Chain of Functions: A Programmatic Pipeline for Fine-Grained Chart Reasoning Data Generation

Zijian Li[12*], **Jingjing Fu**[1], **Lei Song**[1], **Jiang Bian**[1], **Jun Zhang**[2†], **Rui Wang**[1†]
[1]Microsoft Research [2]Hong Kong University of Science and Technology
{zijian.li}@connect.ust.hk, {jjfu,lesong,jiabia,ruiwa}@microsoft.com,
eejzhang@ust.hk

Abstract

Visual reasoning is crucial for multimodal large language models (MLLMs) to address complex chart queries, yet high-quality rationale data remains scarce. Existing methods leveraged (M)LLMs for data generation, but direct prompting often yields limited precision and diversity. In this paper, we propose *Chain of Functions (CoF)*, a novel programmatic reasoning data generation pipeline that utilizes freely-explored reasoning paths as supervision to ensure data precision and diversity. Specifically, it starts with human-free exploration among the atomic functions (e.g., maximum data and arithmetic operations) to generate diverse function chains, which are then translated into linguistic rationales and questions with only a moderate open-sourced LLM. *CoF* provides multiple benefits: 1) Precision: function-governed generation reduces hallucinations compared to freeform generation; 2) Diversity: enumerating function chains enables varied question taxonomies; 3) Explainability: function chains serve as built-in rationales, allowing fine-grained evaluation beyond overall accuracy; 4) Practicality: it eliminates reliance on extremely large models. Employing *CoF*, we construct the *ChartCoF* dataset, with 1.4k complex reasoning Q&A for fine-grained analysis and 50k Q&A for reasoning enhancement. Experiments show that *ChartCoF* improves performance for MLLMs on widely used benchmarks, and the fine-grained evaluation on *ChartCoF* reveals varying performance across question taxonomies and step numbers for each MLLM. Furthermore, the novel paradigm of function-governed rationale generation in *CoF* could inspire broader applications beyond charts. The code and data have been publicly available at https://github.com/microsoft/Chain-of-Functions.

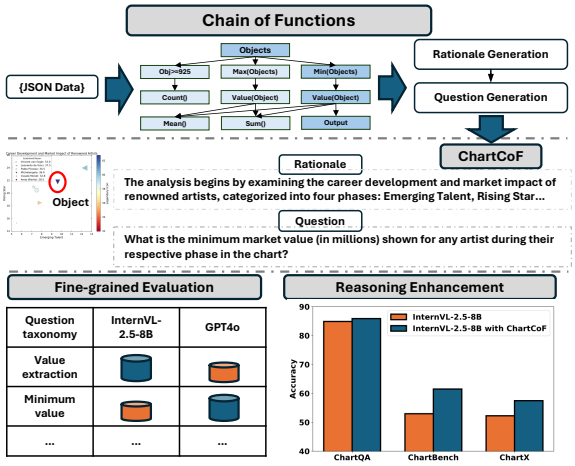


Figure 1: Our proposed *CoF* constructs a high-quality reasoning dataset *ChartCoF* for the fine-grained evaluation and reasoning enhancement of MLLMs.

1 Introduction

Recent advancements in large language models (LLMs) (Chowdhery et al., 2023; Dubey et al., 2024; Guo et al., 2025) have paved the way for the development of multi-modal large language models (MLLMs) (Liu et al., 2024b; Bai et al., 2023b), which have demonstrated a remarkable ability to understand visual semantics through the alignment between visual and embedding spaces. Despite this progress, current MLLMs exhibit limitations in their reasoning capabilities and encounter difficulties in accurately interpreting charts in scholarly articles and financial documents (Xu et al., 2023; Xia et al., 2024). This is particularly evident when they handle complex reasoning questions that necessitate accurate and step-by-step thought processes (Wang et al., 2024b). The analysis in ChartQA (Masry et al., 2022), as shown in Table 2, highlights a significant performance discrepancy between complex reasoning questions (Human set) and simpler perceptual questions (Augmented set). For instance, InternVL-2.5-8B (Chen et al., 2024b)

[*]Work done during an internship at Microsoft Research Asia.
[†]Corresponding Author.

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 200–234
December 20-24, 2025

demonstrates a performance gap of nearly 20%, which underscores the challenges that MLLMs face in bridging the gap between human-like reasoning and current computational capabilities.

Training with chain-of-thought (CoT) data has emerged as an effective strategy to enhance the reasoning abilities of MLLMs on chart understanding (Wei et al., 2022; Zhang et al., 2024c,d; He et al., 2024). Nonetheless, high-caliber CoT data for chart reasoning are scarce, which require complete reasoning processes and accurate chart information (e.g., object values and positions) in the rationales (Dong et al., 2024; Masry et al., 2024b; He et al., 2024). To generate CoT data, recent investigations have leveraged the capabilities of advanced (M)LLMs to autonomously produce questions, answers, and their corresponding rationales by either directly analyzing the charts or their textual descriptions with well-designed prompts (Liu et al., 2024a; He et al., 2024; Masry et al., 2024b). Despite these efforts, directly prompting (M)LLMs to generate questions and rationales based only on charts may result in low accuracy and limited diversity. Moreover, relying on extremely large (M)LLMs poses a notable barrier to the data scalability.

In addition to the scarcity of CoT data for effective finetuning, the evaluation of MLLMs' reasoning capabilities remains underexplored. While current benchmarks have incorporated reasoning questions to evaluate the reasoning capabilities of MLLMs (Masry et al., 2022; Xia et al., 2024; Wang et al., 2024b), these questions often lack complexity and require only short reasoning chains. More importantly, these benchmarks tend to gauge the reasoning performance in a broad sense with an overall accuracy metric, which overlooks the nuanced analysis of MLLMs' proficiency across questions that require varying reasoning chains. A fine-grained reasoning evaluation of the models' specific strengths and weaknesses on question taxonomies remains a valuable avenue for research.

In response to the scarcity of diverse and high-caliber reasoning datasets for the fine-grained evaluation and enhancement of chart reasoning, as presented in Fig. 1, we introduce a novel automatic reasoning data synthesis pipeline named *Chain of Functions (CoF)*. Unlike prior methods that rely on end-to-end LLM prompting, our approach first systematically explores chart elements through a set of atomic functions to ensure correct and diverse reasoning paths and then translate them into linguistic rationales, which greatly reduces hallucinations and enables more precise supervision. Concretely, *CoF* encompasses two key processes: *program-based functional discovery* and *reverse linguistic CoT data synthesis*. In program-based functional discovery, we carefully design atomic functions and their corresponding conditions, which are intelligently combined to form a coherent function chain based on a chart. Then in the reverse linguistic CoT data synthesis process, these function chains are translated into natural language instructions using LLMs in a reverse manner, with rationales first, and then questions. This method ensures the precision of questions, rationales, and answers. Crucially, since the reasoning process is determined by the function chain rather than by generative prompts alone, we can leverage a moderate open-sourced LLM (Qwen2.5-32B-instruct (Yang et al., 2024) used in experiments) for linguistic transfer, greatly lowering dependence on extremely large models. Furthermore, *CoF* effectively bridges structured reasoning and language modeling, with potential applications beyond charts.

Key contributions: 1) Our proposed reasoning data generation pipeline *CoF* greatly ensures explainability, precision, and diversity of generated reasoning data, thus enabling the fine-grained evaluation and reasoning enhancement for MLLMs.

2) We introduce *ChartCoF*, which encompasses an extensive variety of over 19 chart types, with a test set comprising 648 charts paired with 1,451 Q&A pairs and a training set featuring 18,349 charts with 50,329 Q&A pairs for fine-grained evaluation and model finetuning.

3) Extensive experiments demonstrate that *ChartCoF* improves accuracy for MLLMs in widely used benchmarks. Out-of-distribution (OOD) analysis and dataset comparison demonstrate the high quality of CoT data in *ChartCoF*.

4) The fine-grained evaluation reveals the weak performance of existing MLLMs on complex reasoning questions and provides deep insights into their skilled and unskilled question taxonomies.

2 Related Works

Multimodal large language models (MLLMs) have aligned the vision space with the embedding space of LLMs for visual understanding (Vaswani, 2017; Radford, 2018; Brown et al., 2020; Zhang et al., 2022; Chowdhery et al., 2023; Dubey et al., 2024; Team, 2023; Bai et al., 2023a; Yin et al., 2023), which is normally achieved via connec-

tors, e.g., Q-Former (Li et al., 2023) or MLP (Bai et al., 2023b). With connectors, Mini-GPT4, mPLUG-Owl, and InstructBLIP have extended language-only instruction tuning to multimodal tasks. LLaVA (Liu et al., 2024b; Li et al., 2024a) also maps visual features into the LLaMA (Touvron et al., 2023) embedding space using a linear layer. Its modularization and high efficiency in training make it a popular architecture of MLLMs. Despite the impressive achievements of existing open-sourced MLLMs, e.g., QwenVL (Bai et al., 2023b; Wang et al., 2024a), InternVL (Chen et al., 2024c,b), and DeepSeek-VL (Lu et al., 2024; Wu et al., 2024) in common multimodal tasks like VQA (Antol et al., 2015) and image captioning (Vinyals et al., 2015), they focus more on perception tasks while paying less attention to the visual reasoning capabilities, especially for chart understanding. In this work, we focus on improving and evaluating the reasoning capabilities for MLLMs on charts.

Chart reasoning refers to dealing with intricate tasks related to both chart-related and commonsense knowledge (Xu et al., 2024; He et al., 2024). The early two-stage inference studies first extracted structural information like tables and markdowns and then leveraged textual information for downstream understanding (Liu et al., 2023b,a; Lee et al., 2023; Wang et al., 2023). Afterwards, unified MLLMs, e.g., OneChart (Chen et al., 2024a), UniChart (Masry et al., 2023), ChartMoE (Xu et al., 2024), and TinyChart (Zhang et al., 2024a), are trained to handle varying chart-related tasks. However, these methods focus on the perception capabilities of MLLMs and overlook the reasoning capabilities. In this work, we aim to improve and evaluate the reasoning capabilities from a data aspect by generating high-quality CoT data. Many studies have utilized powerful proprietary GPT or Gemini series to generate reasoning instruction tuning data (Xu et al., 2023; Liu et al., 2024a; Xia et al., 2024; Han et al., 2023; Masry et al., 2024a,b; Fan et al., 2024; He et al., 2024; Liu et al., 2024c; Shen et al., 2024). However, directly prompting (M)LLMs based only on charts may affect the precision and diversity of training data. The excessive reliance on extremely large models also poses a significant barrier to data generation. To generate accurate Q&A, many methods attempted to manually set up templates to obtain Q&A in an end-to-end manner (Huang et al., 2024; Methani et al., 2020; Meng et al., 2024; Li et al., 2024b). Nevertheless, the predefined question templates follow a fixed pattern and may lead to limited diversity, affecting the generalization of MLLMs. In contrast, we propose a functional discovery workflow to ensure the diversity of reasoning paths and a reverse linguistic CoT data synthesis to enhance the reality and diversity of generated questions. The extra supervision of function chains during generation also refrains from the reliance on extremely large (M)LLMs. A more detailed comparison between *ChartCoF* and existing datasets is presented in Appendix B. Some examples of *ChartCoF* and existing datasets are shown in Appendix J.

3 Chain of Functions

In this section, we propose the reasoning data synthesis pipeline *chain of functions (CoF)*, including chart rendering, program-based function discovery, and reverse linguistic CoT data synthesis. An overview of *CoF* is presented in Fig. 2.

3.1 Chart Rendering with JSON Data

To ensure the consistency between charts and generated CoT data, we leverage JSON data as the intermediate representation, which is then used for chart rendering and reasoning data generation.

JSON template. We predefine the essential elements of charts in a structural presentation for subsequent chart rendering and CoT data generation, which includes the title, x label, y label, chart type, legend number, legend list, group number, group list, data points, colors, and legend colors. For some special charts, e.g., boxes, candlesticks, and node links, we include additional elements. The JSON templates for all chart types are displayed in Appendix H. The elements of the chart provide ground-truth information for chart rendering and subsequent reasoning data generation.

JSON generation. To generate realistic information for charts, the titles are generated using LLMs for each chart type. These titles are then used to generate the JSON files by prompting LLMs. To ensure the diversity of JSON data, we randomly sample the group number, legend number, and colors for JSON templates and prompt LLMs to only fill in the rest of the elements that require realistic knowledge, e.g., the group list, legend list, and data points, producing JSON seed files. To scale up, we further prompt LLMs to evolve the JSON seed and generate more realistic and accurate JSON data. All the prompts for JSON seed generation and JSON evolvement are present in Appendix I.

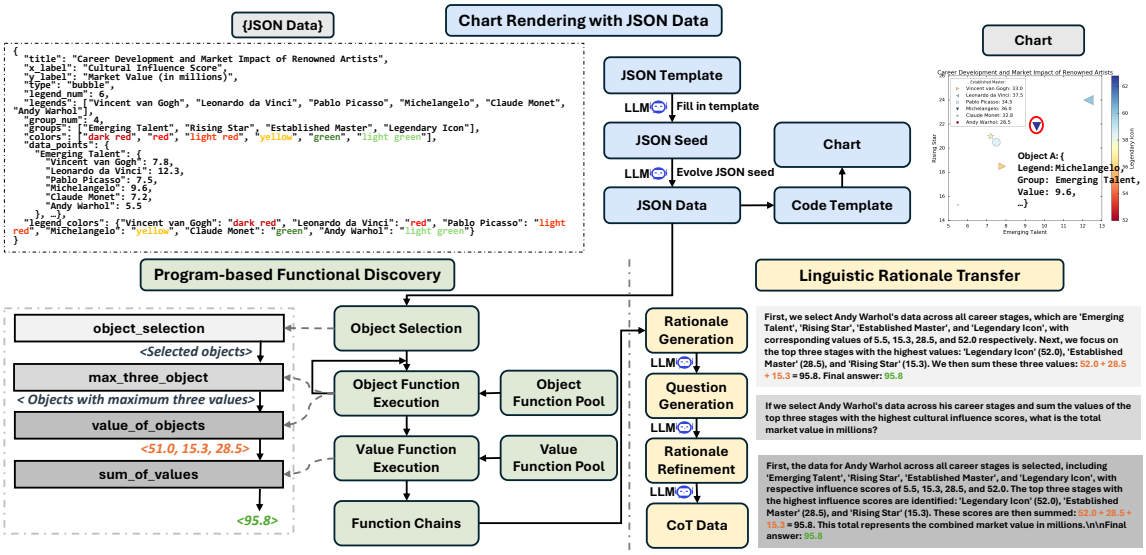


Figure 2: Overview of *chain of functions*. We prompt LLMs to fill in the JSON template to construct JSON seed and evolve (modify) it to more accurate and diverse JSON data. The JSON data are then used to generate function chains by combining functions one by one. The function chains are then transferred to CoT data by prompting LLMs.

