On Pre-training of Multimodal Language Models Customized for Chart Understanding

Wan-Cyuan Fan[1,3] **Yen-Chun Chen**[2] **Mengchen Liu**[2] **Lu Yuan**[2] **Leonid Sigal**[1,3,4]

[1]UBC [2]Microsoft [3]Vector Institute for AI [4]CIFAR AI Chair

{wancyuan, lsigal}@cs.ubc.ca

Abstract

Recent studies customizing Multimodal Large Language Models (MLLMs) for domain-specific tasks have yielded promising results, especially in the field of scientific chart comprehension. These studies generally utilize visual instruction tuning with specialized datasets to enhance question and answer (QA) accuracy within the chart domain. However, they often neglect the fundamental discrepancy between natural image-caption pre-training data and digital chart image-QA data, particularly in the models' capacity to extract underlying numeric values from charts. This paper tackles this oversight by exploring the training processes necessary to improve MLLMs' comprehension of charts. We present three key findings: (1) Incorporating raw data values in alignment pre-training markedly improves comprehension of chart data. (2) Replacing images with their textual representation randomly, during end-to-end fine-tuning, transfers the language reasoning to chart interpretation skills. (3) Requiring the model to first extract the underlying chart data and then answer the question in the fine-tuning can further improve the accuracy. Consequently, we introduce CHOPINLLM, an MLLM tailored for in-depth chart comprehension. CHOPINLLM effectively interprets various types of charts, including unannotated ones, while maintaining robust reasoning abilities. Furthermore, we establish a new benchmark to evaluate MLLMs' understanding of different chart types across various comprehension levels. Experimental results show that CHOPINLLM exhibits strong performance in understanding both annotated and unannotated charts across a wide range of types. The latest version of the paper will be updated here.

1 Introduction

In today's data-driven world, visualizations like bar and pie charts are crucial for deciphering complex datasets. However, the increasing diversity and complexity of these charts highlights the need for advanced tools to enhance human capabilities in data analysis. Artificial Intelligence (AI), particularly Multimodal Large Language Models (MLLMs) [1, 2, 12, 13, 17, 18, 21–23, 27–29], is increasingly used to automate the understanding of scientific charts, promising more efficient and accurate analysis. Robust benchmarks are also essential, setting standards and metrics that drive the development and evaluation of these AI tools.

Prior studies have introduced end-to-end neural models aimed at enhancing chart comprehension [7, 9, 30], such as masked table prediction [8, 30], chart question answering [15], and chart de-rendering [9]. These models specialize in handling one task each within the domain of chart analysis. Furthermore, advancements in Multimodal Large Language Models (MLLMs), exemplified by LLaVA [11, 12] and miniGPT [31], have showcased their versatility in vision-language tasks. These generalist

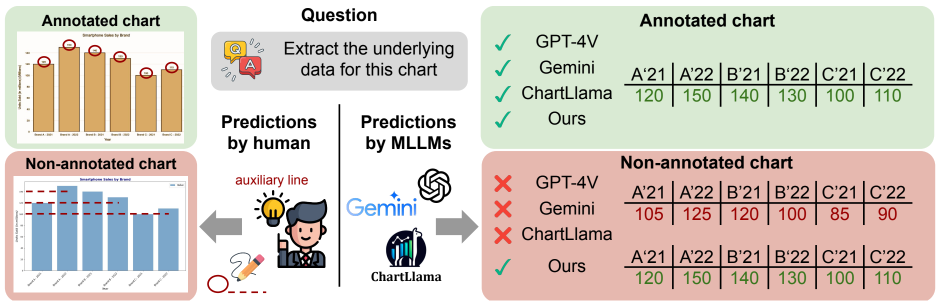


Figure 1: The underlying data values can be inferred regardless of whether the chart is annotated. However, existing MLLMs rely on annotations and struggle with unannotated charts. In contrast, our model bridges this fundamental discrepancy between natural image-caption pre-training data and digital chart image-QA data, enabling it to extract values regardless of whether the chart is annotated.

models undergo a two-stage training process: initially learning visual-language alignment through image-caption pairs, followed by end-to-end fine-tuning using image-QA pairs. This training not only enables LLMs to interpret visual data but also retains their extensive pre-trained knowledge, which supports their reasoning abilities and leads to strong performance across diverse visual language understanding tasks.

Recent advancements have further ignited interest in tailoring MLLMs to specialized domains such as scientific chart understanding. Han et al. [4], Liu et al. [10] have explored collecting instruction-tuned chart data and low-rank adaptation [5] to enhance MLLMs' proficiency with unique chart characteristics. However, research on the fundamental-training regimes – namely, pre-training to align across modalities and comprehensive end-to-end fine-tuning – for chart-specific understanding remains scarce. As shown in Fig. 1, existing MLLMs often struggle to extract the underlying data from charts when numerical values are not annotated. We hypothesize that this issue stems from a gap in vision-language alignment between natural image-caption pairs and digital chart-data pairs. Without targeted pre-training for chart-data alignment, models may resort to relying on a "shortcut" of recognizing numeric annotations through OCR during fine-tuning with QA pairs, rather than truly understanding the visual subtleties of diverse charts.

This paper addresses the above issues by concentrating on the essential training methodologies for MLLMs, including cross-modal feature alignment pre-training and comprehensive end-to-end fine-tuning. Our research is guided by the question, *"How does fundamental MLLM training influence the enhancement of general MLLMs with chart-specific domain understanding?"* Our findings indicate that: (1) Raw data extraction are pivotal in alignment pre-training to bolster chart data comprehension; (2) Substituting some chart images with purely textual data during end-to-end fine-tuning not only preserves LLM's text-only reasoning ability but also augments chart interpretation capabilities; (3) Augmenting QAs with data extraction tasks in the fine-tuning phase allows model to achieve the data prompting during testing, where it first extract data and then answer the QAs, further improving the its reasoning skills.

In summary, our key contributions in this paper are: (1) We introduce ChopinLLM,[1] a Multimodal Large Language Model tailored for comprehensive chart understanding. This model excels at interpreting various chart types including unannotated ones, underpinned by our detailed analysis and training guidance that emphasizes the importance of foundational training for chart-specific tasks. (2) We propose a novel data generation pipeline using text-only Large Language Models to efficiently produce large-scale pairwise data. This approach significantly reduces the costs and complexity of data generation for MLLM training. Furthermore, (3) we establish a robust benchmark comprising a diverse array of chart types and question-answering levels, designed to rigorously evaluate MLLMs' fundamental understanding of the scientific chart domain.

