided in Appendix A.2. Since all data splits are generated using the same process and model, we analyze potential data con-

Statistics	Train Set	Test Set
Total charts	3,249	500
- # Chart types	10 / 32	10 / 32
- # Overlay plots	1,030	220
- # Multiple plots	593	251
- Average size (px)	2480×1571	2798×1601
Unique questions	19, 963	2,000
- # Reco. per chart	2.53	2
- # Reas. per chart	3.62	2
Avg. Reco. Q. length	22.1	21.0
Avg. Reco. A. length	38.3	7.0
Avg. Reas. Q. length	38.2	35.4
Avg. Reas. A. length	68.4	24.9

Table 3: REACHQA dataset statistics. Sequence lengths are calculated based on the GPT-4o tokenizer.

tamination in Appendix A.4. The prompt templates we use in each step are shown in Appendix F.

4 Experiments

4.1 Experimental Setups

Benchmarks. We evaluate the models on three categories of tasks that cover both chart-related and general multimodal recognition and reasoning. First, traditional chart-related benchmarks are considered, including ChartQA, ChartBench, and ChartX, which primarily test recognition capabilities. Second, we assess novel chart-related benchmarks that require both recognition and reasoning, including CharXiv and our REACHQA test set. Third, we evaluate general multimodal reasoning abilities on MathVista and MATH-Vision.

Models and baselines. We evaluate a range of MLLMs from three categories: (1) Powerful proprietary models, including GPT-4o (OpenAI, 2024a), GPT-4o mini (OpenAI, 2024b), and Claude 3.5 Sonnet (Anthropic, 2024). (2) Chart-augmented open-source models, such as ChartInstruct-7B (Masry et al., 2024a), ChartAssistant-13B (Meng et al., 2024), and ChartGemma-3B (Masry et al., 2024b), which are specifically enhanced for chart-related tasks. (3) Latest general open-source models, including LLaVA-Next-Llama3-8B (Li et al., 2024), MiniCPM-Llama3-V2.5-8B (Yao et al., 2024), and InternVL2-8B (Chen et al., 2024b). For each general model, we conduct supervised fine-tuning (SFT) using the REACHQA training set. Specifically, we train three variants: one using 8k recognition-oriented samples (denoted as Reco.), one using 12k reasoning-oriented samples (denoted as Reas.), and a combined version incorporating both (denoted as All). More details on the datasets and evaluation can be found in Appendix C.

4.2 Experimental Results

Table 4 presents the quantitative results for all models across each task. We can find that:

Synthetic datasets can also effectively measure abilities. Our REACHQA test set effectively evaluates models' reasoning and recognition skills, showing trends similar to human-annotated datasets like CharXiv. For instance, GPT-4o exhibits a reasoning score of 39.70 and a recognition score of 66.80 on REACHQA, closely mirroring its performance on CharXiv (i.e., 47.10 and 84.45, respectively). This consistency suggests that LLM-generated datasets, with minimal human intervention, can rival human-labeled data. Moreover, REACHQA presents a significant challenge to models' visual abilities, as random guessing results in very low scores. In contrast, traditional benchmarks like ChartQA may allow models to leverage pre-existing knowledge, inflating results without truly testing visual capabilities (Yue et al., 2024).

Proprietary models demonstrate more balanced performance. Proprietary models like GPT-4o achieve competitive results on both traditional chart-related tasks and reasoning-intensive tasks like REACHQA and CharXiv. In contrast, open-source models, whether chart-augmented or general-purpose, excel in recognition tasks but struggle in complex ones. This disparity highlights their imbalanced capabilities, and also suggests potential overfitting to simpler charts. Although proprietary models may not always lead in specific tasks, the stable and balanced performance makes them more suitable for real-world applications.

Specialized training data significantly improves model performance. Models trained on 8k REACHQA recognition data outperform in recognition tasks, while those trained on 12k reasoning data could do better in reasoning tasks. When both data types are combined (i.e., 20k in total), models see the greatest improvement, with performance increasing by at least 15% across all models we test. Notably, the LLaVA-Next-Llama3-8B achieves a 34.8% boost in average performance. This suggests that a model's visual capability comprises two complementary aspects, and training on both

Models	Avg. (↑)	ChartQA	ChartBench		ChartX	REACHQA		CharXiv		MathVista		MATH-V
		QA	Binary	NQA	QA	Reas.	Reco.	Reas.	Desc.	Math	General	QA
Proprietary Multimodal Large Language Models												
GPT-4o mini	49.34	77.52	70.26	34.93	35.45	27.20	53.50	34.10	74.92	56.70		28.85
GPT-4o	59.85	85.70	**81.03**	**52.88**	46.60	39.70	66.80	47.10	**84.45**	63.80		30.39
Claude 3.5 Sonnet	**64.50**	**90.80**	76.72	48.29	**58.24**	**51.70**	**74.30**	**60.20**	84.30	**67.70**		**32.76**
Chart-augmented Multimodal Large Language Models												
ChartInstruct-7B	25.93	66.64	61.40	26.95	26.62	6.00	10.50	8.80	**21.40**	15.37	31.52	**10.07**
ChartAssistant-13B	28.25	79.90	58.15	24.62	23.20	**10.70**	19.60	11.70	16.93	17.78	**39.57**	8.55
ChartGemma-3B	**33.08**	**80.16**	**78.90**	**34.10**	**35.15**	9.20	**27.80**	**12.50**	21.30	**19.07**	38.04	7.70
Open-Source Multimodal Large Language Models												
LLaVA-Next-Llama3-8B	24.46	45.80	42.90	15.86	15.45	6.50	17.90	17.20	31.45	22.41	44.13	9.44
+ REACHQA (Reco.)	32.88 (+34.4%)	**66.96**	56.95	**29.52**	**27.25**	8.80	29.00	22.20	32.58	27.40	49.78	11.25
+ REACHQA (Reas.)	32.39 (+32.4%)	64.48	56.80	25.14	25.90	8.40	26.30	**22.70**	**35.67**	**28.89**	**50.65**	**11.38**
+ REACHQA (All)	**32.98 (+34.8%)**	64.56	**57.00**	29.33	27.08	**11.10**	**29.60**	22.50	32.33	27.59	50.43	11.25
MiniCPM-Llama3-V2.5	33.39	66.92	48.90	22.29	23.72	10.30	25.30	22.00	46.20	37.22	53.04	11.45
+ REACHQA (Reco.)	38.62 (+15.7%)	71.12	**56.65**	**33.29**	29.53	10.60	34.10	25.60	**48.75**	41.48	**60.43**	13.22
+ REACHQA (Reas.)	38.52 (+15.4%)	71.72	**56.65**	29.62	28.23	**11.00**	33.00	27.50	48.70	**43.52**	60.22	13.52
+ REACHQA (All)	**38.67 (+15.8%)**	**71.44**	55.80	30.43	**29.68**	**11.00**	**35.10**	**28.30**	47.62	42.22	60.00	**13.75**
InternVL2-8B	40.03	73.80	52.05	32.86	35.10	16.20	33.70	26.30	46.10	46.11	61.74	16.38
+ REACHQA (Reco.)	48.21 (+20.4%)	**82.92**	**66.35**	46.14	**46.62**	19.90	49.50	32.20	54.38	47.96	**67.61**	16.78
+ REACHQA (Reas.)	47.87 (+19.6%)	82.84	64.05	46.52	44.88	20.10	49.40	**32.80**	52.40	**49.44**	66.52	**17.66**
+ REACHQA (All)	**48.35 (+20.8%)**	82.44	65.90	**47.29**	45.38	**21.30**	**49.80**	32.70	**54.83**	48.89	66.30	17.01

Table 4: Evaluation results on seven benchmarks. The best performance for each category and task is in **bold**. The percentage of performance improvements compared to the vanilla model is denoted by (↑).