Chart rendering with code templates. To avoid visual conflicts of chart images, we carefully design the code template for each type of chart. With the detailed information in JSON data and the well-designed code templates, we generate the chart image for each JSON file. To ensure the diversity of chart images, we use varying code libraries for chart rendering, including matplotlib, sklearn, mplfinance, plotly, seaborn, and networkx. For each type of chart, we set up different plotting styles, color transparency, and element locations. The chart examples are shown in Appendix L.

3.2 Program-based Functional Discovery

CoF conducts human-free exploration in the function pools, resulting in diverse and numerous function chains. This mirrors the pattern of chart understanding of humans: we select partial or all of the objects from the chart, extract information from them, recognize their trend and pattern, or conduct comparison and calculation between them. The function chain is discovered via a three-step workflow, which is elaborated as below.

1) **Object selection.** We regard each data point in the charts as one object. At the beginning of the workflow, partial or all of the objects from the chart are selected using chart information, including groups, legends, and colors. For the example in Fig. 2, the objects with values 5.5, 15.3, 28.5, and 52.0 are selected using the legend name 'Andy Warhol'. The selected objects are used for sequential function execution.

2) **Object function execution.** We define the functions with objects as input as the object functions to imitate reasoning processes on charts, e.g., an information extraction process to get the legend of a data point and a pattern recognition process to get the maximum data value. The selected objects are greedily input into the object functions and obtain the corresponding output, which results in a functional triplet $< input, function, output >$. For example, a function 'max_three_object' can be executed with the selected objects above and output the objects with maximum three values. The $input$ are objects, and the $output$ can be objects, numerical values, strings, or booleans. The executed functional triplet is recorded and spliced with the previously executed function triplet to form a function chain. The function chain with the final output of numerical values, strings, or booleans is regarded as a complete chain. Notably, the function chain with object output continues to perform step 2) and executes object functions again, enabling a longer function chain. In Fig. 2, a function 'value_of_objects' is executed to obtain values of the above three objects: 52.0, 28.5, and 15.3.

3) **Value function execution.** We define the functions with numerical values as input as the value functions to imitate the reasoning processes of value comparison and arithmetical operation. The function chains with the output of numerical values after step 2) are greedily input into the value functions from the value function pool

Data split	#chart types	#charts	#Q&A	#words of rationales	Lengths of function chains						#Function chains	#Functions	Question type		
					2	3	4	5	6	≥7			Binary	NQA	Text
Training set	19	18,349	50,329	66.62	38.58%	26.42%	2.0%	20.87%	9.83%	2.30%	3,134	107	16.68%	55.41%	27.91%
Test set	19	648	1,451	-	22.54%	20.74%	9.72%	16.68%	13.58%	16.75%	728	107	16.40%	67.88%	15.72%

Table 1: Statistics of training and test sets. *CoF* enables synthetic data with long and diverse reasoning paths. Detailed statistics of each chart type and function taxonomy are present in Appendix A and Appendix K, respectively.

and obtain the final answer. In Fig 2, a function 'sum_of_values' is executed to compute the sum of the obtained three values: 52.0 + 28.5 + 15.3 = 95.8. We also allow multiple separate function chains to execute value functions jointly to achieve the combination of them, resulting in a longer and more complex function chain.

To ensure the realisticity of function chains, we set up the execution conditions for each function and explore feasible function chains that meet these conditions. The details of functions for object selection, object functions, and value functions are present in Appendix K. With the above three-step workflow, we generate accurate and diverse function chains, which also provide explainability for the subsequent CoT data generation.

3.3 Reverse Linguistic CoT Data Synthesis

To generate precise and realistic rationales and questions, we transfer function chains to linguistic CoT data in a reverse manner by first rationales, then questions, and finally refining rationales.

1) **Linguistic rationale transfer.** We prompt LLMs to transfer function chains to linguistic rationales. To help LLMs better understand each function and generate more precise lingustic rationales, we also include the description of each function into the prompt.

2) **Question generation.** We prompt LLMs to generate realistic questions using JSON data, function chains, and the generated rationales. The chart information and the generated lingustic rationales enable LLMs to better understand the reasoning process and generate more precise questions.

3) **Rationale refinement.** We empirically found that initial-generated rationales are still function-like and redundant. Thus, we prompt LLMs to concisely refine the initial-generated rationales based on function chains and questions, making them align better with MLLMs. The effectiveness of rationale refinement is discussed in Appendix E.

Under the supervision of function chains, the reverse linguistic CoT data synthesis can be regarded as a translator task between function chains and linguistic CoT data, without the requirement of extremely large models. All the prompts for CoT data synthesis are present in Appendix I.

4 ChartCoF

Employing *CoF*, we construct a dataset named *ChartCoF*, which encompasses an extensive variety of 19 chart types, with a test set comprising 648 charts and 1,451 Q&As and a training set featuring 18,349 charts and 50,329 Q&As. We adopt Qwen2.5-32B-instruct (Yang et al., 2024) for data generation in *CoF*. The statistics of *ChartCoF* from the aspects of charts, function chains, and questions are described in Table 1.

Chart types. *ChartCoF* covers all the chart types that can be represented using the JSON format, with totally 19 chart types. We categorize the chart types into two groups based on their usage frequency. **Regular chart types:** We include bar charts (with single and multiple groups of bars and stacked bars), line charts (with single and multiple lines), and pie charts. These six chart types are commonly used in most of the existing datasets (Masry et al., 2022; Methani et al., 2020). **Extra chart types:** We also cover the complex chart types on existing datasets (Xu et al., 2023; Xia et al., 2024), including rings, radar, rose, candlestick, 3D-bar, treemap, funnel, heatmap, treemap, box, area, bubble, multi-axes, and node link. Note that each chart type can be **annotated or not** if allowed.

Question types. *ChartCoF* focuses on MLLMs' reasoning capabilities and thus adopts chart-related question answering (QA) tasks. We categorize the question types based on the contexts of output. **Binary:** Binary questions aim to assess the correctness of arguments. **Text:** For text questions, the answers are from the elements of charts, such as group names and legends. **Numerical question answering (NQA):** We also provide numerical questions that contain numerical computing processes.

Function chains: In *ChartCoF*, 99 object functions and 8 value functions are used to construct function chains, which results in 3,134 and 728 function chains for the training set and test set, respectively. The length of these function chains

ranges from 2 to 13, constructing the rationales with 66.62 average words for the training set.

Evaluation metrics: We follow ChartQA (Masry et al., 2022) and ChartX (Xia et al., 2024) to adopt accuracy (Acc) as the evaluation metric and allow 5% margin for numerical responses. For those MLLMs with weak instruction-following capabilities that cannot output the final answer in a correct format, we additionally prompt GPT4o to extract the final answer (Xu et al., 2023). This makes the 5% margin feasible for these MLLMs to ensure a fair comparison. The prompt for answer extraction is presented in Appendix I.6.

5 Experiments

Employing *ChartCoF*, we conduct experiments to demonstrate its effectiveness in enhancing reasoning capabilities and provide fine-grained evaluations for existing MLLMs. The out-of-distribution (OOD) analysis and dataset comparison are conducted to demonstrate the high quality of generated data in *ChartCoF*. The experiments on model and data scalability are discussed in Appendix G.

5.1 Experimental Setups

Benchmarks. Besides our proposed *ChartCoF*, we also evaluate the MLLMs in existing benchmarks about chart reasoning, including ChartQA (Masry et al., 2022), ChartBench (Xu et al., 2023), and ChartX (Xia et al., 2024). For ChartQA and ChartBench, we adopt all the test samples. For ChartX, we select only the QA task samples for evaluation and leave other unrelated tasks like chart redrawing. By following the evaluation metrics of these benchmarks, we allow 5% margin for the NQA tasks, and *Acc+* is used to evaluate the binary tasks in ChartBench (Xu et al., 2023). Since we find that inferencing with a CoT strategy cannot improve performance for baseline MLLMs, we prompt them to direct output final answers on these three benchmarks by following the recent work (Xu et al., 2024). Since the questions in the Augmented set of ChartQA are the perceptual questions without the need of thinking, we prompt our finetuned MLLMs to direct output the answer. For *ChartCoF*, we adopt the CoT strategy for all MLLMs with a focus on reasoning questions.

Models and baselines. We evaluate a wide range of MLLMs in *ChartCoF* and other benchmarks across three categories: 1) **Proprietary models**, including GPT4o (Achiam et al., 2023), GPT4V (Achiam et al., 2023), and Gemini-1.5-Flash (Team et al., 2024); 2) **Open-sourced MLLMs**, including InternLM-XComposer-2.5 (Zhang et al., 2024b), DeepSeek-VL2-small (Wu et al., 2024), LLaVA-v1.6-mistral-7B (Li et al., 2024a), CogVLM2 (Hong et al., 2024), Qwen2VL-7B (Wang et al., 2024a), and InternVL-2.5-8B (Chen et al., 2024b); 3) **Chart-specific MLLMs**. including ChartInstruct (Masry et al., 2024a), ChartVLM (Xia et al., 2024), ChartGemma (Masry et al., 2024b), ChartMoE (Xu et al., 2024), TinyChart (Zhang et al., 2024a), ChartLlama (Han et al., 2023), and ChartAst (Meng et al., 2024).

Experiment details. To demonstrate the effectiveness of *ChartCoF* in enhancing reasoning capabilities of MLLMs, we finetune two off-the-shelf MLLMs, i.e., InternVL-2.5-8B and Qwen2VL-7B, with the training set of *ChartCoF*. We finetune them in one epoch by tuning the LLM part and freezing the vision encoder and projector in 4 A100-80G GPUs, with a batch size of 32, a learning rate of $5e-6$, and a weight decay of 0.01. To achieve better instruction-following capabilities, we adopt a CoT prompt "*Think step by step to generate the rationales, and then answer the question using a single word or phrase. The output format is Rationale: [Your Rationale] Answer: [Your Answer]*" for both finetuning and inference. We also leverage self-consistency technologies to further enhance the performance by setting a temperature of 0.8 and selecting the final answer with a majority vote of 5 attempts. The evaluation metrics on *ChartCoF* can be referred to in Section 4.

5.2 Main Results

Main results on existing benchmarks. Our proposed *ChartCoF* can be used to enhance performance on widely used benchmarks. As shown in Table 2, after finetuning with *ChartCoF*, InternVL-2.5-8B and Qwen2-VL-7B significantly improve the accuracy over ChartBench, ChartQA, and ChartX, with an improvement of 8.56% for InternVL-2.5-8B in ChartBench and 7.47% for Qwen2VL-7B in ChartX, demonstrating the effectiveness of *ChartCoF* in enhancing the reasoning capabilities of existing MLLMs. The self-consistency technique can further improve the performance of finetuned MLLMs.

Main results on ChartCoF. We evaluate MLLMs on *ChartCoF* and present the results in Table 2. The existing MLLMs, including the proprietary and chart-specific models, still struggle with

Models	ChartBench			ChartQA			ChartX	ChartCoF
	Reg.	Extra	Avg.	Human	Aug.	Avg.	NQA	-
GPT4o (Achiam et al., 2023)	60.02	**58.89**	59.45	-	-	84.70	46.60	60.23
Gemini-1.5-Flash (Team et al., 2024)	49.05	41.79	45.76	60.16	85.68	72.92	47.31	57.13
ChartVLM-14.3B (Xia et al., 2024)	15.16	8.38	11.96	42.08	82.48	62.28	40.71	21.78
ChartLlama-13B (Han et al., 2023)	20.99	21.71	21.31	58.40	93.12	75.76	13.80	-
ChartGemma-3B (Masry et al., 2024b)	39.89	42.27	38.46	67.84	85.28	76.56	35.15	30.67
TinyChart-3B (Zhang et al., 2024a)	26.71	22.56	22.51	70.24	91.04	76.80	40.10	31.63
ChartAst-13B (Meng et al., 2024)	3.82	1.58	2.81	64.88	93.12	79.00	30.99	-
ChartMoE-8B (Xu et al., 2024)	56.31	55.58	51.67	78.32	90.96	84.64	46.62	42.80
InternVL-2.5-8B (Chen et al., 2024b)	62.23	41.73	52.96	75.20	**94.56**	84.88	52.26	50.65
InternVL-2.5-8B + **ChartCoF**	68.44	53.14	61.52 (+8.56)	77.12	94.48	85.80 (+1.00)	57.47 (+5.19)	71.95 (+21.3)
+self-consistency	**70.72**	56.61	**64.33** (+11.37)	**78.64**	94.40	**86.32** (+1.56)	58.94 (+6.68)	73.81 (+23.16)
Qwen2VL-7B (Wang et al., 2024a)	63.13	56.23	60.01	73.28	94.40	83.84	52.17	49.55
Qwen2VL-7B + **ChartCoF**	67.01	55.35	61.73 (+1.72)	76.00	93.76	84.88 (+1.04)	**59.64** (+7.47)	75.12 (+25.57)
+self-consistency	69.10	57.71	63.94 (+3.93)	76.64	93.52	85.08 (+1.16)	59.38 (+7.21)	**76.50** (+26.95)

Table 2: Accuracy of MLLMs on ChartBench, ChartQA, ChartX, and ChartCoF. The best and second-best scores are highlighted in **bold** and underline, respectively. Performance improvements over vanilla models are present in brackets. Accuracy on ChartCoF in terms of annotation, task type, and chart type is present in Table 15 in Appendix.

Function chain taxonomies	InternVL2.5-8B	GPT4o	Gemini-1.5-Flash	Qwen2VL-7B	ChartMoE
object_selection/**value**	**83.6**	70.5	**83.6**	67.2	54.1
object_selection/**value**/object_selection/**value**/arithmetical_operation	**62.0**	44.0	48.0	44.0	46.0
object_selection/**value**/object_selection/**value**/statistics	62.2	59.5	**79.5**	61.5	56.4
object_selection/**value**/object_selection/**value**/object_selection/**value**/statistics	80.0	73.3	**83.9**	71.0	58.1
object_selection/**min_max**/value	48.5	**60.6**	59.5	45.2	33.3
object_selection/**min_max**/**text_information**	36.4	**72.7**	53.1	37.5	18.8
object_selection/**count**	71.0	**93.5**	90.3	54.8	67.7
object_selection/**filter**/**count**	44.7	**76.3**	44.7	34.2	26.3
object_selection/**text_information**	71.9	**81.3**	71.9	65.6	59.4
object_selection/**value**/object_selection/**value**/**compare**	**90.3**	80.6	83.9	87.1	80.6
object_selection/**if_match_condition**	74.1	85.2	81.5	**88.9**	77.8

Table 3: Accuracy of function chain taxonomies of MLLMs. The best and second-best scores are highlighted in **bold** and underline, respectively. The description of the function taxonomies (e.g., *value* stands for the value extraction function) can be referred to Table 18 in Appendix.

the complex reasoning questions on *ChartCoF*, and all of the MLLMs achieve low accuracy. Among them, GPT4o achieves the best performance, with an accuracy of 60.23%, a testament to its significant reasoning capabilities. We also observe that the chart-specific models achieve lower accuracy compared to other models, demonstrating the necessity of our proposed *ChartCoF* for reasoning enhancement on these complex reasoning questions. After finetuning Qwen2VL-7B with *ChartCoF*, it achieves the state-of-the-art performance.