2 Generating data for chart understanding

To build a chart understanding MLLM and study its fundamental training process, a comprehensive dataset containing chart images paired with captions and raw data is essential for pre-training, alongside different types of question-answer pairs for end-to-end fine-tuning. However, no existing

[1]**Ch**art **O**riented **P**retraining **In**tegration in **L**arge **L**anguage **M**odels

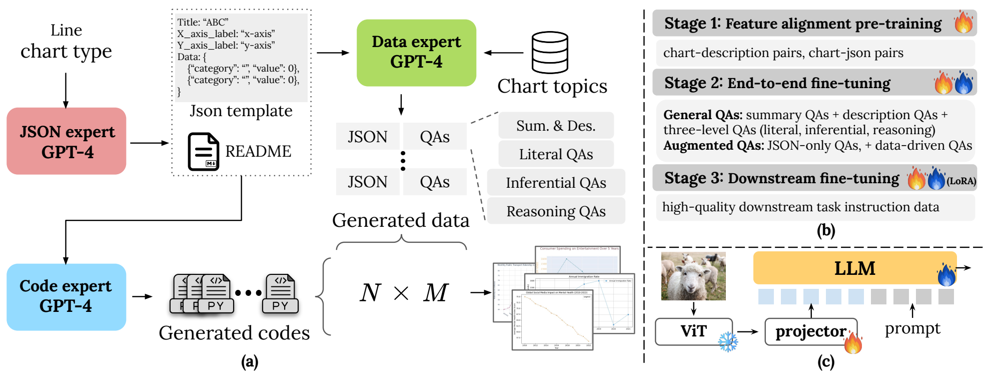

Line chart type
JSON expert GPT-4
Title: "ABC"
X_axis_label: "x-axis"
Y_axis_label: "y-axis"
Data: {
{"category": "", "value": 0},
{"category": "", "value": 0},
}
Json template
README
Data expert GPT-4
Chart topics
JSON
QAs
JSON
QAs
Sum. & Des.
Literal QAs
Inferential QAs
Reasoning QAs
Generated data
Code expert GPT-4
PY
PY
Generated codes
$N \times M$
(a)
Stage 1: Feature alignment pre-training
chart-description pairs, chart-json pairs
Stage 2: End-to-end fine-tuning
General QAs: summary QAs + description QAs + three-level QAs (literal, inferential, reasoning)
Augmented QAs: JSON-only QAs, + data-driven QAs
Stage 3: Downstream fine-tuning
(LoRA)
high-quality downstream task instruction data
(b)
LLM
ViT
projector
prompt
(c)

Figure 2: **Overview of (a) the proposed data generation pipeline and (b) training strategies of CHOPINLLM.** Generating code and data points conforming to a shared JSON template enables quadratic scaling of the data size (w.r.t. to #GPT calls). The 3-stage training equips our model to grasp the underlying data, thereby achieving a fundamental understanding of charts. (N and M denote the number of generated scripts and data, respectively.)

dataset provides the necessary variety of chart types, topics, and styles. To bridge this gap, we introduce a novel data generation pipeline for large-scale chart data generation (Sec. 2.1) and QAs generation (Sec. 2.2). With the data at hand, we then explore various training strategies in the later sections, including feature alignment pre-training and end-to-end fine-tuning for LLMs. Figure 2 presents an overview of our framework.

2.1 Efficient data generation with quadratic scaling

Our data generation leverages the promising text content generation and coding abilities of large language models, *e.g.*, GPT-4, to generate chart images and data. Specifically, LLMs allow us to synthesize raw data for chart images, and then the generated Python script turns the raw data into a chart image. In this way, we can produce image data without accessing costly multimodal LLMs like GPT-4V. Unlike previous and concurrent works [4, 25] that prompt LLMs to iteratively generate CSV data, QAs, and Python script for each chart image – a process that is costly to massively scale – our pipeline features parallel code and data generation through shared templates and READMEs for consistent definitions and formats across the same chart types. Most importantly, since all code script and data share the same structure, our generated data can be universally applied to any generated code and vice versa, significantly enhancing scalability without exhaustively prompting LLMs. Please refer to Appendix A for more details about shared templates and the raw data generation.

2.2 Diverse QA synthesis

Based on the parallel data generation pipeline, we are able to collect massive amount of chart image and JSON raw data pairs for the feature alignment pre-training. Now, we details how we generate different types of QAs for end-to-end fine-tuning. Specifically, having each JSON data as input, we use text-only LLM to generate question-answer (QA) pairs. To cover various question-anwser for chart data, we include general QAs, containing not only description and summary QA but also three different level of QAs: literal QAs, inferential QAs, and reasoning QAs (as illustrated in Fig. A3). Furthermore, to enhance the training of chart understanding, we introduce two additional augmented QAs (for training only): text-only QAs and data-driven QAs. Please refer to Appendix A for the details of each QA type.

2.3 A new benchmark for comprehensive chart understanding

A chart expert model should be capable of understanding a wide range of common chart types and, like a human, should not only be able to answer questions of varying complexity but also grasp the underlying data. However, as shown in Table 1, existing chart benchmarks either cover only a limited range of chart types (*e.g.*, line, bar, and pie charts) or lack comprehensive QA sets to evaluate a model's understanding of charts from various perspectives, including raw data comprehension, inferential abilities, and mathematical reasoning capabilities. To bridge this gap, we propose a

Table 1: Comparative analysis with existing benchmarks for chart understanding evaluations. * denotes unbounded chart types. Chart variation refers to whether the dataset contains chart images with different styles but sharing the same raw data.

Benchmark	# Image	# Chart type	Avg. # QAs per image	Multi-level QAs per image	Raw data per image	Chart Variation
PlotQA [20]	33.7k	3	1	✗	✗	✗
ChartQA [14]	1.5k	3	1	✗	✓	✗
Chart-to-text [6]	6.6k	6	1	✗	✗	✗
MMC [10]	2k	6	1	✗	✓	✗
Chartbench [26]	2.1k	9	9	✓	✗	✗
ChartX [25]	6k	18	1	✗	✓	✗
CharXiv [24]	2.3k	*	5	✓	✗	✗
Ours	5.48k	20	13.5	✓	✓	✓

comprehensive benchmark derived from the aforementioned synthetic dataset. It covers 20 different chart types, three different levels of QAs (literal, inferential, and reasoning QAs), and provides both long and short answers. Notably, the chart images in the benchmark are not all annotated, allowing assessment of the model's ability to understand the underlying data of a chart as humans do. To ensure the quality of the images in the benchmark, we employed human evaluations to filter the data and obtain a high-quality test set. The evaluations are based on two criteria: *Answerability*: whether the question is answerable given the chart image. *Correctness*: whether the provided answer is correct. Please refer to Appendix B in the supplementary materials for more details about benchmark statistics, filtering, analysis, etc. Note that these QAs equally cover literal, inferential, and reasoning questions for measuring chart understanding of MLLMs.

3 Experiments and model analysis

3.1 Experimental setup

Benchmark. Our evaluation utilizes four classical benchmarks: (1) ChartQA dataset [14], which includes 1.5k chart images in its test set, divided into human-written and machine-generated questions with 1.2k QA pairs each. The human-written questions often require mathematical reasoning. (2) ChartQA also provides CSV data for each image, enabling us to conduct a Chart-to-Table task to assess the ability of MLLMs to extract raw data from charts, following previous studies [4, 8]. (3) Additionally, we use the PlotQA dataset [20] where images generally lack numerical value annotations, necessitating value inference relative to the Y-axis. (4) Lastly, for evaluating the models' capability to capture global concepts, we assess on the Chart-to-Text task using the *Pew* and *Statista* splits from the dataset [6]. For the evaluation metrics, we follow previous works [4, 6, 14] and use relaxed accuracy for QA tasks, as well as the *F1* score and *Relative Number Set Similarity* (RNSS) for chart-to-table tasks. Additionally, *BLEU-4* is used for the chart-to-text task.