Models	Avg.	REACHQA	CharXiv	MathVista	Math-V
Base Model	16.39	6.50	17.20	32.40	9.44
+ ChartBench	17.06	7.30	17.00	33.60	10.33
+ ChartAst	17.67	7.10	20.40	32.10	11.08
+ The Cauldron	18.61	10.10	19.10	35.60	9.64
+ ChartGemma	19.11	10.00	19.40	36.40	10.62
+ REACHQA	**20.74**	**11.10**	**22.50**	**38.10**	**11.25**

Table 5: Performance comparison of models trained on different datasets. The REACHQA and CharXiv scores refer to reasoning splits here.

data types together produces optimal results. Moreover, despite the absence of math-target data in the training set, the models generalize well to the MathVista and MATH-Vision benchmarks, highlighting the transferability of multimodal reasoning abilities distilled from expert rationales. We also provide a qualitative analysis of the models' reasoning patterns in Appendix D and Figure 7, to further demonstrate REACHQA's working mechanism.

5 Discussion

5.1 The Role of Expert Rationales

We analyze how training data quality affects visual reasoning abilities by comparing major open-source datasets (ChartBench, ChartAst, ChartGemma and The Cauldron[2]). We uniformly sample 20k Q&A instructions from each dataset and train LLaVA-Next-Llama3-8B under controlled settings.

As shown in Table 5, the model trained on ChartBench performs the worst, likely due to the absence of reasoning steps in its responses. Although ChartAst includes rationale annotations, the template-based questions limit its effectiveness for learning reasoning patterns. The model trained on the mixed dataset of The Cauldron show modest improvements, but is still restricted by the subsets' quality. In contrast, models trained on ChartGemma and REACHQA perform better, likely due to the distillation of expert rationales from stronger models (e.g., Gemini Flash 1.5 and GPT-4o), which directly affect the visual reasoning abilities. Additionally, we believe the visual richness of charts, as detailed in Appendix E, may also help improve generalization.

5.2 Interaction between Recognition & Reasoning Abilities

As previously noted, the recognition and reasoning abilities are likely interdependent. Wang et al. (2024c) suggest that recognition skills serve as prerequisites for effective reasoning. To investi-

[2]Unlike other datasets, The Cauldron (Laurençon et al., 2024) is selected for its generality as a collection of 50 vision-language datasets, from which we use 7 chart-related subsets.

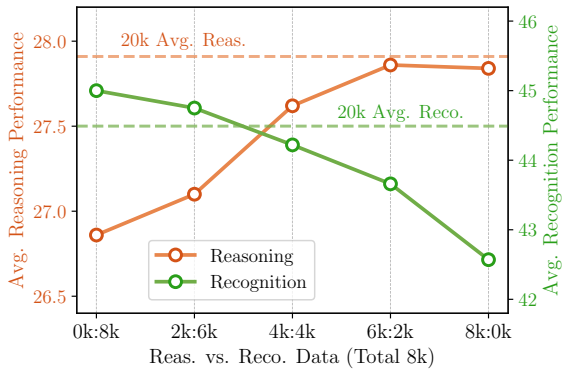


Figure 3: Performance comparison of different training data ratios with 8k total data. The dashed line represents the model's performance trained with full 20k data.

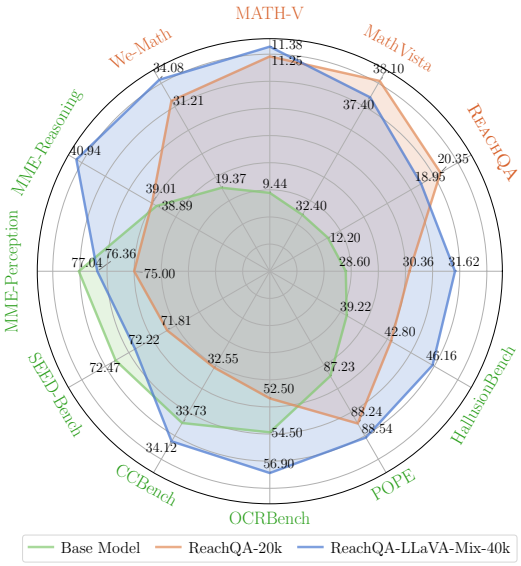


Figure 4: Performance comparison of models on 7 general tasks and 5 specialized reasoning tasks.

gate this further, we conduct an experiment with LLaVA-Next-Llama3-8B, using a fixed 8k total training data size and varying the ratio of reasoning to recognition data from 0:8 to 8:0. We evaluated the models on recognition tasks (i.e., ChartQA, ChartBench, ChartX) and reasoning tasks (i.e., REACHQA-Reas., CharXiv-Reas., MathVista).

Figure 3 shows that increasing the proportion of recognition or reasoning data improves performance on the respective tasks. Models with more recognition data outperform those trained on 20k full data for recognition tasks. However, the reasoning performance gains plateau and even decline when reasoning data exceeds 50%, suggesting diminishing returns when reasoning data is overemphasized. This supports the hypothesis that reasoning abilities are partially dependent on recognition skills. When the model fails to interpret the image accurately, its reasoning ability is likely compromised (Wang et al., 2024c). Although this study is limited by data constraints, we expect the interaction between recognition and reasoning to become more pronounced with larger datasets.

5.3 Balancing General & Specialized Abilities

We investigate how models trained on specialized data perform on general-purpose multimodal tasks. Using 7 general multimodal benchmarks and 5 reasoning-focused benchmarks, we test three versions of LLaVA-Next-Llama3-8B: the vanilla model, the one trained on 20k REACHQA samples, and another trained on 20k REACHQA plus 20k general-purpose multimodal data sampled from 779k LLaVA-NeXT-Data[3]. This dataset is chosen because the LLaVA-NeXT family of models was officially fine-tuned on it (Li et al., 2024), allowing us to approximate its original data distribution.

As shown in Figure 4, the vanilla model (green area) struggles with reasoning tasks, while the REACHQA-trained model (orange area) improves on reasoning but loses general performance. Surprisingly, by incorporating only 20k general data (blue area), the model not only restores its general multimodal performance but also retains the enhanced reasoning ability. It results in a well-balanced model with notable reasoning improvements and minimal drops in general domains.

6 Conclusion

In this work, we delve into the key challenges MLLMs face in complex chart Q&A tasks, highlighting their deficiencies in both recognition and reasoning. Building on our analysis of existing datasets and the untapped potential of LLMs, we propose Code-as-Intermediary Translation (CIT) as a novel method for distilling LLMs' abilities to improve MLLMs. With code as a bridge between visual and textual modalities, CIT enables language models to interpret complex charts more precisely, facilitating the generation of higher-quality Q&A pairs. Our synthetic dataset, REACHQA, demonstrates significant performance improvements across multiple models and benchmarks, with gains extending beyond chart-specific tasks to broader multimodal reasoning. We believe CIT offers a promising direction for scalable and cost-effective multimodal instruction data synthesis.

[3]https://huggingface.co/datasets/lmms-lab/LLaVA-NeXT-Data

Limitations

We summarize the limitations of our method as follows: (1) While CIT effectively uses code to link text and abstract images like charts and diagrams, applying this approach to natural images remains challenging. Current text-to-image models still lack precise control over fine details (Betker et al., 2023; Zhang et al., 2023), which can lead to misaligned synthetic data. Once more controllable techniques are developed, the synthesis of multimodal data could become more flexible and applicable. (2) Although multimodal validation steps were introduced to reduce errors, the synthesized charts and Q&A pairs might still contain occasional inaccuracies. Therefore, to ensure data quality for larger-scale applications, stronger models and stricter thresholds are essential. (3) Our method may not be as effective for teacher models with limited capabilities, as it is inherently on a form of distillation (Hinton et al., 2015). The success of distillation depends on the strength of the teacher model, and in our scenario, weaker models may face challenges in interpreting charts via code. Nevertheless, we believe that future models will not only become more capable but also more cost-efficient for data synthesis.