5.3 Fine-grained Evaluation on ChartCoF

Fine-grained evaluation on different function chain taxonomies. We provide a fine-grained evaluation on different function chain taxonomies for well-performed MLLMs on Table 3. MLLMs achieve the significant performance difference in questions that possess different function chain taxonomies. Specifically, InternVL-2.5-8B and Gemini-1.5-Flash achieve a notably higher accuracy for the questions with the function *value* that stands for the value extraction function compared with GPT4o. However, GPT4o achieves significantly high accuracy on questions with *min_max*, *count*, *filter*, and *text_information* functions. In addition, Qwen2VL-7B excels in comparison (with *compare*) and condition matching functions (with *if_match_condition*). The fine-grained evaluation on function chain taxonomies illustrate the strength and weakness for MLLMs, which provides effective guidance for data selection and model training.

Fine-grained evaluation on question lengths and failure modes. We present the performance of MLLMs on questions with different lengths of function chains in Fig. 3. With the increase in lengths, MLLMs achieve lower accuracy since questions generally become difficult. When adopting CoT, the performance for the questions with long function chains is improved, and the gap across lengths of function chains is minimized. To investigate why MLLMs struggle with longer chains, we manually

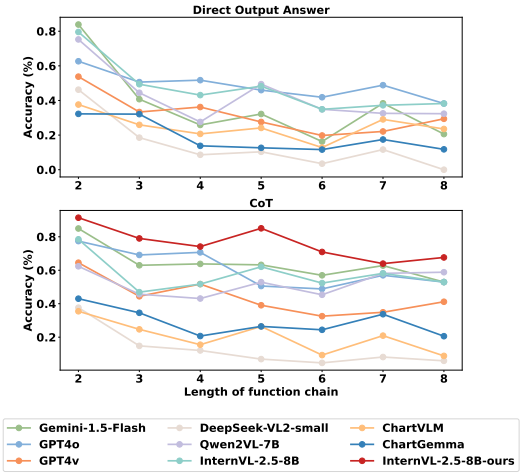


Figure 3: Accuracy of MLLMs across questions with different lengths of function chains.

Length of function chains	2	3	4	5	6	7	8	total
# Question	0	11	6	6	5	8	4	40
The step with first-occur error	2	3	4	5	6	7	8	total
# Question	15	14	6	4	1	0	0	40

Table 4: Statistics of length of function chains and step of first-occur error for questions.

analyzed 40 incorrect predictions and reasoning processes from InternVL2.5-8B. The statistics of length of function chains and the step with the first-occur error are present in Table 4. The results show that the reasoning processes generally make mistakes in the early steps during reasoning: 72.5% of first errors occur in early steps (steps 2-3). Besides, in some cases, while it can achieve correct reasoning logic, the incorrect intermediate result leads to an incorrect answer. For a question example 'What is the average number of mobile device visitors in November and December?', it achieves the correct logic but extracts an incorrect value for 'November', leading to an incorrect answer. The weak reasoning capabilities of MLLMs necessitate the accurate and diverse CoT data of *ChartCoF*.

5.4 Out-of-distribution Analysis

To further demonstrate the effectiveness of *ChartCoF*, we evaluate the OOD performance on unseen chart types and longer function chains. We finetune InternVL-2.5-8B with only the regular charts (i.e., bar, line, and pie) and evaluate accuracy in the extra test set of *ChartCoF*, ChartBench, and ChartX (i.e., removing the regular charts from these benchmarks). Results in Table 5 show that even with only the regular charts, InternVL-2.5-8B finetuned

Models	ChartCoF	ChartBench	ChartX
InternVL-2.5-8B (direct answer)	42.84	41.73	42.64
InternVL-2.5-8B (CoT)	40.00	29.07	38.19
InternVL-2.5-8B + ChartCoF (Reg.)	**51.04**	**46.40**	**43.25**

Table 5: Accuracy of MLLMs on benchmarks without regular chart types (bar, line, and pie).

Length of function chains	InternVL-2.5-8B (Vanilla)	≤ 4	≤ 5	≤ 6
ChartCoF (≥ 7)	49.75	59.10	60.10	**62.07**

Table 6: Accuracy of InternVL-2.5-8B with different training sets on the OOD test set of ChartCoF (i.e., length of function chains ≥ 7).

with *ChartCoF* improves accuracy performance on the extra test set of all these three benchmarks, demonstrating that *ChartCoF* enhances the generalized reasoning capabilities on the unseen chart types. The detailed accuracy of each chart type is discussed in Appendix F.

ChartCoF also enhances the generalization capabilities for longer function chains. We finetune InternVL-2.8-8B with the short-function-chain data (length of function chains $\leq 4, 5, 6$) and evaluate it on the long-function-chain test samples (length of function chains ≥ 7). Results on Table 6 show that short-function-chain data significantly enhance the reasoning capabilities and improve accuracy on the long-function-chain test samples. The OOD analysis demonstrates the effectiveness of *ChartCoF* in boosting generalized reasoning capabilities, which attributes to the accurate and diverse CoT data generated by our proposed *CoF* pipeline.

5.5 Comparison with Existing Datasets

Diversity comparison. We compare the function chain number in *ChartCoF* with the template number of other template-based generation methods, including ChartAst (Meng et al., 2024) and ChartQA-PoT (Zhang et al., 2024a). The comparison in Table 7 show that ChartCoF produces 3,134 function chains, significantly more than that of the existing datasets.

Accuracy comparison. We finetune InternVL2.5-8B with 5k samples in ChartQA-PoT, ChartAst, and *ChartCoF*. We also include another dataset SciGraphQA (Li and Tajbakhsh, 2023), an LLM-based generation method to prompt GPT-4 to generate Q&As. For ChartQA-PoT, we use Oracle pattern by selecting the best answer between direct and program-based outputs. Results in Table 7 show that *ChartCoF* significantly outperforms the

	#Function chains/templates	ChartQA	ChartX
SciGraphQA	-	84.40	46.18
ChartQA-PoT (Oracle)	40	85.16	51.38
ChartAst	101	83.96	47.66
ChartCoF	**3,134**	**85.84**	**56.94**

Table 7: Comparison of number of function chains/templates and accuracy between datasets.

other three datasets on ChartQA and ChartX, explicitly demonstrating its superiority in enhancing reasoning capabilities for MLLMs.

We further evaluate the linguistic transfer accuracy and logic alignment between questions and rationales in Appendix D. The comparison with the position-aware dataset Evochart (Huang et al., 2024) and joint training with existing datasets are present in Appendix C.

6 Conclusion

In this work, to overcome the scarcity of high-quality reasoning data for fine-grained evaluation and enhancement of chart reasoning capabilities, we proposed *chain of functions (CoF)*, which utilized two key processes, namely *program-based functional discovery* and *reverse linguistic CoT data synthesis*, to generate accurate and diverse reasoning data. Employing *CoF*, we introduced *ChartCoF*, which enables the fine-grained evaluation on different reasoning questions and enhances the reasoning capabilities for chart understanding. We believe that the ideas of *functional discovery* and *first exploration then task generation* in *CoF* have the potential to extend to other step-wise tasks, such as mathematical Q&A and graphical user interface tasks.

7 Limitations

We summarize the limitations of our work as below: 1) The current research emphasizes the critical role of chart data accuracy in the reasoning process for chart understanding. Consequently, we have chosen to represent charts using JSON data, rather than extracting charts directly from websites (Wang et al., 2024b; Masry et al., 2022). Despite our conscientious efforts to craft code templates specific to each chart type and the incorporation of diverse code libraries to increase the variety of charts, there remains a discernible difference between our synthesized charts and those that are naturally occurring on the internet. Future research could explore methodologies for the precise extraction of information from web-based charts or for the advancement of chart rendering techniques. Such innovations could narrow the existing chasm and enhance the reasoning proficiency of MLLMs.

2) Our approach leverages function chains as supervisory signals and employs LLMs as translators to generate reasoning data. Nevertheless, LLMs may still produce questions or rationales that are not entirely accurate on occasion. To ensure higher data quality, future efforts could focus on developing mechanisms to filter out these inaccuracies using state-of-the-art MLLMs. This would further refine the data generation process and enhance the reliability of the reasoning tasks performed by MLLMs.

8 Use of Large Language Models

The Large Language Models (LLMs) were employed to refine grammar and improve the clarity of the text. Furthermore, LLMs function as the agents for the open-ended deep research. The authors have reviewed all LLM-generated contributions and take full responsibility for the content and integrity of this work.

References

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

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

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

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023a. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023b. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*.

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.

Jinyue Chen, Lingyu Kong, Haoran Wei, Chenglong Liu, Zheng Ge, Liang Zhao, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. 2024a. Onechart: Purify the chart structural extraction via one auxiliary token. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 147–155.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. 2024b. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. 2024c. 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.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Yuhao Dong, Zuyan Liu, Hai-Long Sun, Jingkang Yang, Winston Hu, Yongming Rao, and Ziwei Liu. 2024. Insight-v: Exploring long-chain visual reasoning with multimodal large language models. *arXiv preprint arXiv:2411.14432*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Wan-Cyuan Fan, Yen-Chun Chen, Mengchen Liu, Lu Yuan, and Leonid Sigal. 2024. On pre-training of multimodal language models customized for chart understanding. *arXiv preprint arXiv:2407.14506*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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, Zhiheng Xi, Wanxu Zhao, Xiaoran Fan, Yiwen Ding, Zifei Shan, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Distill visual chart reasoning ability from llms to mllms. *arXiv preprint arXiv:2410.18798*.

Wenyi Hong, Weihan Wang, Ming Ding, Wenmeng Yu, Qingsong Lv, Yan Wang, Yean Cheng, Shiyu Huang, Junhui Ji, Zhao Xue, et al. 2024. Cogvlm2: Visual language models for image and video understanding. *arXiv preprint arXiv:2408.16500*.

Muye Huang, Lai Han, Xinyu Zhang, Wenjun Wu, Jie Ma, Lingling Zhang, and Jun Liu. 2024. Evochart: A benchmark and a self-training approach towards real-world chart understanding. *arXiv preprint arXiv:2409.01577*.

Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In *International Conference on Machine Learning*, pages 18893–18912. PMLR.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. 2024a. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

Shengzhi Li and Nima Tajbakhsh. 2023. Scigraphqa: A large-scale synthetic multi-turn question-answering dataset for scientific graphs. *arXiv preprint arXiv:2308.03349*.

Zhuowan Li, Bhavan Jasani, Peng Tang, and Shabnam Ghadar. 2024b. Synthesize step-by-step: Tools templates and llms as data generators for reasoning-based chart vqa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13613–13623.

Fangyu Liu, Julian Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin Altun. 2023a. Deplot: One-shot visual language reasoning by plot-to-table translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10381–10399.

Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Eisenschlos. 2023b. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12756–12770.

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)*, pages 1287–1310.

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

Mengsha Liu, Daoyuan Chen, Yaliang Li, Guian Fang, and Ying Shen. 2024c. Chartthinker: A contextual chain-of-thought approach to optimized chart summarization. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 3057–3074.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, et al. 2024. Deepseek-vl: towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq 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, Parsa Kavehzadeh, Xuan Long Do, Enamul Hoque, and Shafiq Joty. 2023. Unichart: A universal vision-language pretrained model for chart comprehension and reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14662–14684.

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

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 *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1527–1536.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Lingdong Shen, Kun Ding, Gaofeng Meng, Shiming Xiang, et al. 2024. Rethinking comprehensive benchmark for chart understanding: A perspective from scientific literature. *arXiv preprint arXiv:2412.12150*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

InternLM Team. 2023. Internlm: A multilingual language model with progressively enhanced capabilities.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Peifang Wang, Olga Golovneva, Armen Aghajanyan, Xiang Ren, Muhao Chen, Asli Celikyilmaz, and Maryam Fazel-Zarandi. 2023. Domino: A dual-system for multi-step visual language reasoning. *arXiv preprint arXiv:2310.02804*.

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

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, et al. 2024b. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. 2024. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*.

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*.

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. 2024. Chartmoe: Mixture of expert connector for advanced chart understanding. *arXiv preprint arXiv:2409.03277*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. 2023. A survey on multimodal large language models. *arXiv preprint arXiv:2306.13549*.

Liang Zhang, Anwen Hu, Haiyang Xu, Ming Yan, Yichen Xu, Qin Jin, Ji Zhang, and Fei Huang. 2024a. Tinychart: Efficient chart understanding with visual token merging and program-of-thoughts learning. *arXiv preprint arXiv:2404.16635*.

Pan Zhang, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Rui Qian, Lin Chen, Qipeng Guo, Haodong Duan, Bin Wang, Linke Ouyang, et al. 2024b. Internlm-xcomposer-2.5: A versatile large vision language model supporting long-contextual input and output. *arXiv preprint arXiv:2407.03320*.

Ruohong Zhang, Bowen Zhang, Yanghao Li, Haotian Zhang, Zhiqing Sun, Zhe Gan, Yinfei Yang, Ruoming Pang, and Yiming Yang. 2024c. Improve vision language model chain-of-thought reasoning. *arXiv preprint arXiv:2410.16198*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhuosheng Zhang, Aston Zhang, Mu Li, George Karypis, Alex Smola, et al. 2024d. Multimodal chain-of-thought reasoning in language models. *Transactions on Machine Learning Research*.

Chart types		Training set		Test set	
		#charts	#Q&As	#charts	#Q&As
Regular	Bar_multi	1819	6050	60	165
	Bar_single	1516	6052	60	153
	Bar_stacked	1868	6052	57	146
	Line_multi	1541	3050	53	123
	Line_single	1532	3046	62	119
	Pie	655	803	37	75
Extra	Radar	104	353	25	50
	Rings	514	2050	30	50
	Rose	274	1244	25	50
	3D-Bar	611	2054	23	51
	box	627	2050	23	50
	funnel	964	2054	20	50
	heatmap	696	2055	19	50
	area	1007	2050	27	50
	bubble	1120	3107	28	80
	node link	1040	2101	34	50
	candlestick	562	2050	17	50
	treemap	989	2052	30	50
	multi-axes	910	2052	18	39
Total		18349	50329	648	1451

Table 8: Detailed quantity statistics of each chart type for training and test sets of *ChartCoF*.