The 3-stage Training Process. Unlike previous approaches that convert a general MLLM into a chart-specific expert by only applying LoRA fine-tuning on limited high-quality data [4], training ChopinLLM unfolds in three stages, illustrated in Fig. 2 (b). The 3-stage training enables our model not only to understand chart QAs and downstream tasks but also to capture the underlying data, thereby achieving a fundamental understanding of charts. In the initial pre-training stage, we fix the ViT and LLM while training the projector from scratch using original LLaVA data alongside our newly generated chart-description and chart-json pairs. The second stage involves freezing ViT and jointly fine-tuning the projector and LLM with both original LLaVA QA pairs and our generated chart QA pairs, enabling the LLM to comprehend visual tokens and facilitate chart question answering. Finally, we apply LoRA fine-tuning to align the LLM's response distribution with the target downstream dataset. In this section, we showcase the model's performance at the final stage on the downstream tasks. For the results of stages 1 and 2, please refer to Appendix C.1 and Appendix C.2.

3.2 Downstream fine-tuning

We build ChopinLLM with the best setting based on the observation in the stage 1 and 2 (the data used in each stage can be found in Fig. 2 (b)), and we compare ChopinLLM with existing chart understanding approaches, including Pix2struct [7], Matcha [9], Unichart [15], Deplot [8], LLaVA [12], and ChartLlama [4]. The results are shown in Table 2.

Table 2: **Comprehensive evaluation across four chart benchmarks.** H and A denote the human and augmented branch in ChartQA, respectively. Stat. represent the statista split. †: our reproduction using the official code. Note that for fair comparison, we don't use chain-of-reasoning in the inference. The best result is highlighted in **Bold** and the second underlined. # chart data denotes the number of pairwise chart data used in the training. A and S in the data source represent annotated data and synthetic data, respectively.

Method	Data #	Data Source	ChartQA			Chart-to-Table		Chart-to-Text		PlotQA*	
			H	A	Avg.	F1	RNSS	Pew	Stat.	v1	v2
Pix2struct [7]	80M	A	30.50	81.60	56.00	-	-	10.30	38.00	-	-
Matcha [9]	16M	S+A	38.20	90.20	64.20	-	-	12.20	39.40	-	-
Unichart [15]	7M	S+A	43.92	88.56	66.24	52.71	-	12.48	38.21	-	-
DePlot [8]	0.5M	S+A	-	-	-	87.22	94.28	-	-	-	-
LLaVA$_{7B}$† [12]	-	-	36.00	67.44	51.72	56.96	91.83	8.50	21.50	27.26	30.64
LLaVA$_{13B}$	-	-	37.68	72.96	55.32	48.95	-	7.16	24.65	-	-
LLaVA$_{13B}$†	-	-	42.56	73.60	58.08	63.18	93.18	8.83	22.39	27.68	30.98
ChartLlama$_{13B}$ [4]	0.16M	A	48.96	90.36	69.66	89.84	94.65	14.23	40.71	29.76	29.93
MMC$_{7B}$ [10]	0.6M	S+A	-	-	57.40	-	-	-	-	-	-
ChartInstruct$_{7B}$ [16]	0.19M	A	45.52	87.76	66.64	18.87	34.59	13.83	**43.53**	-	-
ChartAst$_{13B}$ [19]	24M	S+A	**65.9**	**93.9**	**79.9**	**91.6**	-	**15.5**	41.0	-	-
CHOPINLLM $_{7B}$	5M	S	52.28	87.68	69.98	83.63	95.27	11.50	38.97	30.06	31.08
CHOPINLLM $_{13B}$	5M	S	54.11	88.67	71.39	88.12	**95.95**	12.66	40.81	**33.98**	**33.96**

Classical question-answering on ChartQA. We find that CHOPINLLM achieves the second best performance on ChartQA, as shown in Table 2. Notably, compared to the recent work of ChartAst, we use significantly less data, and most importantly, our training data is fully synthetic, requiring no additional human effort. In comparison to the third-best model, ChartLlama, we outperform it by $\approx 5\%$ on the human split of ChartQA. Note that the human split in ChartQA is more challenging than the augmented split, as it contains more reasoning questions, suggesting that CHOPINLLM is better at performing reasoning tasks.

Raw data and global concept understanding. As shown in Table 2, CHOPINLLM achieves the competitive F1 score and the highest RNSS result, indicating that CHOPINLLM can capture not only the structure but numerical values of raw data of chart images. We note that the performance on the chart-to-table task may have been saturated, as the images are mostly annotated. In this context, this primarily measures the OCR capability and does not assess the ability to capture the underlying data. As for the Chart-to-Text, shown in Table 2, CHOPINLLM performs comparable in the global concept capturing and can caption chart image with meaningful texts.

Performance on unannotated chart images. Most of the images in ChartQA [14] are annotated, which means the numerical values of data points are explicitly shown on the images. We observe that existing chart MLLMs, such as ChartLlama [4], seem to heavily rely on this annotation for chart understanding, which is not ideal since real-world charts may be unannotated. We further evaluate them using the PlotQA dataset, and the results are shown in the last column of Table 2. Notably, since training previous models like ChartLlama on PlotQA is infeasible, we load the model weights as used in ChartQA and perform zero-shot prediction on PlotQA. The results show that our model performs significantly better ($\approx 3\%$ improvement) on unannotated chart images, suggesting that our methods with fundamental training rely less on numerical annotations. Note that the comparison with ChartAst and ChartInstruct is not included, as it was trained on PlotQA, which would affect the validity of the zero-shot predictions on PlotQA.

4 Conclusion

In this paper, we explore the impact of fundamental training strategies in adapting generalist Multimodal Large Language Models (MLLMs) to chart understanding. We offer practical guidance for optimizing feature alignment pre-training and end-to-end fine-tuning. Leveraging these enhanced training strategies, we introduce a specialized chart MLLM, named CHOPINLLM, capable of interpreting diverse chart types independently of numerical annotations. Extensive experiments confirm that CHOPINLLM surpasses the previous state-of-the-art across four benchmarks, validating our framework's effectiveness. Additionally, we present a new benchmark specifically designed to evaluate MLLMs' comprehension across various chart types and multiple levels of understanding.