Ethical Statement

This research utilizes synthetic datasets for experimentation. We have ensured that all datasets comply with relevant ethical and privacy standards. All synthetic data have been rigorously processed to prevent the disclosure of any potentially sensitive information. We are committed to adhering to the ACL's ethical policies, ensuring transparency and reproducibility throughout the research process.

References

Anthropic. 2024. Introducing claude 3.5 sonnet.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. 2023. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Wei-Lin Chen, Cheng-Kuang Wu, Yun-Nung Chen, and Hsin-Hsi Chen. 2023. Self-icl: Zero-shot in-context learning with self-generated demonstrations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 15651–15662. Association for Computational Linguistics.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024a. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. 2024b. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2024. Colpali: Efficient document retrieval with vision language models. *CoRR*, abs/2407.01449.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Zhenyu Qiu, Wei Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, and Rongrong Ji. 2023a. MME: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023b. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. 2024. Hallusionbench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14375–14385.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. 2023. Chartllama: A multimodal llm for chart understanding and generation. *arXiv preprint arXiv:2311.16483*.

Wei He, Shichun Liu, Jun Zhao, Yiwen Ding, Yi Lu, Zhiheng Xi, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Self-demos: Eliciting out-of-demonstration generalizability in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 3829–3845. Association for Computational Linguistics.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Anwen Hu, Haiyang Xu, Jiabo Ye, Ming Yan, Liang Zhang, Bo Zhang, Ji Zhang, Qin Jin, Fei Huang, and Jingren Zhou. 2024. mplug-docowl 1.5: Unified structure learning for ocr-free document understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 3096–3120. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

Kung-Hsiang Huang, Hou Pong Chan, Yi R Fung, Haoyi Qiu, Mingyang Zhou, Shafiq Joty, Shih-Fu Chang, and Heng Ji. 2024. From pixels to insights: A survey on automatic chart understanding in the era of large foundation models. *arXiv preprint arXiv:2403.12027*.

Shankar Kantharaj, Xuan Long Do, Rixie Tiffany Ko Leong, Jia Qing Tan, Enamul Hoque, and Shafiq R. Joty. 2022. Opencqa: Open-ended question answering with charts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11817–11837. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

JR Landis. 1977. The measurement of observer agreement for categorical data. *Biometrics*.

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. 2024. What matters when building vision-language models? *arXiv preprint arXiv:2405.02246*.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *CoRR*, abs/2405.17428.

Jacqueline Léon. 2007. From universal languages to intermediary languages in machine translation. In *History of Linguistics 2002: Selected Papers from the Ninth International Conference on the History of the Language Sciences, 27–30 August 2002, São Paulo-Campinas*, volume 110, page 123. John Benjamins Publishing Amsterdam.

Bo Li, Kaichen Zhang, Hao Zhang, Dong Guo, Renrui Zhang, Feng Li, Yuanhan Zhang, Ziwei Liu, and Chunyuan Li. 2024. Llava-next: Stronger llms supercharge multimodal capabilities in the wild.

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

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. 2023b. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 292–305. Association for Computational Linguistics.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. 2022. Evit: Expediting vision transformers via token reorganizations. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Fuxiao Liu, Xiaoyang Wang, Wenlin Yao, Jianshu Chen, Kaiqiang Song, Sangwoo Cho, Yaser Yacoob, and Dong Yu. 2024a. MMC: advancing multimodal chart understanding with large-scale instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 1287–1310. Association for Computational Linguistics.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. 2023. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xucheng Yin, Cheng lin Liu, Lianwen Jin, and Xiang Bai. 2024b. Ocrbench: On the hidden mystery of ocr in large multimodal models. *arXiv preprint arXiv:2305.07895*.

Yuliang Liu, Biao Yang, Qiang Liu, Zhang Li, Zhiyin Ma, Shuo Zhang, and Xiang Bai. 2024c. Textmonkey: An ocr-free large multimodal model for understanding document. *CoRR*, abs/2403.04473.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq R. Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279.

Ahmed Masry, Mehrad Shahmohammadi, Md. Rizwan Parvez, Enamul Hoque, and Shafiq Joty. 2024a. Chartinstruct: Instruction tuning for chart comprehension and reasoning. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 10387–10409. Association for Computational Linguistics.

Ahmed Masry, Megh Thakkar, Aayush Bajaj, Aaryaman Kartha, Enamul Hoque, and Shafiq Joty. 2024b. Chartgemma: Visual instruction-tuning for chart reasoning in the wild. *arXiv preprint arXiv:2407.04172*.

Fanqing Meng, Wenqi Shao, Quanfeng Lu, Peng Gao, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024. Chartassisstant: A universal chart multimodal language model via chart-to-table pre-training and multitask instruction tuning. *arXiv preprint arXiv:2401.02384*.

Nitesh Methani, Pritha Ganguly, Mitesh M. Khapra, and Pratyush Kumar. 2020. Plotqa: Reasoning over scientific plots. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*, pages 1516–1525. IEEE.

OpenAI. 2024a. Gpt-4o.

OpenAI. 2024b. Gpt-4o mini.

Runqi Qiao, Qiuna Tan, Guanting Dong, Minhui Wu, Chong Sun, Xiaoshuai Song, Zhuoma GongQue, Shanglin Lei, Zhe Wei, Miaoxuan Zhang, Runfeng Qiao, Yifan Zhang, Xiao Zong, Yida Xu, Muxi Diao, Zhimin Bao, Chen Li, and Honggang Zhang. 2024. We-math: Does your large multimodal model achieve human-like mathematical reasoning? *arXiv preprint arXiv:2407.01284*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Mingjie Zhan, and Hongsheng Li. 2024a. Measuring multimodal mathematical reasoning with math-vision dataset. *arXiv preprint arXiv:2402.14804*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. 2024b. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, et al. 2024c. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *arXiv preprint arXiv:2406.18521*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Renqiu Xia, Bo Zhang, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Min Dou, Botian Shi, Junchi Yan, et al. 2024. Chartx & chartvlm: A versatile benchmark and foundation model for complicated chart reasoning. *arXiv preprint arXiv:2402.12185*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.

Zhengzhuo Xu, Sinan Du, Yiyan Qi, Chengjin Xu, Chun Yuan, and Jian Guo. 2023. Chartbench: A benchmark for complex visual reasoning in charts. *arXiv preprint arXiv:2312.15915*.

Zhengzhuo Xu, Bowen Qu, Yiyan Qi, Sinan Du, Chengjin Xu, Chun Yuan, and Jian Guo. 2025. Chartmoe: Mixture of diversely aligned expert connector for chart understanding. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. 2024. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*.

Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Ming Yin, Botao Yu, Ge Zhang, et al. 2024. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. *arXiv preprint arXiv:2409.02813*.

Michael Zarechnak. 1986. The intermediary language for multilanguage translation. *Computers and translation*, 1(2):83–91.

Xingchen Zeng, Haichuan Lin, Yilin Ye, and Wei Zeng. 2024. Advancing multimodal large language models in chart question answering with visualization-referenced instruction tuning. *IEEE Transactions on Visualization and Computer Graphics*.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 11941–11952. IEEE.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 3813–3824. IEEE.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Additional Dataset Details

A.1 Chart Types and Topics

We predefined several chart types and topics for Self-Instruct prompting. Table 6 shows the 9 major categories we established, with their corresponding subcategories. Additionally, Table 7 lists the 38 topics we specified. It is important to note that these topics do not reflect the actual topic distributions in the generated charts, as we encourage the model to combine and expand upon them. Regarding the distribution of chart types, we provide a breakdown in Table 8. While we aimed for a roughly balanced representation across different chart types during data construction, some degree of imbalance remains as certain types were more prone to generation errors.