A Data Splitting

ChartCoF dataset encompasses a total of 18,349 charts and 50,329 Q&As in the training set, and 648 charts along with 1,451 Q&As in the test set. The detailed quantity statistics for training and test sets of *ChartCoF* are presented in Table 8. *ChartCoF* is meticulously categorized into two distinct groups: Regular and Extra chart types. Within the Regular category, there are six different chart types, with Bar_multi, Bar_single, and Bar_stacked leading in quantity for the training set, comprising 1,819, 1,516, and 1,868 charts, respectively. These three types also contribute to a substantial proportion of Q&As, with each type exceeding 6,000 Q&As. The Extra category encompasses a wider variety of 13 chart types for better generalization on chart types, which covers the chart types of existing benchmarks ChartBench (Xu et al., 2023) and ChartX (Xia et al., 2024). Compared with Regular charts, the quantity of charts and Q&As for each Extra type is slightly lower. This comprehensive collection allows for robust training and effective evaluation of chart comprehension models, providing extensive coverage across a diverse range of chart types and complexity levels.

B Dataset Comparison

We provide a detailed comparison between *ChartCoF* and existing datasets from the aspects of evaluation and quality of training data, as presented in Table 10. ChartQA (Masry et al., 2022), SCICQA (Shen et al., 2024), and ChrXiv (Wang et al., 2024b) provide reasoning questions with the charts from webs, where the questions are annotated by humans. Despite the delicate charts and reasoning questions, the barier of human annotations makes them hard to scale to the training set. Besides, these benchmarks only provide a coarse evaluation with an accuracy metric. To scalably generate instruction data, some studies, including MMC (Liu et al., 2024a), ChartBench (Xu et al., 2023), ChartX (Xia et al., 2024), ChartLlama (Han et al., 2023), ChartInstruct (Masry et al., 2024a), ChartGemma (Masry et al., 2024b), CHOPINLLM (Fan et al., 2024), and REACHQA (He et al., 2024), have utilized extremely large (M)LLMs to generate reasoning instructions. However, the autoregressive generation and fix-pattern prompts for generation limit precision and diversity of generated instructions. Although EvoChart (Huang et al., 2024), PlotQA (Methani et al., 2020), ChartAst (Meng et al., 2024), and LAMENDA (Li et al., 2024b) have manually set up program or function templates to ensure the precision of instructions, the predefined templates still suffer from the low diversity of instructions, and they cannot provide the linguistic rationales for enhancing the reasoning capabilities. Overall, compared with existing datasets, *ChartCoF* provides more diverse and accurate reasoning data for enhancing the reasoning capabilities and fine-grained evaluation on the varying question taxonomies.

	ChartQA	ChartBench
w.o. rationale refinement	84.64	58.15
with rationale refinement	85.88	61.52

Table 9: Ablation study of the effectiveness of rationale refinement for InternVL-2.5-8B on ChartQA and ChartBench.

C Empirical Comparison with Existing Datasets

As discussed in Appendix B, *ChartCoF* provides more diverse and accurate reasoning data compared existing datasets. We further empirically verify these two points in terms of diversity and generalization performance.

Evaluation on EvoChart (Huang et al., 2024). We evaluate the ChartCoF on the EvoChart dataset (Huang et al., 2024), which emphasizes positional questions. We select InternVL2-8B (Chen et al.,

Dataset	Chart properties		Q&A properties						
	#Chart Types	Repre. Format	Func. Usage	Func. Scal.	Rea. Q&A	Lingui. Rat.	Func. Lengths Eval.	Ques. Tax. Eval.	Annotators
ChartQA (Masry et al., 2022)	3	Table	✗	-	✔	✗	✗	✗	Human
SCI-CQA (Shen et al., 2024)	21	-	✗	-	✔	✗	✗	✗	Human
CharXiv (Wang et al., 2024b)	-	-	✗	-	✔	✗	✗	✗	Human
MMC (Liu et al., 2024a)	6	Caption	✗	-	✔	✔	✗	✗	GPT-4
ChartBench (Xu et al., 2023)	9	Table	✗	-	✗	✗	✗	✗	GPT3.5
ChartX (Xia et al., 2024)	18	Table	✗	-	✔	✗	✗	✗	GPT-4
ChartLlama (Han et al., 2023)	10	Table	✗	-	✔	✔	✗	✗	GPT-4
ChartInstruct (Masry et al., 2024a)	-	Table	✗	-	✔	✔	✗	✗	GPT-4
ChartGemma (Masry et al., 2024b)	-	-	✗	-	✔	✔	✗	✗	Gemini Flash-1.5
CHOPINLLM (Fan et al., 2024)	18	JSON	✗	-	✔	✔	✗	✗	GPT-4
REACHQA (He et al., 2024)	10	Code	✗	-	✔	✔	✗	✗	GPT4o
EvoChart (Huang et al., 2024)	4	Code	✔	✗	✔	✗	✗	✗	GPT-4
PlotQA (Methani et al., 2020)	3	Table	✔	✗	✔	✗	✗	✗	-
ChartAst (Meng et al., 2024)	9	Table	✔	✗	✔	✗	✗	✗	-
LAMENDA (Li et al., 2024b)	3	Table	✔	✗	✔	✗	✗	✗	-
ChartCoF (ours)	19	JSON	✔	✔	✔	✔	✔	✔	Qwen2.5-32B

Table 10: Comparison between *ChartCoF* and existing chart-related datasets. Abbreviations: Repre.=Representation, Scal.=Scalability, Rea.=Reasoning, Lingui.=Linguistic, Rat.=Rationale, Func.=Function Ques.=Questions Tax.=taxonomy. *ChartCoF* enables accurate and diverse reasoning data via scalable function usage and additionally provides fine-grained evaluation on different function lengths and question taxonomies.

2024c) to eliminate the interference of backbone models since EvoChart used InternVL2-8B as the baseline backbone and Phi3-Vision-4B (Abdin et al., 2024) is better than InternVL2-8B on EvoChart. The results in Table 11 show that ChartCoF notably improves accuracy performance for InternVL2-8B on all these three datasets. Even though domain-specific EvoChart-4B outperforms our model on its native benchmark EvoChart due to the similar distribution of questions, our model still significantly outperforms EvoChart-4B on ChartQA and ChartX benchmarks, demonstrating its effectiveness in enhancing reasoning capabilities on OOD benchmarks.

	EvoChart	ChartQA	ChartX
EvoChart-4B (Phi3-Vision-4B)	**54.2**	81.5	40.1
InternVL2-8B	38.6	83.3	43.7
InternVL2-8B + ChartCoF	48.2	**83.8**	**54.9**

Table 11: Accuracy comparison with EvoChart.

ChartCoF is complementary with perception-focused datasets. ChartCoF focuses on generating accurate and diverse reasoning data—a gap in existing datasets like ChartQA (Masry et al., 2022) and MMC (Liu et al., 2024a), which prioritize natural charts but overlook accurate reasoning processes. We emphasize that ChartCoF is not a replacement for natural chart training but a complementary resource. We progressively finetune InternVL2.5-8B with ChartQA, MMC, and ChartCoF. The results in Table 12 demonstrate that joint training ChartCoF with natural chart datasets can synergistically improve both perception and reasoning.

	ChartQA	ChartX
InternVL2.5-8B	84.88	52.26
ChartQA + MMC (50k)	86.04	49.83
ChartCoF	85.88	**57.47**
ChartCoF + ChartQA + MMC (50k)	**87.72**	**57.47**

Table 12: Accuracy of InternVL2.5-8B with varying training data.

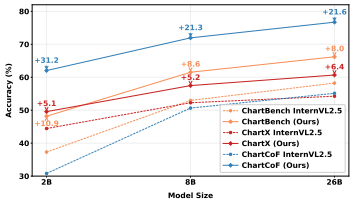


Figure 4: Accuracy of InternVL2.5 series (2B, 8B, and 26B) on ChartBench, ChartX and ChartCoF.

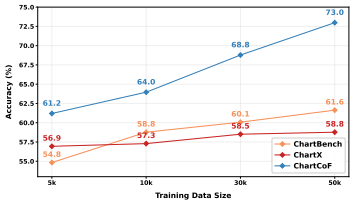


Figure 5: Accuracy of InternVL2.5-8B on ChartBench, ChartX and ChartCoF.

D Data Quality Evaluation

We utilize GPT4o as a judge to verify the correctness of 200 random samples and present results

	Consistency between function chains and linguistic rationales	Consistency between rationales and questions
Accuracy	100%	97.5%

Table 13: Accuracy evaluation for generated CoT data with GPT4o. We leverage the consistency between function chains and linguistic rationales to evaluate the effectiveness of rationale translation and the consistency between rationales and questions to evaluate the logic accuracy of CoT data.

in Table 13. We leverage the consistency between function chains and linguistic rationales to evaluate the correctness of rationale translation and the consistency between rationales and questions to evaluate the logic accuracy of CoT data. The corresponding prompts are shown as follows:

Prompt of assessment for alignment between function chains and linguistic rationales.

```
You are provided with a program process and the
      linguistic rational process. Your task is to
      evaluate whether the linguistic rationale is
      consistent with the program process in terms
      of intermediate reasoning processes and final
      answer. Please respond with "True" if they are
       consistent, or "False" if they are not.

Program process:
{program}

Linguistic rationale process:
{rationale}

Assessment:
Please respond with "True" if they are consistent,
      or "False" if they are not.
```

Prompt of assessment for alignment between rationales and questions.

```
You are provided with a question, a rationale, and
      an answer. Your task is to evaluate whether
      the logic of rationale is correct to the
      question. Please respond with "True" if the
      rationale and answer can be used to
      definitively and correctly response to the
      question, or "False" if they can not.

Question: {question}

Rationale: {rationale}

Answer: {answer}

Please respond with "True" if the rationale and
      answer can be used to definitively and
      correctly response to the question, or "False"
       if they can not.
```

Results in Table 13 show that ChartCoF achieves 100% accuracy for the alignment between function chains and linguistic rationales, demonstrating the translation correctness of ChartCoF. Besides, ChartCoF achieves 97.5% for the alignment between rationales and questions, which demonstrates the logic correctness of ChartCoF.

E Effectiveness of Rationale Refinement

To enhance the effectiveness of rationale refinement, we conduct the ablation study for it. We finetune InternVL-2.5-8B using the same number of samples without rationale refinement and evaluate it on ChartQA and ChartBench. Results on Table 9 show that after rationale refinement, the performance of InternVL-2.5-8B is improved on ChartQA and ChartBench. A comparison example is presented in Example E.1. Before rationale refinement, the rationale manuscript is still function-like and redundant due to the translation from function chains. After refinement, the rationale becomes more fluent and information-intensive, which better aligns with the existing MLLMs and enhances their reasoning capabilities.

E.1 Example of rationale refinement

Example of rationale refinement

```
Question:
According to the chart, which legend is associated
      with the light green data points representing
      satisfaction scores from customer feedback
      across different service providers?

Rationale manuscript:
First, we select all the data points that are
      represented in light green on the chart, which
       corresponds to 'Client Feedback'. These
      points are from 'Provider A' to 'Provider I'.
      Then, we identify the legend associated with
      these data points, which is 'Client Feedback'.
       Final answer: Client Feedback

After rationale refinement:
The data points shown in light green on the chart
      represent 'Client Feedback' from different
      service providers, specifically from Provider
      A to Provider I. The legend associated with
      these light green data points is 'Client
      Feedback'. Final answer: Client Feedback
```

F Fine-grained Generalization Analysis on Chart Types

We present the detailed accuracy improvement on ChartBench in terms of OOD chart types in Table 14. The results show that ChartCoF generalizes to 5 of 6 OOD chart types and significantly improves accuracy except for node-link charts. This is because node-link charts (e.g., food chain charts) demand structural reasoning about connections between elements (e.g., identifying bridges or neighbors), a

	area	box	radar	scatter	node-link	combination	all
InternVL-2.5-8B (direct answer)	33.06	16.93	43.60	44.80	**77.80**	44.60	41.73
InternVL-2.5-8B + ChartCoF (Reg.)	**43.46**	**27.33**	**44.60**	**61.07**	43.60	**55.10**	**46.40**

Table 14: Accuracy improvement on ChartBench in terms of OOD chart types with training on common chart types (bar, line, and pie).

paradigm distinct from the pattern-based or numerical tasks required by common charts. The depth analysis of generalization on detailed chart types, together with the OOD analysis in Section 5.4, explicitly demonstrates that the reasoning capabilities enhanced by ChartCoF can generalize to those unseen chart types, which is attributed to the accurate and diverse CoT data generated by our proposed data generation pipeline CoF.

G Model and Data Scalability

To further demonstrate the effectiveness of our proposed generation pipeline, we finetune different sizes of InternVL2.5 models and evaluate them on ChartBench, ChartX, and *ChartCoF*. The results in Fig. 4 show that, with the increasing model parameters from 2B to 26B, the accuracy of InternVL2.5 models keep increasing on ChartBench, ChartX, and *ChartCoF*. Meanwhile, after finetuning with the training set of our proposed *ChartCoF*, all these three levels of InternVL2.5 models significantly outperform the base models without finetuning. Remind that we utilize only Qwen2.5-32B-Instruct for data generation. The notable improvement of the same-level model (i.e., InternVL2.5-26B) on benchmarks demonstrate that our data generation pipeline *CoF* provide valuable supervision on data generation instead of only knowledge distillation from large models into small models.

To demonstrate the effectiveness of *ChartCoF* on data scalability, we finetune InternVL2.5-8B with varying training data sizes on ChartBench, ChartX, and *ChartCoF*. The results in Fig. 5 show that, with the increasing of the training data sizes, the accuracy of InternVL2.5-8B keeps increasing on these three benchmarks. The effectiveness of *ChartCoF* on data scalability reveals the potential of *CoF* on generating larger scales of data to further improve the reasoning capabilities of MLLMs.

H JSON Templates

Without specific statements on chart types, the general JSON templates for charts are presented in Appendix H.1. The JSON templates for box, candlestick, and node link charts are presented in Appendices H.2, H.3, and H.4, respectively.