Bibliography

[1] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. 1

[2] Guo Chen, Yin-Dong Zheng, Jiahao Wang, Jilan Xu, Yifei Huang, Junting Pan, Yi Wang, Yali Wang, Yu Qiao, Tong Lu, et al. Videollm: Modeling video sequence with large language models. *arXiv preprint arXiv:2305.13292*, 2023. 1

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 6

[4] Yucheng Han, Chi Zhang, Xin Chen, Xu Yang, Zhibin Wang, Gang Yu, Bin Fu, and Hanwang Zhang. Chartllama: A multimodal llm for chart understanding and generation. *arXiv preprint arXiv:2311.16483*, 2023. 2, 3, 4, 5, 6

[5] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 2

[6] Shankar Kantharaj, Rixie Tiffany Ko Leong, Xiang Lin, Ahmed Masry, Megh Thakkar, Enamul Hoque, and Shafiq Joty. Chart-to-text: A large-scale benchmark for chart summarization. *arXiv preprint arXiv:2203.06486*, 2022. 4

[7] Kenton Lee, Mandar Joshi, Iulia Raluca Turc, Hexiang Hu, Fangyu Liu, Julian Martin Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. Pix2struct: Screenshot parsing as pretraining for visual language understanding. In *ICML*, 2023. 1, 4, 5

[8] Fangyu Liu, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhu Chen, Nigel Collier, and Yasemin Altun. Deplot: One-shot visual language reasoning by plot-to-table translation. *arXiv preprint arXiv:2212.10505*, 2022. 1, 4, 5

[9] Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Martin Eisenschlos. Matcha: Enhancing visual language pretraining with math reasoning and chart derendering. *arXiv preprint arXiv:2212.09662*, 2022. 1, 4, 5

[10] Fuxiao Liu, Xiaoyang Wang, Wenlin Yao, Jianshu Chen, Kaiqiang Song, Sangwoo Cho, Yaser Yacoob, and Dong Yu. 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*, 2024. 2, 4, 5

[11] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023. 1

[12] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 1, 4, 5, 6

[13] Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Yaofeng Sun, et al. Deepseek-vl: towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*, 2024. 1

[14] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022. 4, 5, 2

[15] Ahmed Masry, Parsa Kavehzadeh, Xuan Long Do, Enamul Hoque, and Shafiq Joty. Unichart: A universal vision-language pretrained model for chart comprehension and reasoning. *arXiv preprint arXiv:2305.14761*, 2023. 1, 4, 5

[16] Ahmed Masry, Mehrad Shahmohammadi, Md Rizwan Parvez, Enamul Hoque, and Shafiq Joty. Chartinstruct: Instruction tuning for chart comprehension and reasoning. *arXiv preprint arXiv:2403.09028*, 2024. 5, 6

[17] Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Floris Weers, et al. Mm1: Methods, analysis & insights from multimodal llm pre-training. *arXiv preprint arXiv:2403.09611*, 2024. 1

[18] MeetkAI. Functionary, 2024. URL `https://functionary.meetkai.com/`. 1

[19] Fanqing Meng, Wenqi Shao, Quanfeng Lu, Peng Gao, Kaipeng Zhang, Yu Qiao, and Ping Luo. Chartassisstant: A universal chart multimodal language model via chart-to-table pre-training and multitask instruction tuning. In *Findings of the Association for Computational Linguistics ACL*, 2024. 5, 6

[20] Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. Plotqa: Reasoning over scientific plots. In *WACV*, 2020. 4, 2

[21] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023. 1

[22] Shishir G Patil, Tianjun Zhang, Vivian Fang, Roy Huang, Aaron Hao, Martin Casado, Joseph E Gonzalez, Raluca Ada Popa, Ion Stoica, et al. Goex: Perspectives and designs towards a runtime for autonomous llm applications. *arXiv preprint arXiv:2404.06921*, 2024.

[23] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. 1

[24] Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, et al. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *arXiv preprint arXiv:2406.18521*, 2024. 4

[25] Renqiu Xia, Bo Zhang, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Min Dou, Botian Shi, Junchi Yan, et al. Chartx & chartvlm: A versatile benchmark and foundation model for complicated chart reasoning. *arXiv preprint arXiv:2402.12185*, 2024. 3, 4, 2, 5

[26] Zhengzhuo Xu, Sinan Du, Yiyan Qi, Chengjin Xu, Chun Yuan, and Jian Guo. Chartbench: A benchmark for complex visual reasoning in charts. *arXiv preprint arXiv:2312.15915*, 2023. 4

[27] Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S Yu. Large language models for robotics: A survey. *arXiv preprint arXiv:2311.07226*, 2023. 1

[28] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023.

[29] Pan Zhang, Xiaoyi Dong Bin Wang, Yuhang Cao, Chao Xu, Linke Ouyang, Zhiyuan Zhao, Shuangrui Ding, Songyang Zhang, Haodong Duan, Hang Yan, et al. Internlm-xcomposer: A vision-language large model for advanced text-image comprehension and composition. *arXiv preprint arXiv:2309.15112*, 2023. 1

[30] Mingyang Zhou, Yi R Fung, Long Chen, Christopher Thomas, Heng Ji, and Shih-Fu Chang. Enhanced chart understanding in vision and language task via cross-modal pre-training on plot table pairs. *arXiv preprint arXiv:2305.18641*, 2023. 1

[31] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023. 1

Appendix

A Details of the data generation pipeline

A.1 Details of efficient data generation with quadratic scaling

Shared Template and README. As shown in Fig. 2 (a), given a chart type (*e.g.*, line) sampled from a predefined chart type database, the JSON expert GPT-4 first generates a JSON template for the given chart type, along with a README file. In detail, the JSON template contains general information for the chart image, including the title, x-axis, y-axis information, and raw data. The README contains the definition of the chart type and the meanings of the keys and values to enhance understanding of the JSON template. Please refer to Appendix D for some examples. We note that the JSON template, together with the README, ensures the consistency of data generation so that further data and code generation can follow the explicit format and definition guidance of the template data. Note that we choose JSON as our primary data representation format, in contrast to previous works [4, 14, 20, 25], which used CSV. The JSON format allows us to incorporate not only numerical data but also additional chart information, such as titles and the scales of x and y axes, which is beneficial for pair-wise pre-training tasks. Moreover, JSON data is structured, and when paired with a README file, it minimizes ambiguity in data descriptions, which is particularly valuable for complex chart types. For instance, in candlestick charts, we can clearly define a data point as a dictionary containing "open", "close", "high", and "low" values, rather than a list where the meaning of each number might be unclear.

Orthogonal Data and Code Generation. With the template files at hand, we can generate data and code independently. For the data generation branch, to ensure the generated data covers diverse topics, we jointly input the produced template files (*i.e.*, JSON template and README) and a topic sampled from a pre-defined topic set (*e.g.*, energy production and market share) into a data expert GPT-4 module. For the complete topic list, please refer to Appendix E. We require the data expert GPT-4 to follow the definitions in the template files and generate M JSON data along with different kinds of questions and answers (*e.g.*, summary QA) based on the raw data. As for code generation, another code expert GPT-4 is utilized to produce N Python code based on the given chart type, data template, and Python library. Note that to prevent generating simple code repeatedly for the given chart type, we explicitly ask the code expert GPT-4 to introduce visual variations in aspects such as color, legend, grid, font, and mark texture, *etc*. More details can be found in the Appendix.