A.2 Cost of REACHQA Training Data Construction

Table 9 provides a detailed expense breakdown. We executed Self-Instruct and Evol-Instruct 3,000 times each to synthesize chart-plotting code, theoretically generating 6,000 charts. However, after accounting for non-executable code and images filtered out by MLLM rating, we ultimately produced 3,249 charts for Q&A synthesis.

A.3 Reasoning Depth Across Datasets

To further clarify how REACHQA enhances reasoning abilities compared to others, we conducted a small-scale study using GPT-4o. Specifically, we randomly sampled 100 examples from each dataset and evaluated the generated answers along two proxies for reasoning depth: (1) response length (token count) and (2) rationale length (step count), following Fu et al. (2023b). As shown in Table 10, REACHQA elicited significantly longer responses and more multi-step rationales than others, indicating its strength in promoting structured, in-depth reasoning.

A.4 Data Contamination Analysis

To ensure the validity of our experimental results and exclude potential data contamination, we conduct a comprehensive analysis of data overlap from both dataset-level and split-level perspectives. First, to evaluate image-level similarity, we employed the SigLIP-400M encoder (Zhai et al., 2023) to generate embeddings for all chart images across datasets. These embeddings were then projected into a two-dimensional space using t-SNE (Van der Maaten and Hinton, 2008) for visualization, following Xu et al. (2023). Second, we analyzed query-level similarity using the NV-Embed-v2 model (Lee et al., 2024) to generate embeddings for all queries, also visualized through t-SNE.

As shown in Figure 5(a) and (c), the visualization demonstrates clear distributional differences between REACHQA and existing chart-related benchmarks. While some degree of overlap exists due to the shared nature of chart-related tasks, these instances are limited and do not compromise the overall distinctiveness of our dataset. The distinct clustering patterns in both image and query spaces support the validity of our cross-dataset evaluations and confirm that REACHQA presents novel challenges not fully captured by existing benchmarks.

To address potential data leakage between training and testing splits, which were synthesized through the same process, we conduct a more rigorous analysis as visualized in Figure 5(b) and (d). Beyond visualization, we compute pairwise similarities between all training and testing samples using

Major Category	Minor Category
Line Charts	line chart, line chart with data annotation, line chart with error bar
Pie Charts	pie chart, donut pie chart, sector pie chart, ring chart
Bar Charts	bar chart, bar chart with data annotation, stacked bar chart, percentage bar chart, horizontal bar chart
3D Bar Charts	3D bar chart, stacked 3D bar chart, percentage 3D bar chart
Node Charts	directed node chart, undirected node chart
Radar Charts	radar chart, radar chart with area filling
Area Charts	area chart, stacked area chart
Box Charts	vertical box chart, horizontal box chart
Scatter Charts	scatter chart, scatter chart with smooth fitting, 3D scatter chart (bubble chart)
Specific Charts	heat map, rose chart, funnel chart, waterfall chart, histogram, tree map

Table 6: Major categories and minor categories of charts in REACHQA.

Art and Design	Futurism and Innovation	Agriculture and Food Production
Music and Performance	Astronomy and Space	Transportation and Logistics
Business and Finance	Social Media and the Web	Real Estate and Housing Market
Travel and Exploration	Society and Community	Government and Public Policy
Books and Publishing	Physics and Chemistry	Education and Academics
Literature and Writing	Energy and Utilities	Environment and Sustainability
History and Culture	Biology and Life Sciences	Language and Communication
Architecture and Building	Retail and E-commerce	Social Sciences and Humanities
Fashion and Style	Religion and Spirituality	Manufacturing and Production
Marketing and Advertising	Food and Beverage Industry	Artificial Intelligence and Robotics
Law and Legal Affairs	Healthcare and Health	Human Resources and Employee Management
Film and Cinema	Sports and Entertainment	Computer Science and Information Technology
Mathematics and Statistics	Science and Engineering	

Table 7: Predefined chart topics in Self-Instruct prompting.

the chart embeddings. Among the identified top 50 image pairs with similarity scores exceeding 0.9, our careful manual review revealed only 2 cases with notable similarities. We will exclude them from the test set in future versions and update the evaluation accordingly. For the remaining samples, our review confirmed clear differences in chart topics, data values, and query types, ensuring that no further data leakage or contamination is present.

B Additional Methodology Details

B.1 Self-Instruct and Evol-Instruct

Self-Instruct (Wang et al., 2023) is a bootstrapping approach for synthesizing instruction–response pairs by having the model generate new examples based on a small set of seeds. In our implementation, we draw three randomly selected chart-plotting snippets as few-shot prompts and then ask the model to produce novel code in the same style and get C_{easy}. Evol-Instruct (Xu et al., 2024) is an iterative refinement approach that evolves simple instructions into complex variants through incremental modifications. In our implementation, we extend simple examples from C_{easy} into more challenging ones by asking the model to "evolve" according to a specified transformation strategy and get C_{hard}. While both Self-Instruct and Evol-Instruct were originally proposed as generic methods for instruction and example expansion, our implementation adapts them specifically for chart generation. By separating the workflows into "easy" and "hard" phases, we create a curriculum of synthetic samples that progressively increases in visual and data complexity, which was not explored in prior work.

B.2 Example Outputs of Chart Generation

Figure 6 presents a side-by-side comparison of chart outputs generated by Self-Instruct and its evolved counterpart via Evol-Instruct.

B.3 Filtering Criteria

To ensure the quality of synthetic chart–question pairs, we apply a series of automatic filtering criteria, as described in Section 3.3. These filters are designed to balance effectiveness with practicality, given that large-scale manual verification is

Types	Line	Pie	Bar	3D Bar	Node	Radar	Area	Box	Scatter	Specific	Total
Train Set	522	415	478	144	120	238	513	331	244	244	3,249
Test Set	101	40	112	15	18	19	24	44	63	64	500

Table 8: Distribution of chart types in REACHQA dataset.

Step	Avg. #tokens of Input	Avg. #tokens of Output	Times	Cost ($)
Self-Instruct	$1,500 + 2,000 = 3,500$	$500 + 500 = 1,000$	3,000	~ 56.25
Evol-Instruct	$700 + 1,300 = 2,000$	$300 + 700 = 1,000$	3,000	~ 45.00
Self-Repair	500	500	1,500	~ 9.38
Reas-QA-Gen.	$1,000 + 1,500 \times 4 = 7,000$	$500 + 300 \times 4 = 1,700$	3,249	~ 112.09
Reco-QA-Gen.	$800 + 1,200 \times 4 = 5,600$	$300 + 200 \times 4 = 1,100$	3,249	~ 81.23

Table 9: The average number of input and output tokens is calculated for each step in the REACHQA construction process. In the equation, each term represents the average number of tokens per step (used only in a multi-step framework), while each multiplier corresponds to the number of times that step is executed. The pricing for GPT-4o-2024-08-06 is $2.50 per 1M input tokens and $10.00 per 1M output tokens. As a result, the total cost amounts to approximately $303.95.

Dataset	Original Data		GPT-4o Response	
	Q. Length	A. Length	A. Length	Step Count
ChartBench	18.80	2.53	272.94	11.48
ChartAst	24.17	24.22	263.49	10.65
The Cauldron	16.95	13.93	143.54	7.00
ChartGemma	21.68	27.27	150.32	7.93
REACHQA	**40.83**	**56.02**	**340.24**	**15.20**

Table 10: Comparative analysis of reasoning depth. "Q. and A. length" refer to the average token counts of original questions and answers. GPT-4o results are averaged over 100 sampled examples.

often infeasible. For high-impact subsets (e.g., test splits), we additionally incorporate targeted manual review to further enhance reliability. Possible extensions, such as raising filtering thresholds or incorporating more human effort, may improve data quality but would also entail higher costs.