H.1 JSON Elements for Charts

JSON elements

```
{
"title": {title},
"x_label": {x_label},
"y_label": {y_label},
"type": {type},
"legend_num": {legend_num},
"legends": [legend list],
"group_num": {group_num},
"groups": [group list],
"colors": {colors},
"data_points":
    {
    "group1": {
        legend1: xxxx,
        legend2: xxxx
        },
    },
"legend_colors":
    {
    "legend1": "color1",
    "legend2": "color2",
    },
}
```

H.2 JSON Elements for Box

JSON elements for box

```
{
"title": {title},
"x_label": {x_label},
"y_label": {y_label},
"type": {type},
"legend_num": {legend_num},
"legends": [legend list],
"group_num": {group_num},
"groups": [group list],
"colors": {colors},
"legend_colors":
    {
    "legend1": "color1",
    "legend2": "color2"
    },
"median": {"legend1": xxx},
"first_quartile": {"legend1": xxx},
"third_quartile": {"legend1": xxx},
"minimum_values": {"legend1": xxx},
"maximum_values": {"legend1": xxx},
"outlier_values": {"legend1": xxx}
}
```

Models	Annotation		Task			Chart type		Avg.
	w.o.	w.	Binary	NQA	Text	Regular	Extra	
proprietary models								
GPT4o (Achiam et al., 2023)	42.16	76.85	81.51	55.74	57.46	65.17	54.48	60.23
GPT4V (Achiam et al., 2023)	26.62	59.26	68.49	39.59	35.09	46.86	39.85	43.63
Gemini-1.5-Flash (Team et al., 2024)	44.46	68.78	80.67	54.31	44.74	64.79	48.21	57.13
Open-sourced models								
InternLM-XComposer-2.5-7B (Zhang et al., 2024b)	34.67	51.72	61.34	42.34	30.26	50.19	35.82	43.56
DeepSeek-VL2-small (Wu et al., 2024)	18.41	24.87	55.04	12.18	28.51	20.49	23.28	21.78
LLaVA-v1.6-mistral-7B (Liu et al., 2024b)	22.73	30.16	50.84	21.92	21.49	27.53	25.53	26.60
Qwen2VL-7B (Wang et al., 2024a)	39.28	58.99	78.15	44.77	40.35	55.19	42.99	49.55
InternVL-2.5-8B (Chen et al., 2024b)	36.98	63.23	69.33	48.63	39.91	59.80	40.00	50.65
CogVLM2-7B (Hong et al., 2024)	25.47	46.43	65.97	32.18	23.68	37.90	34.63	36.39
Chart-specific models								
ChartInstruct-7B (Masry et al., 2024a)	13.52	16.01	55.88	7.92	1.75	13.96	15.82	14.82
ChartVLM-14.3B (Xia et al., 2024)	20.29	23.15	49.16	18.48	7.46	24.07	19.10	21.78
ChartGemma-2B (Masry et al., 2024b)	25.04	35.85	58.40	26.90	17.98	35.08	25.52	30.67
ChartMoE-8B (Xu et al., 2024)	34.96	50.00	72.27	38.17	32.02	47.50	37.31	42.80
InternVL-2.5-8B + **ChartCoF**	**63.74**	**79.50**	**89.50**	**68.63**	**67.98**	**77.85**	**65.07**	**71.95**

Table 15: Accuracy performance of MLLMs with CoT prompts on **ChartCoF**. The best and second-best scores are highlighted in **bold** and underline, respectively.

H.3 JSON Elements for Candlestick

JSON elements for Candlestick

```
{
"title": {title},
"x_label": {x_label},
"y_label": {y_label},
"type": {type},
"legend_num": {legend_num},
"legends": [legend list],
"group_num": {group_num},
"groups": [group list],
"colors": {colors},
"legend_colors":
    {
    "legend1": "color1",
    "legend2": "color2"
    },
"opening_price": {"legend1": xxx},
"closing_price": {"legend1": xxx},
"highest_price": {"legend1": xxx},
"lowest_price": {"legend1": xxx}
}
```

H.4 JSON Elements for Node Link

JSON elements for Node Link

```
{
"title": {title},
"x_label": {x_label},
"y_label": {y_label},
"type": {type},
"legend_num": {legend_num},
"legends": [legend list],
"group_num": {group_num},
"groups": [group list],
"colors": {colors},
"data_points":
    {
    "group1": {legend1: [pointed_object_list_1]},
    "group2": {legend1: [pointed_object_list_1]},
    },
"legend_colors":
    {
    "legend1": "color1",
    "legend2": "color2"
    }
}
```

I Prompts Usage

The prompts for JSON seed generation and JSON evolement are presented in Prompts I.1 and I.2, respectively. The prompts for rationale generation, question generation, and rationale refinement are presented in Prompts I.3, I.4, and I.5, respectively. The prompt for answer extraction is presented in Prompt I.6. The prompt for assessing the correctness between function chains and linguistic rationales and the alignment between rationales and questions are presented in Prompt I.7 and I.8, respectively.

I.1 Prompt for JSON Seed Generation

Prompt for JSON Seed Generation

```
You are a language model tasked with generating augmented datasets to train machine learning models for chart
      understanding. These models need to be exposed to various chart configurations, data patterns, and types to
      perform accurately in diverse scenarios.
Given a JSON template that contains the basic information for a chart, your task is to fill in the missing details to
       generate a new JSON data.

Instructions:
1. The title, type, colors, legend_num, and group_num are given, and you need to add x_label, y_label, data_points,
      legends, and groups.
2. Ensure that the augmented data is diverse and realistic.
3. Maintain the structure and integrity of the original data.
4. According to the legend_num and group_num, generate the corresponding legends and groups.
5. Assign the colors in "colors" to each legend.

The original JSON data is as follows:
{JSON element file}

The output format should be: JSON Data 1: <Augmented JSON data 1>.
Only output the augmented JSON data that can be directly used to generate the chart.
```

I.2 Prompt for JSON Evolement

Prompt for JSON Evolement

```
You are a language model tasked with generating augmented datasets to train machine learning models for chart
      understanding. These models need to be exposed to various chart configurations, data patterns, and types to
      perform accurately in diverse scenarios.
Given a JSON script, your task is to correct and enrich the JSON data to generate a new JSON data.

Instructions:
1. Title: change the title of the chart to make it more descriptive and informative to the type.
2. x_label and y_label: change the x_label and y_label to make them more compatible to the title.
3. Data points: if the data points are not satisfying with the type, title, x_label, and y_label, recorrect the data
      points to make them more realistic. Your can add some noise to the data points to make them more diverse.
4. Legends: keep the legend_num unchanged. Change the legends to make them more informative and diverse.
5. Groups: change the group_num and groups to make them more diverse and informative. Make sure that the length of
      groups is the same as the group_num.
6. Colors: change the colors of the chart to make it more visually appealing and informative. Make sure that the
      colors are different and sampled from {color_list}, and the color number should be the same as the legend_num.
7. Save the new JSON data as {data_save_path}.

The original JSON data is as follows:
{json_data}

The output format should be: JSON Data 1: <Augmented JSON data 1>.
Only output the augmented JSON data that can be directly used to generate the chart.
```

I.3 Prompt for Rationale Generation

Prompt for Rationale Generation

```
You are an AI assistant specialized in translating technical reasoning processes into clear, natural language
      explanations for chart reasoning. You will be given the JSON data of the chart and a structured description of a
       chart understanding process, which includes inputs, functions, and outputs. Your task is to convert this
      structured information into a coherent, easy-to-understand paragraph.

Please follow these guidelines to generate rationale with natural language:

1. Before the reasoning process, different legends, categories, or colors are sampled. You should take them as
      conditions.
2. The reasoning processes should be related to chart understanding.
3. Describe the purpose and action of each function in simple terms.
4. When the function is related to the values of data, list all the values of the data.
5. When the function is related to the numerical calculation, you should provide calculation process and the final
      answer uising numerical operations, e.g., A + B = D, A - B = D, A * B = D, A / B = D, (A + B + C) / 3 = D etc.
6. Some functions that related to position, like left, right, top, bottom are used to render the data using the
      position information. You should emphasize the position in the rationale.
7. Some functions that related to colors are used to render the data using the color information. You should
      emphasize the color in the rationale.
8. If the function is specific to some charts, like bar, line, and pie, you should mention the chart type.
9. The final output should be the final answer.
{addition_prompt}

The JSON data of the chart:

{json_str}

Here's the structured process description: {structured process description}

Only transfer the structured process to a natural languages in short sentances.
The output format should be like:
Reasoning process: [Your reasoning process], Final answer: [Your final answer]
```

I.4 Prompt for Question Generation

Prompt for Question Generation

```
You are an AI assistant specialized in generating questions for chart reasoning. You will be given the JSON data of
      the chart, the reasoning process, and its corresponding structured description of a chart understanding process,
       which includes inputs, functions, and outputs. Your task is to generate a coherent, easy-to-understand question
       that can be answered by the reasoning process.

Please follow these guidelines:

1. Your question should follow the structured process of the chart.
2. The question can be answered by the structured process.
3. During the reasoning process, different legends, categories, or colors are used to refer data. You should consider
       them as conditions and emphasize them in the question.
4. If the rationale contains the color, you should take it as a condition and emphasize it in the question.
5. If the rationale contains the position information, like upper, bottom, left, and right, you should take them as
      conditions and emphasize them in the question.
6. The question should consider all the functions in the structured process.
7. For the length of structured process description is longer than 4 steps, you can first illustrate the conditions
      to get the data and then ask the question. You can use the patterns like "If we get a value through xxx and get
      another value through yyy, what/how/...?".
8. For the length of structured process description is shorter than 4 steps, you can directly ask the question.
9. Do not appear the important intermediate values or information (categories, legends, and colors) of data in the
      question directly since they need to be calculated by the question.

The JSON data of the chart:

{json_data}

Here's the structured process description:
{structured process description}

Here's the reasoning process in short sentences:
{rationale}

Please generate a question that can be answered by the structured process and reasoning process.
The output format should be Question: [Your question]
```

I.5 Prompt for Rationale Refinement

Prompt for Rationale Refinement

```
You are an AI assistant specialized in answering questions. You are given a structured process description, the
      rationale manuscript, and the question. You need to answer the question according to the structured process
      description and the rationale manuscript.

The structured process description is as follows:
{structured process description}

The question is as follows:
{question}

The rationale manuscript is as follows:
{rationale}

You should answer the question under the following constraints:
1. Imagine that you are answering the question about charts in a real-world scenario. You answer should be related to
       the chart understanding.
2. You should first answer the question step by step to generate rationale by taking the structured process
      description as evidence, but "structured process description" should not be mentioned in the answer.
3. The answer should be consistent with the structured process description.
4. You should keep the rationale fluent, understandable, and concise.
5. You can fuse the structured process description and the rationale manuscript to make the answer more
      understandable and concise.
6. You should remove the personal pronoun and focus on the elements that are related to the question.
7. If there are some numerical values in the reasoning processes, try to maintain the numerical values in the answer
      to make the answer more accurate.
8. If there are calculations in the reasoning processes, you should use the mathematical symbols in the natural
      language description to improve the readability.

The output format should be like:
Rewritten rationale: [Your rewritten rationale], Final answer: [Your final answer]
```

I.6 Prompt for Answer Extraction

Prompt for Answer Extraction

```
Please extract the answer from the model response and type it.

Note:
1. The responses may be a phrase, a number, or a sentence.
2. If the content of the responses is not understandable, return "FAILED".
3. If the content of the responses is understandable, extract the numerical value from it.
4. If the responses is a yes or no judgment, return yes or no.
5. If the answer contains a unit, please exclude the unit and only return the numerical value.

Special requirements: ** Only numbers, short texts, "FAILED", or yes/no are allowed to be returned for each response,
       please do not return anything else! **

Please read the following example.

Question 1: Which number is missing?
Model response: The number missing in the sequence is 14.

Question 2: What is the fraction of females facing the camera?
Model response: The fraction of females facing the camera is 0.6, which means that six out of ten females in the
      group are facing the camera.

Question 3: How much money does Luca need to buy a sour apple candy and a butterscotch candy? (Unit: $)
Model response: Ax00 Ax00 Ax00 Ax00 Ax00 Ax00 Ax00 Ax00 Ax00 Ax00 Ax00.

Question 4: In the chart titled \"Quarterly Sales Breakdown by Product Category\", if we identify the product
      category with the second lowest sales value for Q1 2023, what is the color associated with that category?
Model response: The product category with the second lowest sales value for Q1 2023 is Jewelry. The color associated
      with that category is gray.

Question 5: Which month shows the smallest difference in visitors between mobile devices and desktop devices?
Model response: The difference in visitors between mobile devices and desktop devices is the smallest in Apr.

Your answer:
14
0.6
FAILED
gray
Apr

Question: {}
Model response: {}
Expected answer:
```

I.7 Prompt of Assessment for Correctness between Function Chains and Linguistic Rationales

Prompt of Assessment for correctness between function chains and linguistic rationales

```
You are provided with a program process and the linguistic rational process. Your task is to evaluate whether the
      linguistic rationale is consistent with the program process in terms of intermediate reasoning processes and
      final answer. Please respond with "True" if they are consistent, or "False" if they are not.

Program process:
{program}

Linguistic rationale process:
{rationale}

Assessment:
Please respond with "True" if they are consistent, or "False" if they are not.
```

I.8 Prompt of Assessment for Alignment between Rationales and Questions

Prompt of Assessment for alignment between rationales and questions

```
You are provided with a question, a rationale, and an answer. Your task is to evaluate whether the logic of rationale
       is correct to the question. Please respond with "True" if the rationale and answer can be used to definitively
      and correctly response to the question, or "False" if they can not.

Question: {question}

Rationale: {rationale}

Answer: {answer}

Please respond with "True" if the rationale and answer can be used to definitively and correctly response to the
      question, or "False" if they can not.
```

J Example Comparison between Datasets

Our proposed generation pipeline *CoF* generates high-quality and diverse CoT data and provides corresponding step number and function chain. Compared with existing datasets, e.g., ChartQA, ChartAst, EvoChart, *ChartCoF* enables more difficult questions and corresponding detailed and accurate rationales. In addition, the corresponding step numbers and function chains allow the fine-grained evaluation. This can indicate the weakness and strength of MLLMs, thus providing guidance to the model training and data selection.

Examples of ChartCoF

Question: If we select the average attendance counts for 'Marketing' and 'Operations', find the second minimum value between them, then identify the leftmost department with the highest attendance and the overall second minimum attendance, what is the median of these values?
Rationale: Selecting the average attendance counts for 'Marketing' and 'Operations' yields values of 150 and 162, respectively. Among these, 'Operations' holds the second minimum value. Identifying the leftmost department with the highest attendance reveals 'Sales' at 215. The overall second minimum attendance is found in 'HR' with a count of 128. From the values 162 (Operations), 215 (Sales), and 128 (HR), the median is 162. Final answer: 162
step_num: 10
Function chain: legend_selection/second_min_object/value_of_objects/all_object_selection/leftmost_object /value_of_objects/all_object_selection/second_min_object/value_of_objects/median_of_values/

Question: If we select the customer satisfaction scores for the 'Central' and 'Southwest' regions and filter for scores higher than 91, then include the highest score across all regions and the score for the 'Northeast' region, what is the mean of these selected scores?
Rationale: Initially, the customer satisfaction scores for the 'Central' and 'Southwest' regions are selected, yielding scores of 96 and 87, respectively. These scores are then filtered to include only those higher than 91, resulting in the 'Central' region's score of 96. Next, the highest satisfaction score across all regions is identified as 96, again belonging to the 'Central' region. Additionally, the score for the 'Northeast' region is noted as 85. With these selected scores (96, 96, and 85), the mean is calculated as (96 + 96 + 85) / 3, which equals 92.33333333333333. Final answer: 92.33333333333333
step_num: 9
Function chain: legend_selection/objects_that_larger_than_value/value_of_objects/argument_selection/max_one_object /value_of_objects/one_object_selection/value_of_objects/mean_of_values/

Examples of ChartQA

Question: What is the difference in value between Lamb and Corn?
Answer: 0.57

Question: What is the difference between the highest and the lowest green bar?
Answer: 6

Examples of ChartAst

Question: What is the total number of fingerprints in the resulting database?
Answer: The resulting database is composed of two impressions of 1650 fingerprints.