A.2 Dataset filtering

We provide the details for each QA type as follows:

- **Description QAs:** Generate objective descriptions based on the chart data.
- **Summary QAs:** Summarize the chart, highlighting key findings.
- **Literal QAs:** Extract specific values directly from the data.
- **Inferential QAs:** Infer global insights, such as identifying extreme values.
- **Reasoning QAs:** Perform calculations to derive answers from chart data.
- **JSON-only QAs:** Replace images with JSON raw data to augmented previous QAs.
- **Data-driven QAs:** Prompt the model to extract JSON raw data before answering the question.

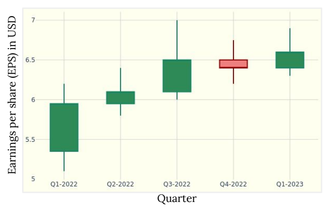
These QAs encompass a range of questions for chart images, covering abilities from basic data understanding and global concept comprehension to advanced reasoning, allowing us to further assess the abilities of MLLMs. Note that, for each QA pair, we use GPT-4 to generate both long and short answers. The long answer, generated first, includes a step-by-step explanation to derive the answer, while the short answer, generated later, contains only the final answer derived from the long explanation. Short answers contain only numerical values or Yes/No response for convenient evaluation purpose. For more examples of generated chart and QAs, please refer to Appendix H.

A.3 Dataset filtering

In Sec. 2.1, we introduce a novel data generation pipeline that leverages text-only LLMs. This pipeline enables us to collect chart images along with various data and QA pairs without extensive human effort, thereby reducing the cost of creating pairwise data. However, LLMs are not perfect

Chart Example 1

Tech Giant Quarterly Earnings Report



Literal QA

Q: What was the opening earnings per share (EPS) for Q4-2022? | Short A: 6.50 USD

Long A: The opening earnings per share (EPS) for Q4-2022 was 6.50 USD.

Inferential QA

Q: Was ... closing EPS was lower than the opening EPS? | Short A: Yes

Long A: Yes, in Q4-2022 ... of 6.40 USD was lower than the opening EPS of 6.50 USD.

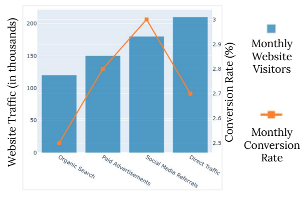
Reasoning QA

Q: How much ... the lowest of Q1-2022 to the highest of Q1-2023? | Short A: 1.80 USD

Long A: The EPS grew by 1.80 USD ... Q1-2022 at 5.10 USD to ...Q1-2023 at 6.90 USD.

Chart Example 2

Website Traffic vs Conversion Rate Analysis



Literal QA

Q: What is the website traffic from Paid Advertisements? | Short A: 150

Long A: According to ... Y-axis in the bar chart, the ... is 150 (in thousands).

Inferential QA

Q: Does Social Media Referrals lead in conversion rate? | Short A: Yes

Long A: Yes, Social ... at 3.0%, the highest among all categories ... line graph.

Reasoning QA

Q: What is the total website traffic from all sources? | Short A: 660

Long A: The total ... sum of ... : 120,000 + 150,000 + 180,000 + 210,000 = 660,000.

Figure A3: **Examples of generated three-level QAs with long and short answers**, accessing the understanding of charts from various perspectives. Best viewed in color.

and can make mistakes in either data generation or code script generation. Thus, in this section, we discuss the data filtering techniques we use to improve the quality of the synthetic dataset. The generation pipeline is split into three parts: shared template generation, data and QA generation, and code script generation. We now detail the filtering process for each part.

Shared template and README The shared template and README file for each chart type form the core of the entire data generation process, as the subsequent raw data, QA, and Python script are based on the shared template. Therefore, for the shared template, we deploy a human check to ensure the template contains necessary elements for the chart (i.e., title, x-axis, y-axis, data). Additionally, humans are required to verify the correctness of the chart type definitions in the README. Note that we consider 20 different chart types; thus, there are 20 template JSON files along with the READMEs in our dataset.

Data and QA generation Since all the data should follow the template JSON, for the data generation part, we apply filtering based on the JSON structure. Specifically, we remove generated data that deviates from the template file by comparing the elements in the keys of the JSON dictionary and the data types of all the values. As for QA generation, we check the structure of the output dictionary. In detail, the keys of the output QA dictionary should contain summary, description, literal, inferential, and reasoning QAs. We filter out QAs with missing attributes.

Code script generation We predefined a code expert GPT-4 to use four different Python libraries to plot the chart images: Matplotlib, Plotly, Pygal, and Seaborn. The advantage of these libraries is that if the Python code or input data is incorrect in terms of structure or other errors, the generated image will either be missing with a Python error or display a "No Data" icon. Thus, we apply a two-step filtering process: (1) Python Error Filtering: If there is an error while running the Python script to generate the image, we will remove the script and the corresponding JSON data. (2) OCR Tool Filtering: If the image is generated but there is some other error, the output image will display a "No Data" icon on it. To this end, we further use an OCR tool to detect whether there is any "No Data" icon in the images. If so, we will remove the data and script accordingly.

B Details of the generated benchmark

B.1 Motivations and Design Principle

To measure the comprehensive chart understanding ability of MLLMs, we assume that models can understand the underlying data of various chart types and perform QAs related to charts. These QAs

Table A3: Comparative analysis with existing benchmarks for chart understanding evaluations. * denotes unbounded chart types. Chart variation refers to whether the dataset contains chart images with different styles but sharing the same raw data.

Benchmark	# Image	# Chart type	Avg. # QAs per image	Multi-level QAs per image	Raw data per image	Chart Variation
PlotQA [20]	33.7k	3	1	✗	✗	✗
ChartQA [14]	1.5k	3	1	✗	✓	✗
Chart-to-text [6]	6.6k	6	1	✗	✗	✗
MMC [10]	2k	6	1	✗	✓	✗
Chartbench [26]	2.1k	9	9	✓	✗	✗
ChartX [25]	6k	18	1	✗	✓	✗
CharXiv [24]	2.3k	*	5	✓	✗	✗
Ours	5.48k	20	13.5	✓	✓	✓

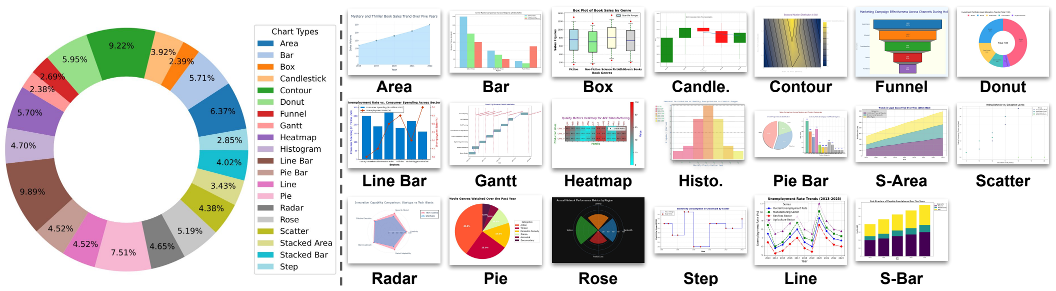


Figure A4: **Overview of our benchmark**. **Left**: Data distribution across different chart types. **Right**: Comprehensive list of chart types in the dataset with examples.