Here we provide additional details regarding reasoning-oriented samples. To control their complexity, we introduce difficulty-aware filtering. Specifically, we discard examples with overly short questions or answers, which often signal shallow or trivial reasoning. We further estimate the depth of reasoning by counting the number of sentences in the answer and enforcing a minimum threshold, thereby ensuring that retained samples exhibit sufficient multi-step inference.

C Additional Experiment Details

C.1 Benchmark Details

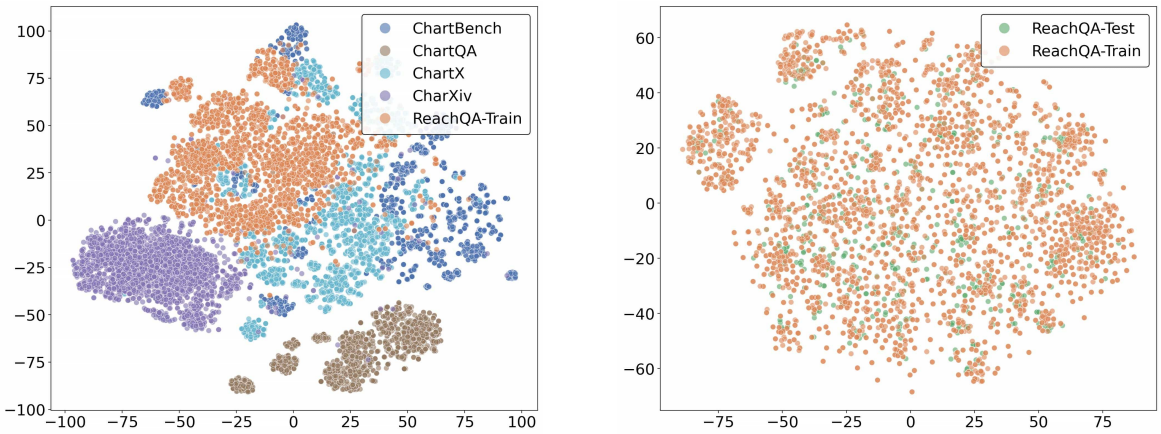
Table 11 summarizes the benchmarks used in our main experiments, including the number of samples for each dataset. Additionally, we use some other popular multimodal datasets in Section 5.3, including MME-Reasoning, MME-Perception (Fu et al., 2023a), SeedBench (Li et al., 2023a), CCBench (Liu et al., 2023), POPE (Li et al., 2023b), HallusionBench (Guan et al., 2024), OCRBench (Liu et al., 2024b), and We-Math (Qiao et al., 2024).

C.2 Training and Evaluation Details

For each general open-source model, we conduct supervised fine-tuning (SFT) using our REACHQA training set. We apply Low-rank Adapters (LoRA, Hu et al., 2022) to all linear layers of the language model and projector, with a LoRA rank of 16, a LoRA alpha of 8 and a learning rate of 2e-5. To fully leverage their capabilities, we prompt all models with a zero-shot CoT prompt, "*Let's think step by step*" (Kojima et al., 2022), following OpenAI (2024a) and Anthropic (2024). Thus, to extract answers from the model responses and assess their correctness, we employ the LLM-as-a-judge method (Zheng et al., 2023) to calculate a relaxed accuracy. The judge model used is GPT-4o, and the prompt template for evaluation can be found in Appendix F.4.

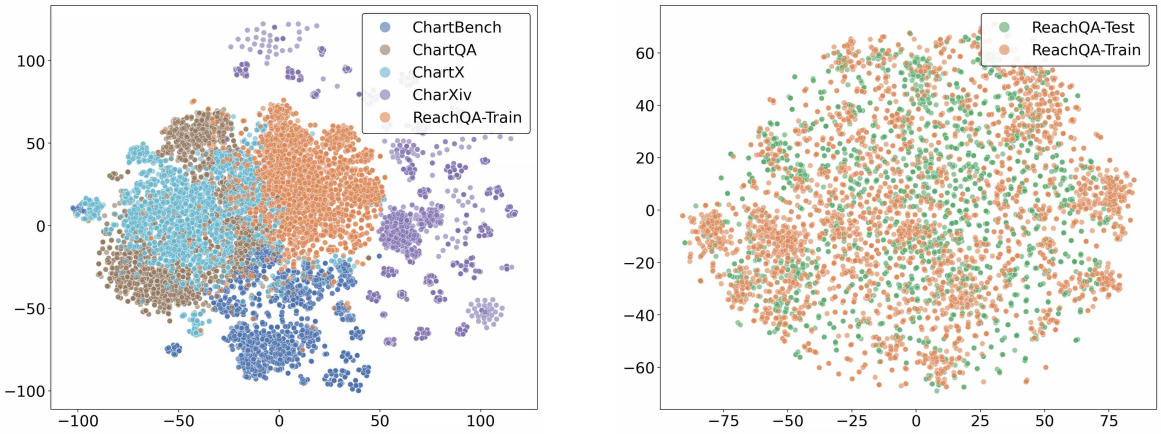
C.3 Additional Results of Recent Models

Due to space limitations, we provide supplementary results here for several recent models, including a stronger chart-augmented MLLM, ChartMoE (Xu et al., 2025), and a general-purpose open-source MLLM, Qwen2-VL (Wang et al., 2024b). As shown in Table 12, ChartMoE demonstrates



(a) **Chart: REACHQA vs. Existing datasets**

(b) **Chart: REACHQA Train vs. Test**



(c) **Query: REACHQA vs. Existing datasets**

(d) **Query: REACHQA Train vs. Test**

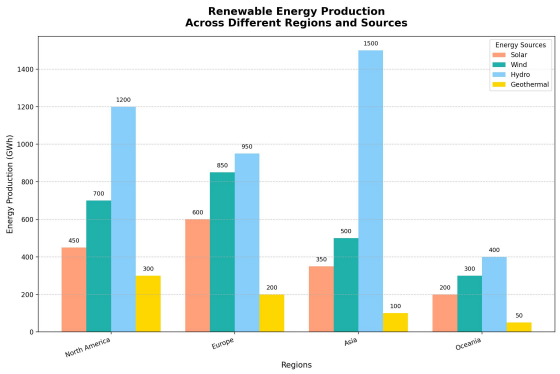
Figure 5: **Data overlap analysis visualization using t-SNE.** We analyze both image-level and query-level similarities through embedding space visualization. (a) and (c) demonstrate the distributional differences between REACHQA and existing datasets, while (b) and (d) examine potential overlap between training and testing splits. The results show clear dataset distinctiveness while revealing expected overlaps due to the shared domain of chart understanding.

more balanced performance compared to other chart-augmented models. Qwen2-VL also exhibits performance improvements that align with the trends observed in Section 4.2. However, the extent of improvement may be moderate, possibly because such models have already been exposed to reasoning-intensive samples during pre-training.

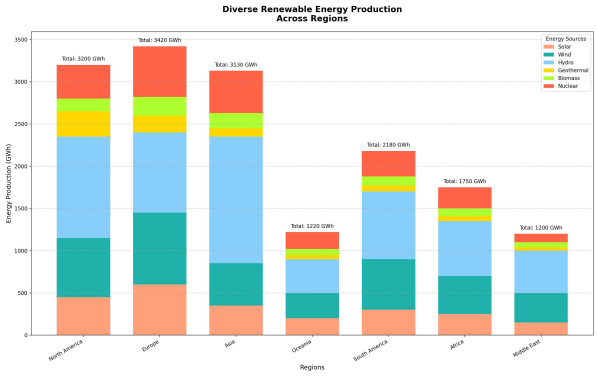
Regarding document-focused models such as TextMonkey (Liu et al., 2024c) and DocOwl 1.5 (Hu et al., 2024), we acknowledge their strengths in structured parsing of general documents (e.g., DocVQA), where *page-level layout and positional information* are crucial. However, our work specifically targets Chart Question Answering (CQA), which requires *chart-specific information extraction and numerical reasoning*. Given these differences, a direct comparison may not fully align with the scope of our study.