Question: What is the maximum DSC among the AX, CO, and SA planes?
Answer: The maximum DSC among the AX, CO, and SA planes is 87.65.

Examples of EvoChart

Question: How many U.S. eligible voters are there in year 2014?
Answer: 25.5

Question: How many American adults support the government banning TikTok during September?
Answer: 38

K Object Functions and Value Functions

We adopt 6 selection methods for object selection and set up 99 object functions and 8 value functions in experiments. The detailed functions for object selection are presented in Table 16. The object functions for box, candlestick, and node link charts are presented in Tables 18, 19, 21, respectively. Without specific statements on chart types, the general object functions for charts are presented in Table 23. The value functions are presented in Table 22.

We categorize the functions into several function taxonomies according to their purpose for statistical analysis. The statistics of the function taxonomy are presented in Table 17. Among them, the most frequent function taxonomy in the test set of *ChartCoF* is "value", which stands for the value extraction functions. This is because value extraction is very common in the reasoning process of chart understanding, and numerous function chains also contain the value extraction function.

There are 123 function chain taxonomies in *ChartCoF* according to the comprehensive breakdown of function taxonomies in Tables 18, 19, 21. We list the statistics of top 20 function chain taxonomies in Table 20, which indicates a balanced distribution for different function chain taxonomies, demonstrating the question diversity of *ChartCoF*.

Object selection	Description
one_object_selection	Select one object using a group name and a legend name
group_selection	Select partial objects using a group name
legend_selection	Select partial objects using a legend name
color_selection	Select partial objects using a color
color_group_selection	Select one object using a group name and a color
all_object_selection	Select all the objects of the chart

Table 16: Overview of object selection.

Function taxonomy	Description	Percentage
value	The functions related to value extraction	43.36%
text_information	The functions related to text information of charts	4.76%
count	The functions related to counting	3.54%
min_max	The functions related to maximum or minimum values	17.61%
arithmetical_operation	The functions related to arithmetical operation	6.88%
compare	The functions related to comparison	3.63%
stat	The functions related to statistics	8.18%
filter	The functions related to filtering unsatisfied objects	4.93%
if_match_condition	The functions related to assessing if the objects or values match the conditions	2.09%
exclude_objects	The functions related to excluding the objects with some conditions	0.36%
position	The functions related to the position of objects	4.57

Table 17: The percentage for each function taxonomy in the test set of *ChartCoF*.

Function taxtonomy	Functions	description	Input conditions
text_information	color_of_objects	Return the color of the object.	len(obejcts)=1
	groups_of_object	Return the groups of the object.	one_object_selection not in function chain
	legends_of_object	Return the legend of the object.	one_object_selection and legend_selection not in function chain
value	median_of_objects	Return the median value of the boxplot.	-
	first_quartile_of_objects	Return the first quartile value of the boxplot.	-
	third_quartile_of_objects	Return the third quartile value of the boxplot.	-
	maximum_value_without_outliers	Return the maximum value of the boxplot without outliers.	-
	minimum_value_without_outliers	Return the minimum value of the boxplot without outliers.	-
	interquartile_range_of_box	Return the interquartile range of the boxplot.	len(obejcts)=1
	outlier_values_of_objects	Return the outlier values of the boxplot.	len(obejcts)=1
min_max	max_median_object	Return the object with the maximum median value of the boxplot.	len(obejcts)>1
	min_median_object	Return the object with the minimum median value of the boxplot.	len(obejcts)>1
	max_maximum_object_without_outliers	Return the object with the maximum maximum value of the boxplot.	len(obejcts)>1
	min_maximum_object_without_outliers	Return the object with the minimum maximum value of the boxplot.	len(obejcts)>1
	max_minimum_object_without_outliers	Return the object with the maximum minimum value of the boxplot.	len(obejcts)>1
	min_minimum_object_without_outliers	Return the object with the minimum minimum value of the boxplot.	len(obejcts)>1
	max_first_quartile_object	Return the object with the maximum first quartile value of the boxplot.	len(obejcts)>1
	min_first_quartile_object	Return the object with the minimum first quartile value of the boxplot.	len(obejcts)>1
	max_third_quartile_object	Return the object with the maximum third quartile value of the boxplot.	len(obejcts)>1
	min_third_quartile_object	Return the object with the minimum third quartile value of the boxplot.	len(obejcts)>1
count	num_of_outliers	Return the number of outliers of the boxplot.	len(obejcts)=1
position	leftmost_box	Return the leftmost box in the boxplot.	len(obejcts)>1
	rightmost_box	eturn the rightmost box in the boxplot.	len(obejcts)>1
	upper_box	Return the upper box in the boxplot.	len(obejcts)>1
	bottom_box	Return the bottom box in the boxplot.	len(obejcts)>1

Table 18: Overview of object functions for box charts.

Function taxtonomy	Functions	description	Input conditions
text_information	legends_of_object	Return the legend of the object.	len(obejcts)=1
value	high_price_of_object	Return the high price of the object.	len(obejcts)=1
	low_price_of_object	Return the low price of the object.	len(obejcts)=1
	open_price_of_object	Return the open price of the object.	len(obejcts)=1
	close_price_of_object	Return the close price of the object.	len(obejcts)=1
min_max	max_high_price_object	Return the object with the maximum high price.	len(obejcts)>1
	min_high_price_object	Return the object with the minimum high price.	len(obejcts)>1
	max_low_price_object	Return the object with the maximum low price.	len(obejcts)>1
	min_low_price_object	Return the object with the minimum low price.	len(obejcts)>1
	max_open_price_object	Return the object with the maximum open price.	len(obejcts)>1
	min_open_price_object	Return the object with the minimum open price.	len(obejcts)>1
	max_close_price_object	Return the object with the maximum close price.	len(obejcts)>1
	min_close_price_object	Return the object with the minimum close price.	len(obejcts)>1

Table 19: Overview of object functions for candlestick charts.

Function chain taxonomies	**Percentage (%)**
object_selection/value	7.8
object_selection/value/object_selection/value/arithmetical_operation	6.4
object_selection/min_max/value	5.4
object_selection/value/object_selection/value/statistics	5.0
object_selection/filter/count	4.9
object_selection/min_max/text_information	4.2
object_selection/text_information	4.1
object_selection/count	4.0
object_selection/value/object_selection/value/compare	4.0
object_selection/value/object_selection/value/object_selection/value/statistics	4.0
object_selection/if_match_condition	3.5
object_selection/min_max/value/object_selection/value/arithmetical_operation	2.6
object_selection/value/object_selection/min_max/value/arithmetical_operation	2.4
object_selection/value/object_selection/min_max/value/statistics	2.3
object_selection/min_max/value/statistics	1.8
object_selection/position/min_max/value	1.5
object_selection/min_max/value/arithmetical_operation	1.5
object_selection/min_max/value/object_selection/value/statistics	1.4
object_selection/position/text_information	1.3
object_selection/min_max/value/object_selection/min_max/value/statistics	1.3

Table 20: Statistics of function chain taxonomies.

Function taxonomy	Functions	description	Input conditions
text_information	legend_of_objects	Return the legends (name) of the objects	-
filter	targets_of_object	Return the target objects that the object points to with an arrow	len(obejcts)=1
	sources_of_object	Return the sourced objects that are pointed by the object with an arrow	len(obejcts)=1
	connected_objects	Return the conntected objects that are connected to the object with a line	len(obejcts)=1
if_match_condition	if_object_point_to_A	Return whether the object point to {A} with an arrow	len(obejcts)=1
	if_object_pointed_by_A	Return whether the object is pointed by {A} with an arrow	len(obejcts)=1
	if_object_connect_to_A	Return whether the object is connected to {A}	len(obejcts)=1

Table 21: Overview of objective functions for node link charts.

Function taxtonomy	Functions	description	Input conditions
stat	sum_of_values	Return the sum of the values of data: A + B + C.	len(values)>1
	mean_of_values	Return the mean of the values of data: (A + B + C) / len = D / len.	len(values)>1
arithmetical_operation	median_of_values	Return the median value of data.	len(values)>1
	A_minus_B	Return A - B.	len(values)=2
	difference_between_A_and_B	Return the difference between two data: | A - B |.	len(values)=2
	A_multiply_B	Return the product of two data: A * B.	len(values)=2
	A_divided_by_B	Return the division of two data: A / B.	len(values)=2
	multiply_constant	Return the value multiplied by a constant {constant}: A * constant.	len(values)=1
compare	A_is_larger_than_B	Return True if the value of the first data is larger than the value of the second data: A >B.	len(values)=2
	A_is_smaller_than_B	Return True if the value of the first data is smaller than the value of the second data: A <B.	len(values)=2

Table 22: Overview of value functions.

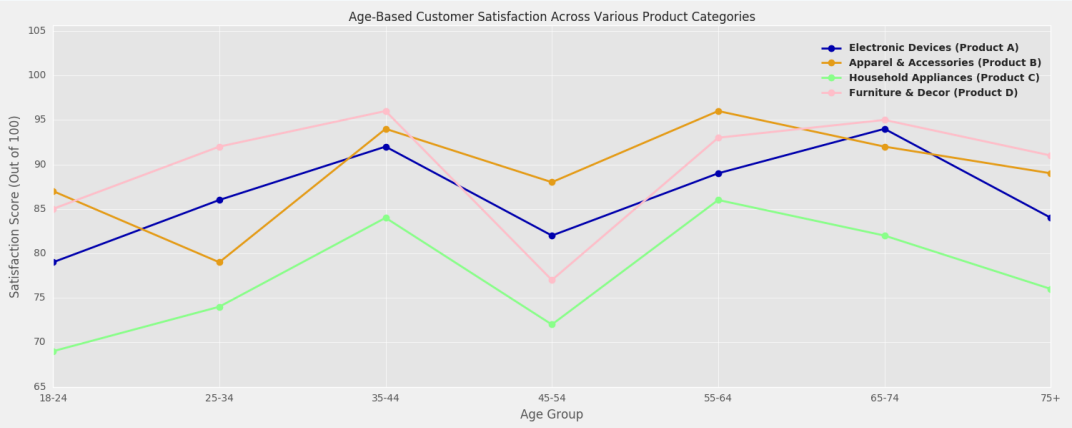
Function taxtonomy	Functions	description	Input conditions
min_max	max_one_object	Return the data with the maximum value {value}.	len(obejcts)>1
	min_one_object	Return the data with the minimum value {value}.	len(obejcts)>1
	max_two_objects	Return the two data with the maximum values {value}.	len(obejcts)>2
	min_two_objects	Return the two data with the minimum values {value}.	len(obejcts)>2
	max_three_objects	Return the three data with the maximum three values {value}.	len(obejcts)>3
	min_three_objects	Return the three data with the minimum three values {value}.	len(obejcts)>3
	second_max_object	Return the data with the second maximum value {value}.	len(obejcts)>1
	second_min_object	Return the data with the second minimum value {value}.	len(obejcts)>1
value	value_of_objects	Return the values of data.	-
text_information	color_of_objects	Return the color of data.	len(obejcts)=1, chart type not in [heatmap, 3D-Bar, bubble], color_group_selection not in function chain
	groups_of_object	Return the groups of data.	one_object_selection not in function chain
	legends_of_object	Return the legend of data.	one_object_selection not in function chain, legend_selection not in function chain
	legend_of_one_object_value	Return the legend of the specific data with value {value}.	len(obejcts)>1
	group_of_one_object_value	Return the group of the specific data with value {value}.	len(obejcts)>1
if_match_condition	if_object_that_equal_to_value	Return if the data\'s value is equal to {value}.	len(obejcts)=1
	if_object_that_larger_than_value	Return if the data\'s value is larger/more than {value}.	len(obejcts)=1
	if_object_that_smaller_than_value	Return if the data\'s value is smaller/less than {value}.	len(obejcts)=1
filter	objects_that_larger_than_value	Return data whose values are larger/more than {value}	len(obejcts)>1
	objects_that_smaller_than_value	Return data whose value are smaller/less than {value}	len(obejcts)>1
	objects_with_same_value	Return one group of data with the same value {value}.	len(obejcts)>1
count	count_of_objects	Return the number of data, with values {value}.	-
	num_of_legends	Return the number of legends used among the data, with legends {value}.	-
	num_of_colors	Return the number of colors used among the data, with colors {value}.	chart type not in [heatmap, 3D-Bar, bubble], color_group_selection not in function chain, color_selection not in function chain.
	num_of_groups	Return the number of groups used among the data, with group {group name}.	-
exclude_objects	exclude_objects_with_groups	Exclude the data with the group {group name} and return the data without the groups.	group number>1
	exclude_objects_with_legends	Exclude the data with the legends {legend name} and return the data without the legends.	legend number>1
min_max_diff_arg	the_group_that_has_maximum_difference	Return the group B that has the maximum difference between the two legends of data, with value = max(|A1-A2|, |B1-B2|, |C1-C2|) = {value}.	groun number >1
	the_group_that_has_minimum_difference	Return the group B that has the minimum difference between the two legends of data, with value = min(|A1-A2|, |B1-B2|, |C1-C2|) = {value}.	groun number >1
if_match_condition	if_objects_consistently_increase	Return if the values of the data consistently increase.	legend_selection or color_selection in function chain, chart type in [bar, line].
	if_objects_consistently_decrease	Return if the values of the data consistently decrease.	legend_selection or color_selection in function chain, chart type in [bar, line].
	if_same_values	Return if the values of the data are the same.	len(obejcts)>1
	if_same_colors	Return if the colors of the data are the same.	len(obejcts)>1
	if_same_groups	Return if the groups of the data are the same.	len(obejcts)>1
	if_same_legends	Return if the legends of the data are the same.	len(obejcts)>1
position	upper_one_bar	Return the upper-position bar in the chart.	chart type = bar
	upper_two_bars	Return the upper two-position bars in the chart.	chart type = bar
	upper_three_bars	Return the upper three-position bars in the chart.	chart type = bar
	bottom_one_bar	Return the bottom bar in the chart.	chart type = bar
	bottom_two_bars	Return the two bottom-position bars in the chart.	chart type = bar
	bottom_three_bars	Return the three bottomposition bars in the chart.	chart type = bar
	leftmost_object	Return the leftmost bars in the chart.	chart type in [bar, line]
	left_two_objects	Return the two leftmost bars in the chart.	chart type in [bar, line]
	left_three_objects	Return the thr.ee leftmost bars in the chart	chart type in [bar, line]
	rightmost_object	Return the rightmost bars in the chart.	chart type in [bar, line]
	right_two_objects	Return the two rightmost bars in the chart.	chart type in [bar, line]
	right_three_objects	Return the three rightmost bars in the chart.	chart type in [bar, line]
	upper_rightmost_object	Return the upper-rightmost bar in the chart.	chart type = bar_stacked
	upper_leftmost_object	Return the upper-leftmost bar in the chart.	chart type = bar_stacked
	lower_rightmost_object	Return the bottom-rightmost bar in the chart.	chart type = bar_stacked
	lower_leftmost_object	Return the bottom-leftmost bar in the chart.	chart type = bar_stacked
	upper_line_of_objects	Return the objects in the upper-position line of the chart.	chart type = line
	lower_line_of_objects	Return the objects in the bottom-position line of the chart.	chart type = line
min_max_diff	maximum_difference_between_two_group_of_data	Return the maximum difference between the two legends of data: max(|A1-A2|, |B1-B2|, |C1-C2|).	groun number >1
	minimum_difference_between_two_group_of_data	Return the minimum difference between the two legends of data: min(|A1-A2|, |B1-B2|, |C1-C2|).	groun number >1

Table 23: Overview of object functions.