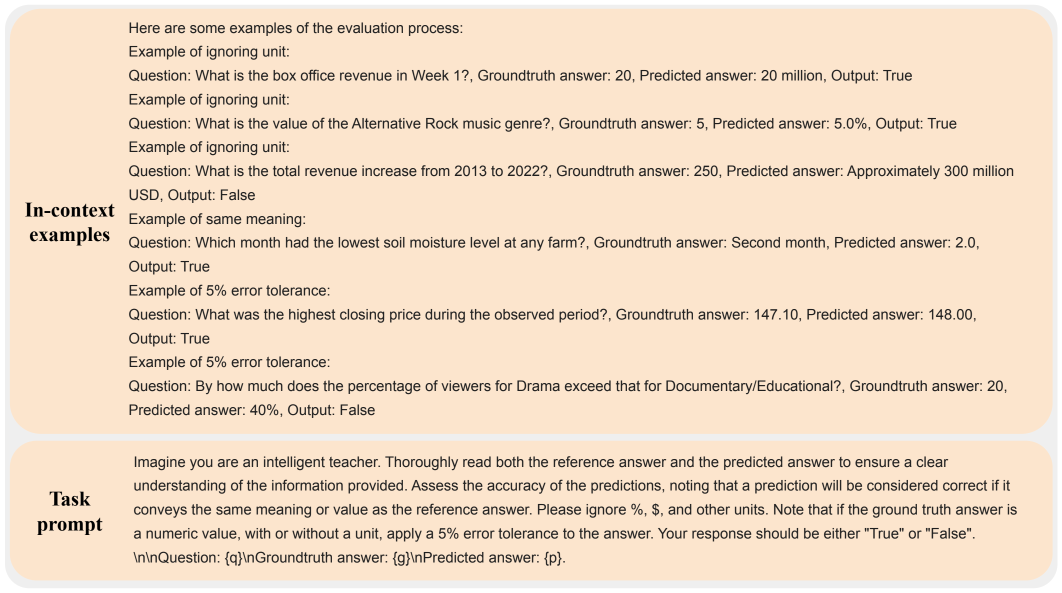
shouldn't be limited to simple questions about the title or x-axis; instead, they should range from basic to advanced QAs, requiring models to have a global conceptual understanding or even perform mathematical reasoning. However, existing benchmarks either lack a diverse range of chart types or fail to provide comprehensive QAs for each image, lack of a full assessment of understanding from multiple perspectives. To this end, we leverage our powerful data generation pipeline to generate a small set of data and apply filtering through an automatic pipeline and human evaluation. The resulting benchmark has several features to facilitate research in scientific chart understanding: (1) 20 different chart types, covering general use cases to scientific reports; (2) comprehensive QAs for each image, including literal, inferential, and reasoning questions, similar to the design of the training data; and (3) raw data for each image, measuring the models' ability to understand the underlying data. We provide a comparison with previous benchmarks in Table A3.

B.2 Dataset statistics

We provide the data statistics after filtering in Table A3. Additionally, a comprehensive list of all the chart types and the data distribution for each type is provided in Fig. A4. Specifically, for each chart type, we initially create $M = 30$ JSON data and $N = 12$ Python scripts. For each JSON dataset, we generate 15 QAs. As mentioned in Sec. 2.1, since all code scripts and data share the same structure, our generated data can be universally applied to any generated code and vice versa. In this way, we obtain approximately 360 unique chart images and around $5.4k$ chart-QA pairs for each chart type. After data filtering, we retain $74k$ chart-QA pairs in total, resulting in an average of $3.5k$ pairs per chart type and an average of 13.5 QAs per image. By applying multiple Python scripts to the same data point, our benchmark includes 608 unique data points, with an average of 9.2 variations per data point.

B.3 Data filtering

The benchmark is generated by our data generation pipeline, while the LLMs are not faultless, resulting in some noisy question-answer pairs. Unlike training data, noisy data can be problematic for benchmark, and cannot accurately access the model performance. Thus, to make sure the quality of the benchmark, we leverage automatic filtering process and human evaluation to filter chart images and question-answer pairs. We now detail the data filtering process of the proposed benchmark.



In-context examples

Here are some examples of the evaluation process:
Example of ignoring unit:
Question: What is the box office revenue in Week 1?, Groundtruth answer: 20, Predicted answer: 20 million, Output: True
Example of ignoring unit:
Question: What is the value of the Alternative Rock music genre?, Groundtruth answer: 5, Predicted answer: 5.0%, Output: True
Example of ignoring unit:
Question: What is the total revenue increase from 2013 to 2022?, Groundtruth answer: 250, Predicted answer: Approximately 300 million USD, Output: False
Example of same meaning:
Question: Which month had the lowest soil moisture level at any farm?, Groundtruth answer: Second month, Predicted answer: 2.0, Output: True
Example of 5% error tolerance:
Question: What was the highest closing price during the observed period?, Groundtruth answer: 147.10, Predicted answer: 148.00, Output: True
Example of 5% error tolerance:
Question: By how much does the percentage of viewers for Drama exceed that for Documentary/Educational?, Groundtruth answer: 20, Predicted answer: 40%, Output: False

Task prompt

Imagine you are an intelligent teacher. Thoroughly read both the reference answer and the predicted answer to ensure a clear understanding of the information provided. Assess the accuracy of the predictions, noting that a prediction will be considered correct if it conveys the same meaning or value as the reference answer. Please ignore %, $, and other units. Note that if the ground truth answer is a numeric value, with or without a unit, apply a 5% error tolerance to the answer. Your response should be either "True" or "False". \n\nQuestion: {q}\nGroundtruth answer: {g}\nPredicted answer: {p}.

Figure A5: Input prompt example for GPT-Acc. q, g, and p in the task prompt denote the question, ground truth answer, and predicted answer, respectively.

General data filtering. In the first stage of data filtering, we leverage the automatic data filtering pipeline used for our training data, as mentioned in Appendix A. Specifically, since the JSON data shares the same structure, we can filter out JSON files with incorrect dictionary structures. Furthermore, since the generated QAs follow a predefined dictionary structure, we can filter out QAs with missing attributes or keys. Lastly, when generating chart images using Python code and JSON, we record any execution errors and further filter out Python scripts that produce error messages.

Human evaluation - image filtering. To make sure the quality of the benchmark, we further adopt human evaluation. Recall that, in our benchmark, we have 608 unique data points, with an average of 9.2 variations per data point. To reduce the complexity and cost of human evaluation, we first invite human workers to check the image first, in which human workers have to go through all chart images and check the readibility of the chart. Specifically, workers will have to remove chart images if they have missing data points comparing to other variation or they are draw incorrect (potentially due to incorrect python code).

Human evaluation - QA filtering. After filtering the chart images, we then conduct QA filtering. In this step, we ask human workers to review 608 unique data points, with each data point having 15 QAs, resulting in approximately 9k test pairs. Specifically, for each test pair, we provide the human workers with a chart image along with the corresponding question and answers. Human workers are asked to filter the QA pairs based on two criteria: (1) Answerability: whether the question is answerable given the chart image, and (2) Correctness: whether the provided answer is correct. After collecting the feedback, we perform final data filtering to obtain the benchmark set.