D A Qualitative Analysis: From the Perspective of Attention Mechanism

To explore the mechanism behind the improved performance of our fine-tuned model, we conduct an analysis of the attention patterns during the next token prediction (Liang et al., 2022; Faysse et al., 2024). Figure 7 presents a comparative case study between the vanilla model and the fine-tuned model. Here, we apply full-parameter fine-tuning instead of LoRA to induce more pronounced changes in the attention layers (Hu et al., 2022). The results show that the vanilla model produces lengthy outputs with redundant analysis and dispersed attention across the image, reaching a wrong conclusion at the end. In contrast, the fine-tuned model identifies the key information at each step, with attention that accurately focuses on relevant visual elements (i.e.,



(a) Self-Instruct Output



(b) Evol-Instruct Output

Figure 6: **Example charts generated by (a) Self-Instruct and (b) Evol-Instruct.** The "chart type" is set to "bar chart", and the "evolution direction" is "expanding the data size or number of data groups". In (a), the chart consists of a simple bar plot with 4 x-axis categories and 4 labeled series. After evolution in (b), the chart becomes a stacked bar chart with 7 x-axis categories and 6 labeled series, exhibiting increased visual and structural complexity.

Benchmark	Task Focus	Sample Details
ChartQA (Masry et al., 2022)	Chart Recognition	2.5k test samples
ChartBench (Xu et al., 2023)	Chart Recognition	2k binary QA samples and 2.1k numerical QA samples
ChartX (Xia et al., 2024)	Chart Recognition	6k QA samples
REACHQA (ours)	Chart Reco. & Reas.	1k recognition-oriented and 1k reasoning-oriented questions
CharXiv (Wang et al., 2024c)	Chart Reco. & Reas.	4k descriptive and 1k reasoning questions (validation set)
MathVista (Lu et al., 2024)	General Reasoning	540 math-targeted and 460 general VQA questions (testmini set)
MATH-Vision (Wang et al., 2024a)	General Reasoning	3,040 math competition problems

Table 11: Summary of benchmarks used in our experiments.

labels, axes and values).

This suggests that the model not only imitates expert rationales but also learns the underlying attention patterns crucial for effective visual reasoning. The model automatically establishes a synergistic relationship between recognition and reasoning capabilities, understanding what to recognize during the reasoning process and utilizing these recognition results to guide subsequent reasoning steps.

E Visualization of Charts from Different Dataset

We randomly sample several charts from the training set of ChartQA (Masry et al., 2022), ChartBench (Xu et al., 2023), ChartAst (Meng et al., 2024), ChartGemma (Masry et al., 2024b), and REACHQA. The visualization of the results is presented in Figure 8.

F Prompt Templates

We present the prompt templates used in our work.

F.1 Intermediary Code Synthesis

The prompts used for code generation via the Self-Instruct method are presented in Figure 9, and Figure 10 shows the prompts for the Evol-Instruct method. As illustrated in Figure 11, we utilize four predefined directions to evolve the simple chart-plotting code.

F.2 Bi-directional Translation

The prompt used for the Self-Repair method is presented in Figure 12. Additionally, the prompt templates for generating reasoning-oriented questions and answers are listed in Figure 13 and Figure 14. The prompt details for generating recognition-oriented questions and answers are listed in Figure 15 and Figure 16.

F.3 Quality Assurance

The prompt details for rating charts and Q&A are illustrated in Figure 17 and 18.

F.4 Evaluation

In the evaluation process, we utilize the LLM-as-a-judge method. The detailed prompt template is illustrated in Figure 19.

Models	Avg. (↑)	ChartQA	ChartBench		ChartX	REACHQA		CharXiv		MathVista		MATH-V
		QA	Binary	NQA	QA	Reas.	Reco.	Reas.	Desc.	Math	General	QA
Chart-augmented Multimodal Large Language Models												
ChartMoE-7B	40.62	80.48	60.70	38.05	39.53	12.60	37.40	27.50	40.38	36.11	60.65	13.42
Open-Source Multimodal Large Language Models												
Qwen-2-VL-7B	47.36	81.40	76.15	41.76	47.03	17.60	42.00	31.10	55.13	48.15	64.35	16.28
+ REACHQA (All)	52.00 (+9.8%)	84.96	79.00	48.90	52.20	26.80	53.60	34.60	59.25	49.26	64.78	18.68

Table 12: Additional evaluation results of recent models on seven benchmarks.

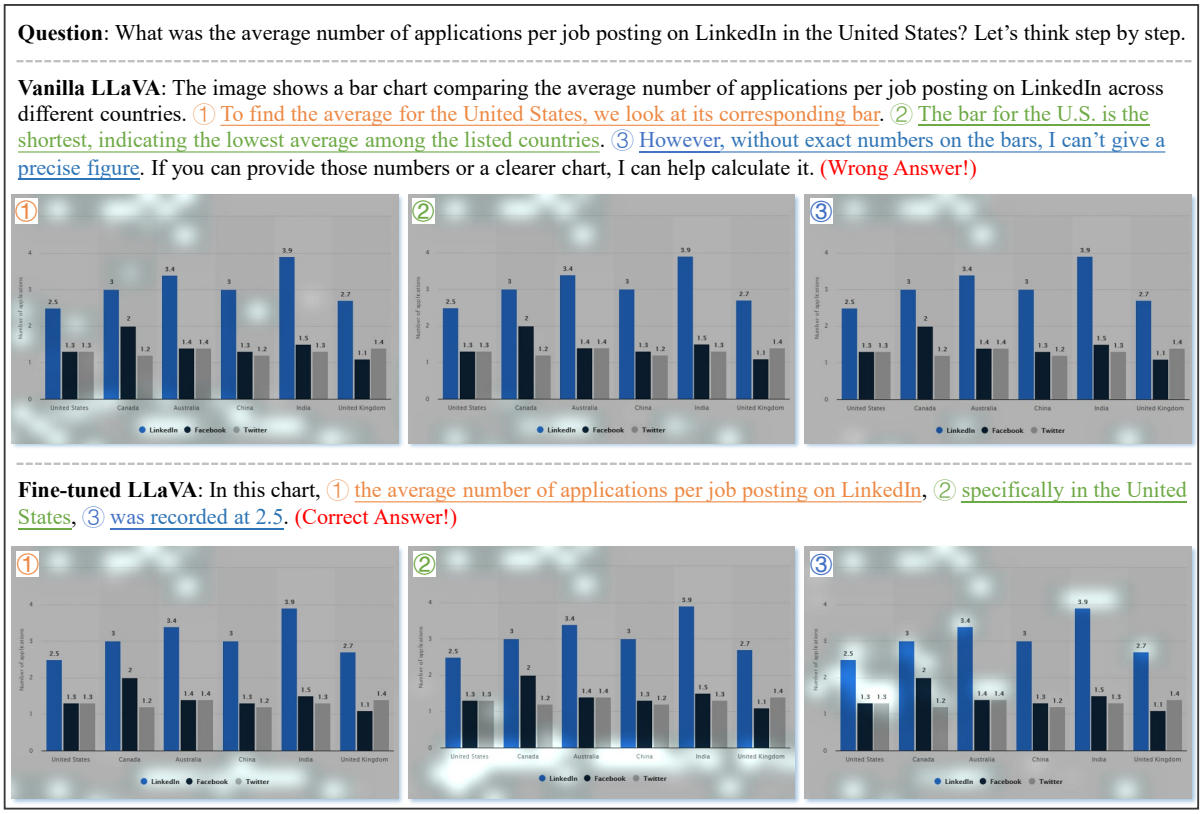
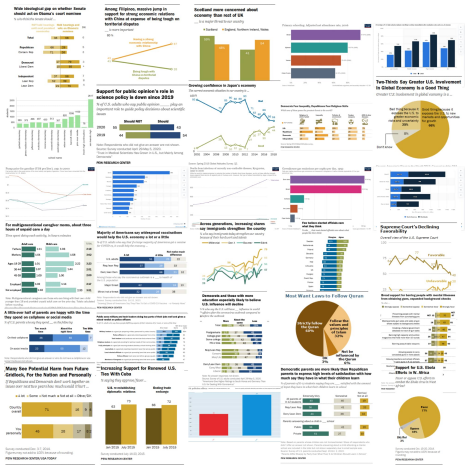