L Examples

L.1 Example 1

Example




```
Question: What is the satisfaction score for Apparel & Accessories (Product B)
     among customers in the age group 18-24 according to the chart?

Function chain: one_object_selection/value_of_objects

Answer: 87
```

InternVL-2.5-8B: Rationale: The chart shows the satisfaction scores for different age groups and product categories. For Apparel & Accessories (Product B), the satisfaction score for the age group 18-24 is indicated by the line that starts from the left side of the chart and rises to the right. The score for this age group is approximately 87 out of 100. Answer: 87

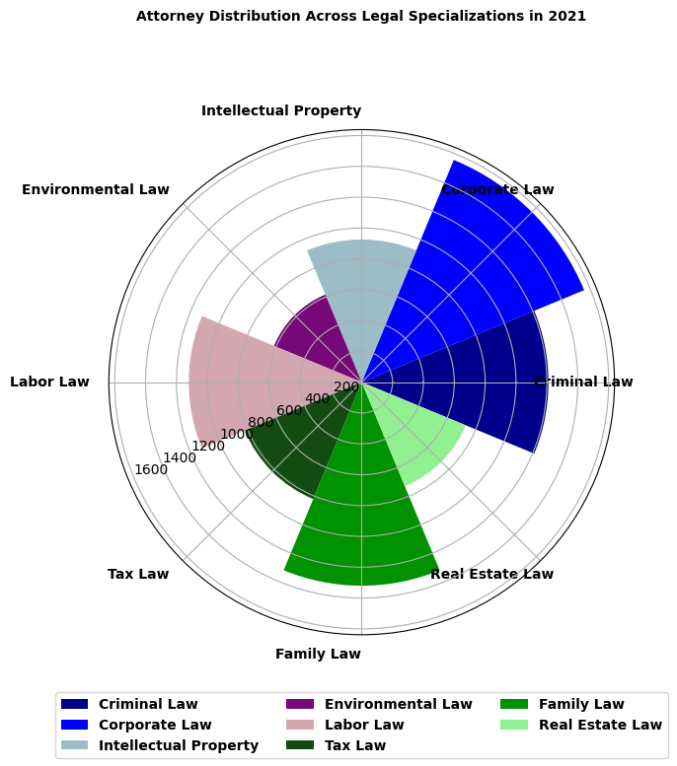
Qwen2VL-7B: Rationale: The satisfaction score for Apparel & Accessories (Product B) among customers in the age group 18-24 is found by looking at the orange line and the corresponding point on the x-axis for the 18-24 age group. The score is approximately 85.

GPT4o: Rationale: To find the satisfaction score for Apparel & Accessories (Product B) among customers in the age group 18-24, we need to look at the orange line on the chart, which represents Product B. At the age group 18-24, the orange line is at the satisfaction score of 85. Answer: 85

Gemini-1.5-Flash: Rationale: The chart shows the satisfaction score for Apparel & Accessories (Product B) among customers in different age groups. Locate the 18-24 age group on the x-axis and follow the orange line representing Apparel & Accessories (Product B) up to the y-axis which represents the satisfaction score. Answer: 86

L.2 Example 2

Example




```
Question: How many legal specializations have more than 925 attorneys
    according to the 'Attorney Distribution' group in the chart?

Function chain: group_selection/objects_that_larger_than_value/
    count_of_objects
Answer: 4
```

InternVL-2.5-8B: Rationale: The chart shows the number of attorneys in each legal specialization. The specializations with more than 925 attorneys are Criminal Law, Corporate Law, and Intellectual Property. These three specializations have bars that extend beyond the 925 mark on the vertical axis. Answer: 3.

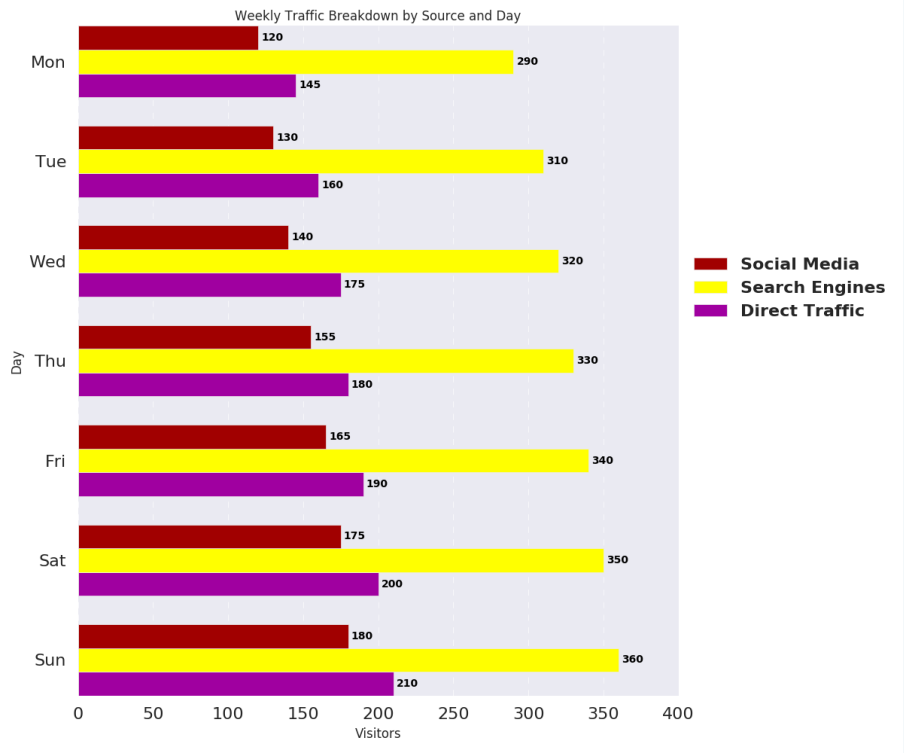
Qwen2VL-7B: Rationale: The 'Attorney Distribution' group in the chart shows that Corporate Law has more than 925 attorneys.

GPT4o: Rationale: To determine how many legal specializations have more than 925 attorneys, we need to examine the chart and identify the specializations that extend beyond the 925 mark on the radial axis. The specializations and their approximate attorney counts are as follows:
1. Criminal Law: Approximately 1400 attorneys
2. Corporate Law: Approximately 1600 attorneys
3. Environmental Law: Approximately 1000 attorneys
4. Family Law: Approximately 1200 attorneys
5. Intellectual Property: Approximately 800 attorneys
6. Labor Law: Approximately 1100 attorneys
7. Real Estate Law: Approximately 1300 attorneys
8. Tax Law: Approximately 900 attorneys
The specializations with more than 925 attorneys are Criminal Law, Corporate Law, Environmental Law, Family Law, Labor Law, and Real Estate Law. Therefore, there are 6 specializations with more than 925 attorneys. Answer: 6

Gemini-1.5-Flash: Rationale: The chart shows that Corporate Law has more than 925 attorneys, Criminal Law has more than 925 attorneys, and Family Law has more than 925 attorneys. Therefore, there are 3 legal specializations with more than 925 attorneys. Answer: Three

L.3 Example 3

Example




```
Question: What is the ratio of Social Media traffic on Sunday compared to the
    day with the second highest Social Media traffic?

Function chain: one_object_selection/value_of_objects/legend_selection/
    second_max_object/value_of_objects/A_divided_by_B

Answer: 1.0285714285714285
```

InternVL-2.5-8B: Rationale: The bar chart shows the weekly traffic breakdown by source and day. The day with the second highest Social Media traffic is Friday with 165 visitors. The ratio of Social Media traffic on Sunday compared to Friday is 180/165, which simplifies to 12/11 or approximately 1.09. Answer: 1.09.

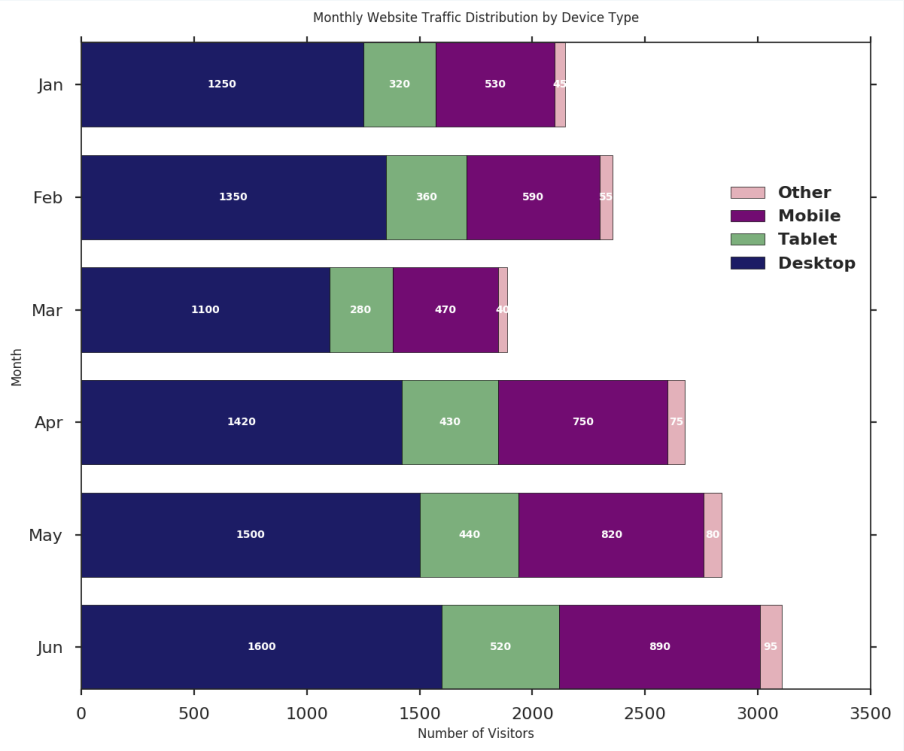
Qwen2VL-7B: Rationale: To find the ratio of Social Media traffic on Sunday compared to the day with the second highest Social Media traffic, we need to identify the Social Media traffic for both days and then calculate the ratio. The Social Media traffic on Sunday is 180 visitors, and the day with the second highest Social Media traffic is Friday, with 165 visitors. The ratio is calculated as follows: 180 (Sunday) / 165 (Friday) = 1.103. Therefore, the ratio of Social Media traffic on Sunday compared to the day with the second highest Social Media traffic is approximately 1.103.

GPT4o: Rationale: From the chart, the Social Media traffic for each day is as follows: Monday - 120, Tuesday - 130, Wednesday - 140, Thursday - 155, Friday - 165, Saturday - 175, Sunday - 210. Sunday has the highest Social Media traffic with 210 visitors. The day with the second highest Social Media traffic is Saturday with 175 visitors. The ratio of Social Media traffic on Sunday compared to Saturday is 210:175. Simplifying this ratio by dividing both numbers by their greatest common divisor, which is 35, we get 6:5. Answer: 6:5

Gemini-1.5-Flash: Rationale:Social media traffic on Sunday is 180. The second highest social media traffic is on Friday (165). The ratio is 180/165 which simplifies to 12/11. Answer:12/11

L.4 Example 4

Example




```
Question: If we find the number of visitors using 'Other' devices in April,
    the number of 'Mobile' visitors in March, and the number of 'Tablet'
    visitors in May, what is the median value among these three numbers?

Function chain: one_object_selection/value_of_objects/one_object_selection/
    value_of_objects/one_object_selection/value_of_objects/median_of_values

Answer: 440
```

InternVL-2.5-8B: Rationale: The number of visitors using 'Other' devices in April is 75, the number of 'Mobile' visitors in March is 470, and the number of 'Tablet' visitors in May is 440. To find the median, we arrange these numbers in ascending order: 75, 440, 470. The median is the middle number, which is 440. Answer: 440

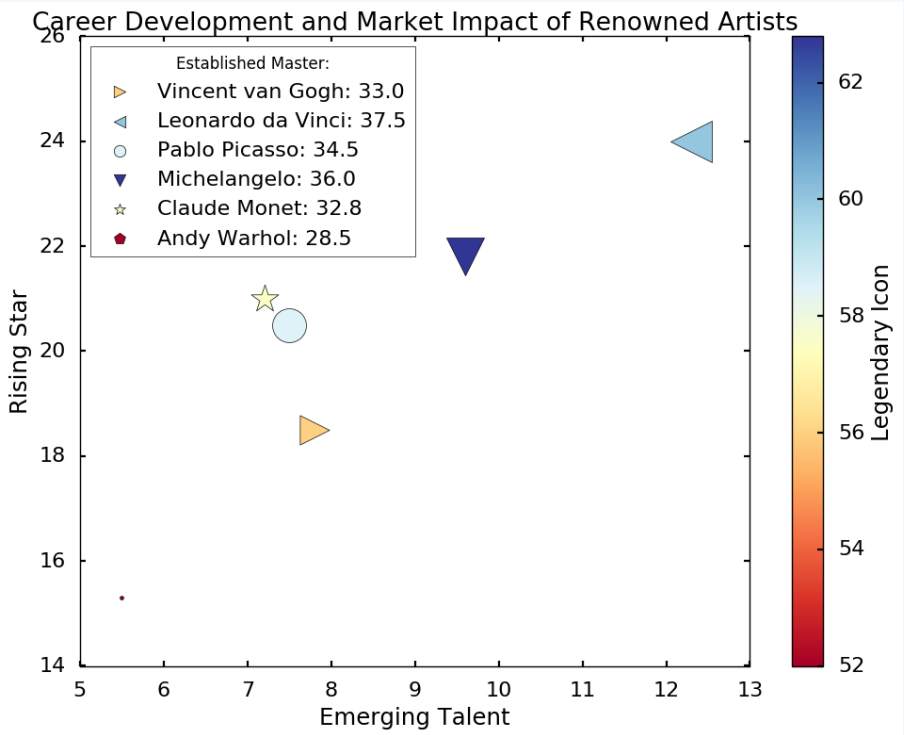
Qwen2VL-7B: Rationale: To find the median value among the three numbers, we first need to arrange the numbers in ascending order. The numbers are: 75 (April), 80 (May), and 40 (March). After arranging them, the middle number is the median. In this case, the median is 75. Answer: 75.