B.4 Evaluation metrics

To quantitative analyze the performance on our benchmark, we adopt the relaxed accuracy metric for numeric answers, allowing a 5% margin of error from the exact value, and use exact match for non-numeric answers as per the standard in previous studies. Since the output of the MLLM model can be open form, following previous works [4, 10, 25], we also provide GPT-accuracy (GPT-Acc) using GPT model to further verify the performance. In Fig. A5, we show how we leverage the LLM to measure the accuracy by providing the detail prompts.[2] Lastly, for underlying data understanding task, we follow previous work Deplot [8] and measure performance using F1 score of Relative Mapping Similarity (RMS) and Relative Number Set Similarity (RNSS) to evaluate numeric accuracy and raw data similarity, respectively.

[2]GPT-4o-mini (2024-07-18)

Table A4: **Comprehensive evaluation on our benchmark.** Note that R-Acc and GPT-Acc denote relaxed accuracy and GPT accuracy, respectively.

Method	Literal QAs		Inferential QAs		Reasoning QAs		Overall		Extraction	
	R-Acc	GPT-Acc	R-Acc	GPT-Acc	R-Acc	GPT-Acc	R-Acc	GPT-Acc	F1	RNSS
GPT-4o	**43.98**	**47.62**	**57.23**	**59.43**	**23.08**	**26.15**	**41.40**	**44.40**	**66.10**	**88.56**
LLaVA$_{13B}$	8.96	8.68	24.91	21.76	2.46	2.46	12.11	10.97	9.22	45.82
ChartLlama$_{13B}$	21.29	21.01	38.05	35.22	8.92	8.62	22.60	21.50	11.40	64.14
ChartInstruct$_{7B}$	26.33	21.07	43.40	32.81	13.23	<u>25.23</u>	27.5	26.15	8.65	39.73
ChartAst$_{13B}$	27.64	24.09	38.22	32.70	14.65	17.97	26.55	24.87	16.73	68.04
CHOPINLLM	<u>34.45</u>	<u>44.82</u>	<u>56.92</u>	<u>58.18</u>	<u>21.85</u>	21.23	<u>37.50</u>	<u>41.40</u>	<u>28.42</u>	<u>75.44</u>

Table A5: **Ablation of stage-1 training.** This empirically verifies that pre-training basic chart visual perception is still important, even with abundant stage-2 instruction fine-tuning data. Moreover, learning to predict JSON data is beneficial, even on top of pre-training with descriptive captions.

Training data	ChartQA		**Our benchmark**		
	human	augmented	literal	inferential	reasoning
LLaVA-CC3M-Pretrain pairs [12]	44.80	83.92	41.45	34.09	22.31
+ Chart-description pairs	48.56	86.89	42.71	33.68	23.51
+ Chart-JSON data pairs	**52.28**	**87.68**	**44.96**	**34.94**	**24.61**

B.5 Performance of Existing Models

We evaluate existing models on our benchmark, and the results are provided in Table A4. We compare our model with four previous works, including ChartLlama [4], LLaVA [12], ChartInstruct [16], and ChartAst [19]. As shown in the table, our model consistently outperforms all existing works across three different QA levels and the raw data extraction task. We note that, compared to existing benchmarks, our benchmark includes chart images with a greater variety of chart types (i.e., 20 different chart types) and features comprehensive QAs for each image, along with raw data extraction to assess the fundamental understanding of scientific charts. Therefore, outperforming previous works demonstrates that our model has broader understanding of the chart domain and a more comprehensive understanding of the underlying data. Lastly, we also evaluate the GPT model on our benchmark. We find that while the GPT model excels at capturing global concepts (i.e., inferential QAs), its performance on raw data understanding tasks (i.e., literal QAs and Extraction) is below 50% accuracy. Moreover, for reasoning QAs, it achieves only $\approx 25\%$ accuracy, highlighting its lack of mathematical reasoning ability with chart data. We note that evaluating GPT-4o vision on the entire benchmark would be expensive. To deal with this while having fair comparison, all results in this table are evaluated on the same small subset of the benchmark, consisting of 1,000 randomly sampled chart QA pairs.

C Extra Experimental results

C.1 Stage 1: Pre-training for chart feature alignment

In the first training stage, the goal is to align visual and linguistic features so that visual data can be seamlessly translated into the textual domain for LLM comprehension. Employing a strategy from Liu et al. [12], we use a projector to translate visual features from ViT [3] into the textual domain, training it with pairwise image-caption data to enhance its capability to capture visual information. We explore three configurations: utilizing only LLaVA CC3M Pretraining data,[3] combining LLaVA data with chart-description pairs, and using LLaVA data with both chart-description and chart-raw data pairs. The data for stage two training remains consistent across these settings, summary QAs, description QAs, three-level QAs, text-only QAs, and data-driven QAs, as depicted in Fig. 2 (b). In stage three, all models undergo LoRA fine-tuning on the downstream dataset, using LLaVA-7B as the baseline for this comparison. Results are detailed in Table A5.

[3]https://huggingface.co/datasets/liuhaotian/LLaVA-CC3M-Pretrain-595K

Table A6: **Ablation of stage-2 training.** Each type of new instruction / QA data improves the final performance consistently across almost all metrics. Best result is highlighted in **Bold** and the second best is <u>underlined</u>. † denotes inference technique without extra data.

Training data	ChartQA		**Our benchmark**		
	human	augmented	literal	inferential	reasoning
LLaVA-Instruct-150K QAs	45.84	86.48	16.54	15.99	6.57
+ description and summary QAs	47.04	**87.76**	19.90	15.69	5.26
+ Literal / infer. / reasoning QAs	48.96	87.52	40.55	33.33	21.30
+ JSON-only QAs	49.60	87.36	41.45	34.84	22.36
+ Data-driven QAs	<u>52.28</u>	<u>87.68</u>	<u>44.96</u>	<u>34.94</u>	<u>24.61</u>
+ Data Prompting†	**56.96**	87.60	**52.00**	**41.75**	**31.90**

Dense data alignment is beneficial for both chart data comprehension and reasoning. For chart images, chart-description pairs act as standard image-caption pairs. However, to more effectively bridge the visual-textual gap, we also utilize chart-json pairs that encompass the underlying numerical data and its schema of the charts. This approach not only aligns visual features with textual descriptions but also significantly enhances model performance, as demonstrated by improvements of approximately 2% in literal QAs and about 1% in reasoning skills, according to results in Table A5.

C.2 Stage 2: End-to-end fine-tuning

The second stage, end-to-end fine-tuning, trains the MLLM to actually understand the aligned visual tokens so that it follows the user instruction and reason about the answer, on top of the inherent language capability from the original LLM. We utilize a significant number of image-QA pairs to jointly tune the LLM and the projector. To evaluate the effectiveness of incorporating chart QAs during fine-tuning, we conduct ablation studies starting with a baseline that uses only LLaVA Instruct-150K data,[4] incrementally adding extra QA pairs. All methods leverage the same pre-training weights, derived from training on LLaVA data with both chart-description and chart-raw data pairs (the best setting in Appendix C.1). In stage three, all models undergo LoRA fine-tuning on the downstream dataset. Comprehensive results are presented in Table A6.