Figure 7: An example of **attention visualization** from the ChartQA dataset. The top row shows the results from the vanilla LLaVA-Next-Llama3-8B model, while the bottom row displays the results from our fine-tuned model. For each output, we present the attention distribution (highlighted zones) at **three key steps**, calculated by averaging the attention values of all tokens in each step.

(a) **ChartQA** contains 3 types of charts collected from 4 websites.

(b) **ChartBench** contains 9 types of synthetic charts but no visual complexity.

(c) **ChartAssistant** contains 9 types of synthetic charts but no visual complexity.



(d) **ChartGemma** contains charts collected from boarder websites.

(e) **REACHQA** contains 10 types of charts and more complex variations.

Figure 8: **Visualizations** of different chart-related training datasets. As shown, REACHQA and ChartGemma exhibit higher chart richness compared to several other datasets. But the charts in ChartGemma require manual collection from multiple sources (Masry et al., 2024b).

User:

As a MatplotLib expert, you are asked to write a new Python plotting script. This script will be used to generate a type-specific chart with artificial data. Here are the requirements:
1. There are several script examples from which you can draw inspiration, but try not to repeat patterns already shown in the examples to maximize diversity.
2. Use the Matplotlib library in Python for plotting. You can use auxiliary libraries such as Numpy, but make sure the code works!
3. The type of chart you need to plot is `{type}`. Therefore, everything you create must be adapted to fit this type of chart.
4. The topic of the chart can be anything you like, for example, `{topic1}`, `{topic2}`, etc.
5. Based on the given chart type and the topic you choose, you need to construct a suitable backstory, which should be reflected in the title, labels, legend, etc.
6. Based on the backstory, construct contextual data inputs in the form of Python lists or Numpy arrays. Information contained in the data can be adapted as appropriate to fit the type of chart.
7. You must not use random() to construct the data, as it needs to be explicitly created regardless of your chart type and topic.
8. Be as imaginative and creative as possible in drawing the chart, both in terms of data and plotting details.

Here are three examples to consider:
`{demo1}`
`{demo2}`
`{demo3}`

Now, let's take this task step by step. First, we have to plan out the title and backstory of the chart and create data based on the above.

Assistant: `{model_response}`

User:

Please complete the entire script by plotting a chart based on the data generated. Here are some highlighted requirements and notes.

Requirements:
1. If you find that the generated data is not appropriate while plotting the chart, modify it further as needed.
2. The information on the chart should be complete enough to be understandable, but avoid including the full backstory or too much text in the figure.
3. Avoid occlusion of visual elements. If necessary, automatically adjust the image layout before plt.show() using tight_layout().
4. If the text in the chart is too long, find a way to make it all visible instead of overlapping. If the title is too long, you can break it into multiple lines.
5. Once again, be as imaginative and creative as possible in creating the details of the chart.
6. Above all, double-check to ensure the code works. Reduce unnecessary comments and focus on functionality.

Now, generate your final plotting script in a single python code block.

Figure 9: Prompt template for code generation via Self-Instruct method.

User:

As a MatplotLib expert, you are asked to optimize a Python plotting script to make the plotted chart more complex. The script will be used to generate charts for a mathematical test, so you should make it a little more challenging.

This is the code you need to optimize:
`{code}`

Here's what I'd like you to do to optimize the chart: `{direction}`

Now, let's take this task step by step. First, please read the given code carefully and analyze the chart it draws. Then, think about your optimization ideas with the given directions.
In this step, you don't need to give the final code, only show the design ideas.

Assistant: `{model_response}`

User:

Please implement the final optimized script based on the above design ideas combined with the original code.

Remember:
1. Avoid visual elements that obscure each other, e.g., legends, labels. Automatically adjust the image layout before plt.show() using tight_layout(). if necessary.
2. If the text in the chart is too long, find a way to make all the text show up instead of overlapping. If the title is too long, you can break it into multiple lines.
3. Be as imaginative and creative as possible in creating details of the chart, but don't make the chart redundant just to cope.
4. If you are adding a new plot, take care that the chart is complete with all the elements, such as labels, axes, legends, and colors, unless it is intended to be shared with the original chart.
5. If you are adding a new plot, carefully construct meaningful data and consider whether to give the new sub-plot a sub-title.
6. You must not use random() to construct the data, as it needs to be explicitly constructed regardless of your chart type and topic.
7. Above all, double-check to make sure the code works. Reduce unnecessary comments and focus on functionality.

Now, generate your optimized plotting script in a single python code block.

Figure 10: Prompt template for code generation via Evol-Instruct method.

Evolution Direction:

- Increase the size of the input data or the number of data groups as appropriate so that it requires a higher level of mathematical understanding. Note if there is a sum requirement.
- Try changing or adding some visual elements to make visual effect better. The elements you add must make sense and not be redundant.
- Incorporate an overlay plot of a different type on the original chart. Use related but not identical data for the added plot.
- Extend an additional subplot of a different type beside the original chart (2 in total). Use related but not identical data for the added plot.

Figure 11: Predefined evolution directions for Evol-Instruct method.

User:

As a Python and Matplotlib expert, you have been asked to fix the following code.

The error code is:
`{code}`

The code reports the following error message when run: `{error}`

Please analyze the error first, and then provide the revised code within a single Python code block. There should only be one Python code block in your response, containing the complete revised code.

Figure 12: Prompt template for Self-Repair.

User:

You are both an expert Matplotlib plotter and a professional maths teacher. Now, you are asked to generate a mathematical reasoning question about a given chart. This chart and question will be used as a question on this year's college admissions examination. As a question writer, you need to ensure that the question is challenging yet fair, testing the student's ability to analyze data, interpret trends, and apply mathematical concepts.

First, please read the following plotting script in Python, try to visualize the figure in your mind and understand the meaning of the chart. After you've analyzed this chart, we'll start generating the associated question.

Here are some tips for you:
1. The plotting script (including the code itself, data mapping and labels) is absolutely correct, and you can trust it completely.
2. The question needs to be based on the chart type, chart topic, and the given data. It can relate to the chart as a whole or to localized details, so you need to look closely.
3. The question should be challenging, requiring visual observation skills and mathematical reasoning skills. So, you need to have a deep understanding of the chart.
4. If there is no data annotation in the figure, try not to generate questions that require too much numerical recognition to reduce inconsistent answers due to visual errors.
5. If some numerical recognition is needed, choose distinguishable colors, lines, heights, and other features that make it easy to estimate without data annotation.
6. You don't need to describe what the chart shows in the question text, including values, labels, etc. This can be left to the student to recognize.

Here is the plotting script:
`{code}`

Now, please generate 4 questions at a time, each of which needs to look at a different aspect of the chart.
Your output needs to follow this JSON format, and no other text included:
{"question_list": ["the question you generate"]}

Figure 13: Prompt template for generating reasoning-oriented questions.