GPT4o: Rationale: To find the median value among the three specified numbers, we first need to identify each number from the chart. The number of 'Other' device visitors in April is 75. The number of 'Mobile' visitors in March is 470. The number of 'Tablet' visitors in May is 440. Now, we have the three numbers: 75, 470, and 440. To find the median, we arrange these numbers in ascending order: 75, 440, 470. The median is the middle number in this ordered list, which is 440. Answer: 440

Gemini-1.5-Flash: Rationale: The number of visitors using 'Other' devices in April is 75. The number of 'Mobile' visitors in March is 470. The number of 'Tablet' visitors in May is 440. The three numbers are 75, 470, and 440. Arranging them in ascending order, we get 75, 440, 470. The median is the middle value, which is 440. Answer:440

L.5 Example 5

Example




```
Question: What is the minimum market value (in millions) shown for any artist
    during their respective phase in the chart, considering the career
    development and market impact of renowned artists categorized into
    Emerging Talent, Rising Star, Established Master, and Legendary Icon?

Function chain: all_object_selection/min_one_object/value_of_objects

Answer: 5.5
```

InternVL-2.5-8B: Rationale: The chart shows different artists at various stages of their career development and market impact. The minimum value is represented by the artist with the lowest market value in the Ëmerging Talentp̈hase, which is Andy Warhol with a value of 28.5 million. Answer: 28.5

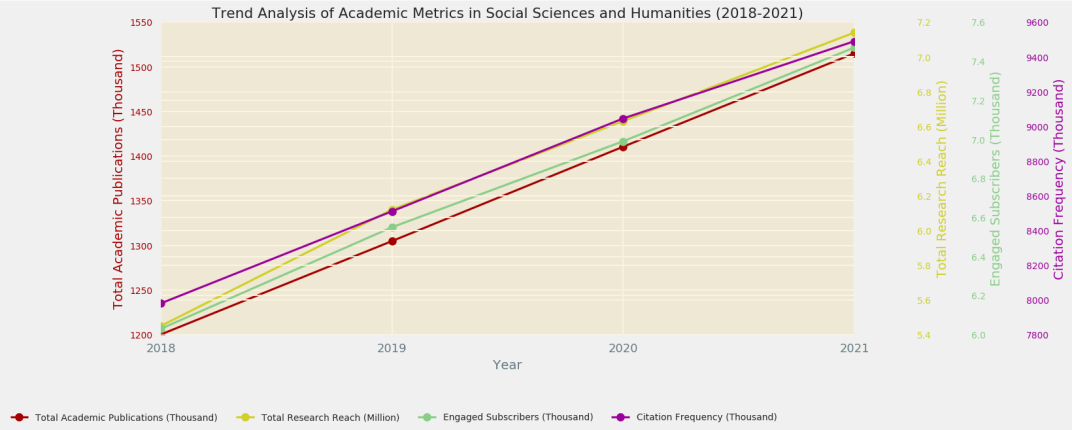
Qwen2VL-7B: Rationale: The minimum market value shown for any artist during their respective phase is 5.5 million, which is represented by the smallest data point in the chart.

GPT4o: Rationale: The chart shows the market values of various renowned artists during their respective phases. The artists listed are Vincent van Gogh, Leonardo da Vinci, Pablo Picasso, Michelangelo, Claude Monet, and Andy Warhol. The market values are given in millions: Vincent van Gogh (33.0), Leonardo da Vinci (37.5), Pablo Picasso (34.5), Michelangelo (36.0), Claude Monet (32.8), and Andy Warhol (28.5). Among these values, Andy Warhol has the lowest market value at 28.5 million. Answer: 28.5

Gemini-1.5-Flash: Rationale: The chart shows the market value (in millions) for several artists categorized by their career phase. The minimum market value among all artists in all phases is 28.5 million for Andy Warhol. Answer: 28.5

L.6 Example 6

Example




```
Question: What was the total number of academic publications (in thousands) in
     the year 2018 according to the chart?

Function chain: one_object_selection/value_of_objects

Answer: 1200.2
```

InternVL-2.5-8B: Rationale: The chart shows the total number of academic publications in thousands for the years 2018 to 2021. In 2018, the line representing total academic publications starts at the point corresponding to 1200 thousand. This indicates that the total number of academic publications in the year 2018 was 1200 thousand. Answer: 1200.

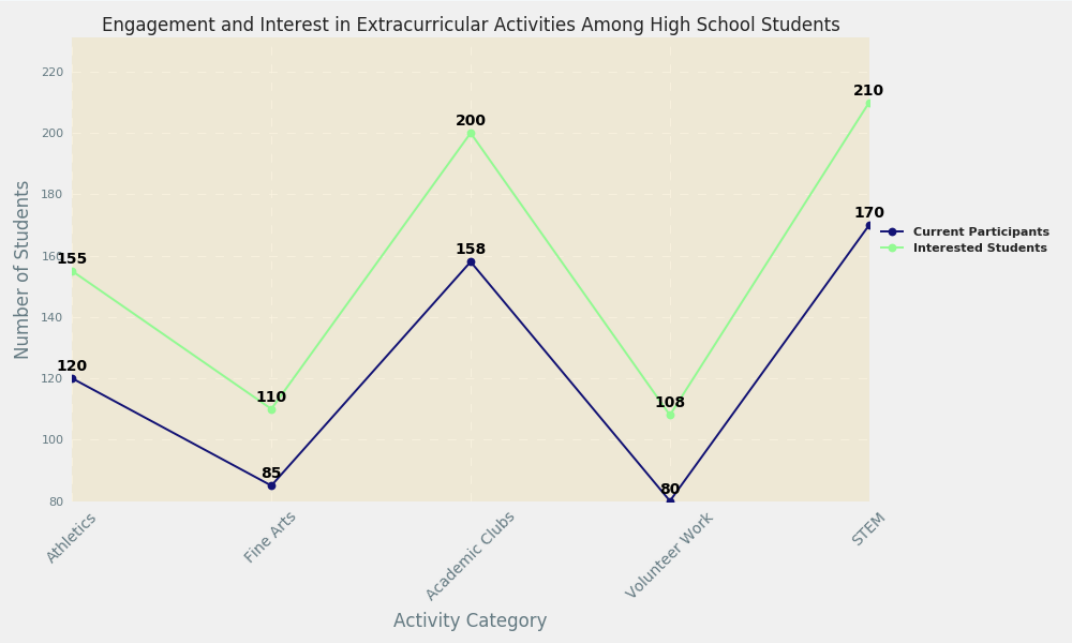
Qwen2VL-7B: Rationale: The total number of academic publications in 2018 is represented by the red line on the chart. The value is approximately 1200 thousand.

GPT4o: Rationale: The chart shows the trend of various academic metrics from 2018 to 2021. The red line represents the total academic publications in thousands. In 2018, the red line is at the 1250 mark on the left y-axis, which indicates the total number of academic publications in thousands for that year. Answer: 1250

Gemini-1.5-Flash: Rationale: In 2018, the red line representing total academic publications (in thousands) intersects the y-axis at approximately 1200. Answer: 1200.

L.7 Example 7

Example




```
Question: If we find the number of current participants in Volunteer Work,
    identify the maximum number of interested students across all activity
    categories, and also find the minimum number of current participants
    across all activity categories, what is the sum of these three values?

Function chain: one_object_selection/value_of_objects/legend_selection/
    max_one_object/value_of_objects/legend_selection/min_one_object/
    value_of_objects/sum_of_values

Answer: 370
```

InternVL-2.5-8B: Rationale: The number of current participants in Volunteer Work is 80. The maximum number of interested students across all activity categories is 210 (in STEM). The minimum number of current participants across all activity categories is 80 (in Volunteer Work). The sum of these three values is 80 + 210 + 80 = 370. Answer: 370

Qwen2VL-7B: Rationale: To find the sum of the three values, we need to identify the number of current participants in Volunteer Work, the maximum number of interested students across all activity categories, and the minimum number of current participants across all activity categories. Then, we add these three values together. Answer: 468

GPT4o: Rationale: The number of current participants in Volunteer Work is 80. The maximum number of interested students across all activity categories is 210 (in STEM). The minimum number of current participants across all activity categories is 80 (in Volunteer Work). The sum of these three values is 80 + 210 + 80 = 370. Answer: 370

Gemini-1.5-Flash: Rationale: The number of current participants in Volunteer Work is 80. The maximum number of interested students across all activity categories is 210. The minimum number of current participants across all activity categories is 80. The sum of these three values is 80 + 210 + 80 = 370. Answer: 370

L.8 Example 8

Example


```
Question: If we get the highest 'Involuntary Exits' rate among all departments
    and add it to the 'Involuntary Exits' rate of 'Sales', and then add the '
    Voluntary Exits' rate of 'Operations', what is the total rate?

Function chain: legend_selection/max_one_object/value_of_objects/
    one_object_selection/value_of_objects/color_group_selection/
    value_of_objects/sum_of_values

Answer: 24.0%
```

InternVL-2.5-8B: Rationale: The highest 'Involuntary Exits' rate among all departments is 8.3% (Marketing). Adding this to the 'Involuntary Exits' rate of 'Sales' (6.5%) gives us 14.8%. The 'Voluntary Exits' rate of 'Operations' is 9.2%. Adding this to the previous sum (14.8%) gives us a total rate of 24%. Answer: 24%

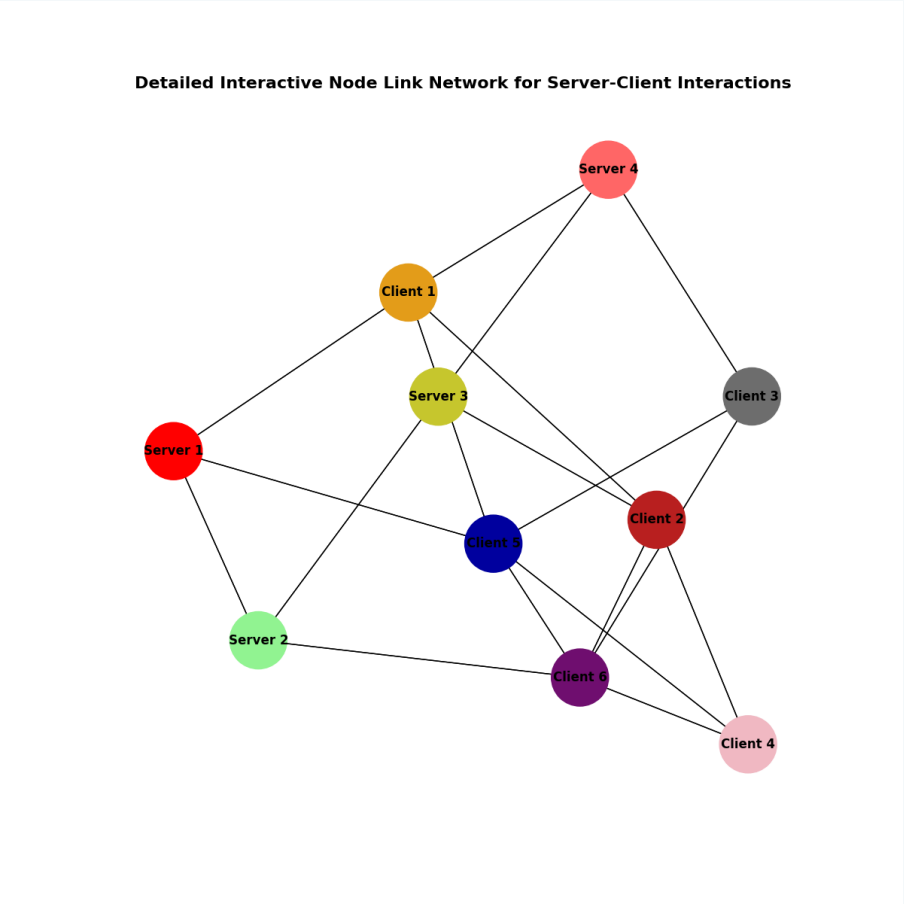
Qwen2VL-7B: Rationale: To find the total rate, we need to add the highest 'Involuntary Exits' rate, which is 6.5 from 'Sales', to the 'Involuntary Exits' rate of 'Sales', which is 6.5, and then add the 'Voluntary Exits' rate of 'Operations', which is 9.2. So, the total rate is 6.5 (from 'Sales') + 6.5 (from 'Sales') + 9.2 (from 'Operations') = 22.2.

GPT4o: Rationale: First, identify the highest 'Involuntary Exits' rate among all departments. From the chart, the 'Involuntary Exits' rates are: Sales (6.5), Marketing (8.3), Engineering (4.7), Finance (5.5), HR (3.3), and Operations (2.2). The highest rate is 8.3 from Marketing. Next, add this to the 'Involuntary Exits' rate of 'Sales', which is 6.5. So, 8.3 + 6.5 = 14.8. Finally, add the 'Voluntary Exits' rate of 'Operations', which is 9.2. Therefore, 14.8 + 9.2 = 24.0. Answer: 24.0

Gemini-1.5-Flash: Rationale: The highest involuntary exit rate is 8.3 (Marketing). Adding this to the involuntary exit rate of Sales (6.5) gives 14.8. Adding the voluntary exit rate of Operations (2.2) to this gives a total of 17. Answer: 17

L.9 Example 9

Example




```
Question: In the Detailed Interactive Node Link Network for Server-Client
    Interactions, is the node represented by 'Client 5' connected to the node
    represented by 'Client 3'?

Function chain: one_object_selection/if_object_connect_to_A

Answer: Yes
```

InternVL-2.5-8B: Rationale: The diagram shows a network of connections between servers and clients. 'Client 5' is connected to 'Server 3' and 'Server 2', while 'Client 3' is connected to 'Server 4'. There is no direct connection between 'Client 5' and 'Client 3' in the diagram. Answer: no

Qwen2VL-7B: Rationale: The node represented by 'Client 5' is connected to the node represented by 'Client 3' as indicated by the line connecting them in the network diagram. Answer: Yes

GPT4o: Rationale: To determine if 'Client 5' is connected to 'Client 3', we need to trace the lines (edges) from 'Client 5' to see if there is a direct connection to 'Client 3'. Upon examining the network, 'Client 5' is connected to 'Server 3', 'Client 2', and 'Client 6'. 'Client 3' is connected to 'Server 4', 'Server 3', and 'Client 2'. There is no direct line connecting 'Client 5' to 'Client 3'. Therefore, 'Client 5' is not directly connected to 'Client 3'. Answer: No

Gemini-1.5-Flash: Rationale: Client 5 is connected to Client 3 via a direct link in the network graph. There is a line connecting the two nodes. Answer: Yes