User:

You are both a Matplotlib graphing expert and a professional math teacher. Now, you have been asked to generate an answer to a given chart and question. This chart and question will be used as a question on this year's college admissions examination. As the answer writer, you need to ensure that the answer is correct, detailed, and educational.

First, please read the following plotting script in Python, try to visualize the figure in your mind and understand the meaning of the chart. After you've analyzed this chart, we'll start generating the answer.

Here is the plotting script:
`{code}`

Here are some tips for you to generate the answer:
1. First and foremost, the answer needs to be based on the chart information.
2. In the answer, you will also need to solve the question step-by-step, including reasoning steps and recognition steps (but keep concise).
3. You need to explicitly involve a final answer; the type of answer can be a certain number, a noun, or Yes/No, etc.
4. The answer should contain multiple reasoning or calculation steps and be presented in an understandable and educational paragraph.
5. NEVER include any information relating to the Python script in the answer text, as students will ONLY have access to the plotted figure.

Here is the question: `{question}`

Your output needs to follow this JSON format, and no other text should be included:
{"analysis": "your analysis about the scirpt and question", "answer": "your step-by-step answer"}

Figure 14: Prompt template for generating reasoning-oriented answers.

User:

You are both an expert Matplotlib plotter and a professional maths teacher. Now, you are asked to generate a recognition-oriented question about a given chart. This chart and question will be used as a question on this year's elementary math examination to test students' ability to read charts.

First, please read the following plotting script in Python, try to visualize the figure in your mind and understand the meaning of the chart. After you've analyzed this chart, we'll start generating the associated question.

Here are some tips for you:
1. The plotting script (including the code itself, data mapping, and labels) is absolutely correct and you can trust it completely.
2. Descriptive questions are questions that can be answered based on basic chart information, such as titles, labels, tick marks, colors, etc.
3. The generated Q&A needs to be based on the chart type and data. It should be answerable through visual observation.
4. If there is no data annotation in the figure, try not to generate questions that require too many numerical recognitions to reduce inconsistent answers due to visual errors.
5. If some numerical recognition is needed, choose distinguishable colors, lines, heights, and other features that make it easy to estimate without data annotation.
6. You don't need to describe the content of the figure in the question text. This can be left for students to think about.
7. This question needs to explicitly involve a final answer; the type of answer can be a certain number, a noun, or Yes/No, etc.
8. NEVER include any information relating to the Python script in the question or answer, as students will ONLY have access to the plotted figure.

Here are some examples of recognition-oriented questions:
- How many colors are used in the chart? How many city categories are in the chart?
- What's the leftmost value of the bar in China? And what is the value of the bar next to it?
- For the subplot at row 2 and column 1, what is the minimum value of the solid line?
- Which name does the second-largest sector represent? What is its value?
- Does the blue triangle in the chart represent a higher value than the red circle?

Here is the plotting script:
`{code}`

Now, please generate 4 questions at a time, each of which needs to look at a different aspect of the chart.
Your output needs to follow this JSON format, and no other text included:
{"question_list": ["the question you generate"]}

Figure 15: Prompt template for generating recognition-oriented questions.

User:

You are both a Matplotlib graphing expert and a professional math teacher. Now, you have been asked to generate an answer to a given chart and question. This chart and question will be used as a question on this year's elementary math examination to test students' ability to read charts. As the answer writer, you need to ensure that the answer is correct, detailed, and educational.

First, please read the following plotting script in Python, try to visualize the figure in your mind and understand the meaning of the chart. After you've analyzed this chart, we'll start generating the answer.

Here is the plotting script:
`{code}`

Here are some tips for you to generate the answer:
1. First and foremost, the answer needs to be based on the chart information.
2. In the answer, you will also need to solve the question step-by-step, including reasoning steps and recognition steps (but keep concise).
3. You need to explicitly involve a final answer; the type of answer can be a certain number, a noun, or Yes/No, etc.
4. The answer should contain multiple reasoning or calculation steps and be presented in an understandable and educational paragraph.
5. NEVER include any information relating to the Python script in the answer text, as students will ONLY have access to the plotted figure.

Here is the question: `{question}`

Your output needs to follow this JSON format, and no other text should be included:
{"analysis": "your analysis about the scirpt and question", "answer": "your step-by-step answer"}

Figure 16: Prompt template for generating recognition-oriented answers.

User:

<image>

You are a strict MatplotLib plotter and have been asked to evaluate the given chart. Rate the chart from 1 to 5 based on these criteria:

1 point: This chart is the poorest in quality and fails to accurately represent any relevant data. It is characterized by a complete breakdown in visual representation; elements are cluttered, text heavily overlaps, legend is missing, or large areas are left blank, making the chart unreadable. The design shows no understanding of effective data visualization practices.
2 points: The chart displays incorrect or irrelevant visual elements, with significant inaccuracies that misrepresent the data. The layout suffers from clutter, substantial overlapping of text and other visual elements, such as the legend or labels, and poorly designed axes that result in uneven distribution, severely impeding accurate interpretation.
3 points: This chart represents some correct data points but makes basic errors in visual representation. It may use misleading scales, inappropriate chart types, omit key data. Visual clutter and overlapping elements, such as text obscuring parts of the chart or sub-diagrams overlapping each other, detract from the chart's clarity and readability.
4 points: The chart accurately represents most of the major data points and important details of the dataset. Minor visual errors exist, such as slight occlusions of text or sub-optimal positioning of elements like legends or labels, but these do not significantly affect the overall accuracy or readability. The chart demonstrates a good understanding of effective visualization techniques but could still be improved in terms of visual layout and the balance of details.
5 points: This is an exemplary chart that perfectly encapsulates all critical data points and relationships with outstanding visual clarity and no occlusions. It demonstrates a thorough understanding of data visualization techniques, making excellent use of space and visual elements. The chart is informative, clear, engaging, and free from any visual errors.

Score the chart on this scale, providing a short analysis and a single value. Your response should be in the format:
Analysis: (your analysis)
Rating: (int)

Figure 17: Prompt template for rating the chart quality.

User:

<image>

You are a visual question answering (VQA) data annotator. Your task is to review the following chart and question, and determine if the answer is correct based on the information in the chart. You should carefully analyze the chart, taking into account all relevant data points, labels, and trends. Then, conduct an in-depth analysis to determine if there are any unreasonable or incorrect aspects in the figure, question, or answer.

Specifically, consider the following points:
1. Are the provided question and answer relevant to the chart? Can the answer be found in the chart?
2. Do the colors in the charts and questions correspond correctly? Are there instances where the colors are incorrectly referred to?
3. Do the data in the charts and questions correspond correctly? Are there any errors in the data or misalignment of information?
4. Is the provided answer correct? Are there any logical errors or unreasonable points?
5. Apart from the points listed above, is there anything else in this question and answer that doesn't make sense?

Here is the question and answer about the given chart:
Question: {question}
Answer: {answer}

You are asked to provide a short analysis and decide whether to keep the example. Your response should be in the format:
Analysis: (your analysis)
Decision: (yes/no)

Figure 18: Prompt template for rating Q&A quality.

User:

Compare the ground truth with the prediction from AI model and determine if the prediction is correct. The question is about an image, which we have not given here. You need to determine whether the model's prediction is consistent with the ground truth. No points will be awarded for wrong answers, over answers or under answers. The reasoning process in the prediction does not need to be considered too much, you only need to determine if the final answer is consistent. There are times when the answer may have a different form of expression and some variation is acceptable.

Question: `{question}`
Ground Truth: `{answer}`
Prediction: `{prediction}`

Now, let's analyze it and then provide your judgment. Your response must follow the format below:
Analysis: (analyze the correctness briefly)
Correctness: (Yes or No)

Figure 19: Prompt template for evaluating the model prediction with LLMs.