JSON-only QAs allow transferring pure text reasoning abilities to multimodal chart understanding. The chart understanding of MLLMs can be seen as two stages: visual and text raw data alignment (which is done in the training of the first stage) and question answering with reasoning ability on the raw textual data (JSON). Thus, with a well-aligned first stage training, we hypothesize that re-blending some pure textual QAs, preserving the ability of reasoning on text raw data, can also benefit the reasoning abilities in visual-text scenarios. As detailed in Sec. 2.2, for JSON-only QAs, rather than utilizing chart images and QAs, we replace the chart image with JSON data and a README, resulting in purely text-based QAs for training. Table A6 demonstrates the effectiveness of each QA type. We discover that re-blending JSON-only data during the end-to-end fine-tuning stage improves chart reasoning skills, matching the assumption.

Data-driven QAs in the fine-tuning stage enable MLLMs to enhance prediction accuracy through data prompting. As detailed in Sec. 2.2, data-driven QAs are multi-turn QAs, which require models to extract raw data before answering given questions. Combined with the raw data reasoning abilities enhanced via JSON-only QAs, the model can perform data prompting during inference, where models achieve better reasoning robustness by first extracting raw data and then answering the given question based on the data. Please refer to Appendix G for some examples. As shown in Table A6, data-driven QAs significantly enhance the model's ability to capture visual information. Furthermore, leveraging data prompting in inference significantly improves performance across all downstream tasks.

[4]`https://huggingface.co/datasets/liuhaotian/LLaVA-Instruct-150K`

D Examples for JSON template and READMEs

D.1 Example 1

Template JSON

{ "chart_title": "Chart Title", "x_axis_label": "X-Axis Label", "y_axis_label": "Y-Axis Label", "data": [{ "category": "Category 1", "value": 0 }, { "category": "Category 2", "value": 0 }, { "category": "Category 3", "value": 0 }] }

README

The JSON template provided below is designed to generate data for a 'Bar Chart' chart. This template includes all the necessary attributes and labels to ensure consistent data formatting for the chart.
- `chart_title`: This attribute represents the title or name of the bar chart. Please replace the placeholder text with an appropriate title for your chart.
- `x_axis_label`: This attribute represents the label for the x-axis of the chart. Please replace the placeholder text with an appropriate label.
- `y_axis_label`: This attribute represents the label for the y-axis of the chart. Please replace the placeholder text with an appropriate label.
- `data`: This attribute represents the data points for the chart. It is an array of objects, where each object represents a category or group and its corresponding value.

For a simple example, the template can be filled as follows:

Definition of a 'Bar Chart' chart:
A bar chart, also known as a bar graph, is a visualization tool that uses rectangular bars to represent data. Each bar represents a category or group, and the length or height of the bar corresponds to the value it represents. Bar charts are commonly used to compare categorical data or show the distribution of data across different categories.

D.2 Example 2

Template JSON

{ "chart": { "title": "Chart Title", "xAxisLabel": "X-axis Label", "yAxisLabel1": "Y-axis Label 1", "yAxisLabel2": "Y-axis Label 2", "datasets": [{ "name": "Dataset 1", "type": "line", "data": [] }, { "name": "Dataset 2", "type": "bar", "data": [] }] } }

README

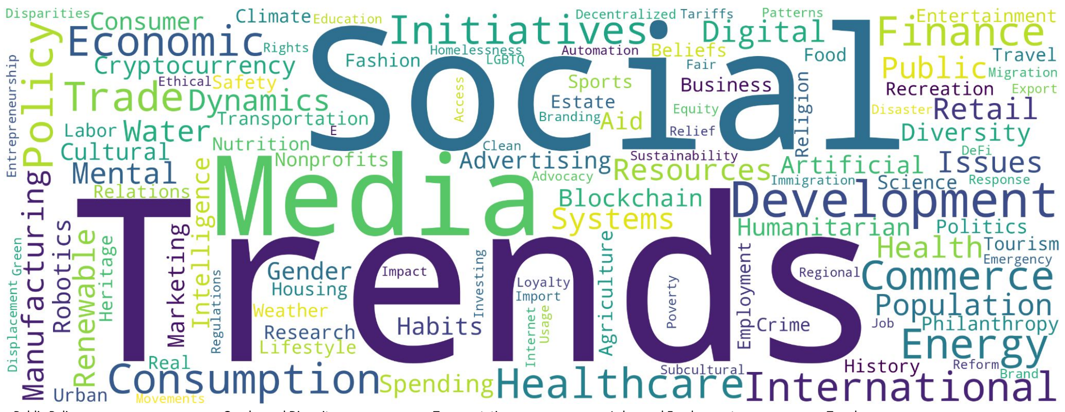
This JSON file template is designed for generating datasets for a 'Multi-axes Line Bar Chart' chart. It includes the following attributes:
1. chart:
  - title: (string) The title of the chart.
  - xAxisLabel: (string) The label for the X-axis.
  - yAxisLabel1: (string) The label for the Y-axis corresponding to the line chart.
  - yAxisLabel2: (string) The label for the Y-axis corresponding to the bar chart.
2. datasets:
  - name: (string) The name or label of the dataset.
  - type: (string) The type of chart component for the dataset. Can be 'line' or 'bar'.
  - data: (array) The array to store the data points for the dataset. Placeholder data should be added here.

The template provides a structure to accommodate both simple and complex examples of a Multi-axes Line Bar Chart. Additional datasets can be added within the "datasets" array.
Please ensure that the generated data adheres to the structure of this JSON template, including the attribute names and data types, to ensure consistency when plotting the chart using Python.

Definition of a 'Multi-axes Line Bar Chart':

A multi-axes line bar chart is a type of chart that combines both line and bar charts in a single visualization. It allows for the comparison of multiple datasets that have different scales or units of measurement. This chart type uses multiple y-axes, one for each dataset, to display the corresponding line and bar components.

E Examples for pre-defined topics



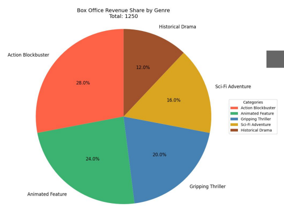
- Public Policy
- Healthcare Systems
- Mental Health
- Renewable Energy
- Water Resources
- Manufacturing
- Retail Trends
- Social Issues
- Population Dynamics
- Digital Media Consumption
- Cryptocurrency and Blockchain
- Humanitarian Aid and Development
- Gender and Diversity
- Economic Development
- Artificial Intelligence and Robotics
- Consumer Spending Habits
- Advertising and Marketing
- Cultural Trends
- Philanthropy and Nonprofits
- International Trade and Commerce
- Politics
- Business and Finance
- Science and Research
- Agriculture
- Transportation
- Weather and Climate
- Sports and Recreation
- Entertainment and Media
- Food and Nutrition
- Fashion and Lifestyle
- Housing and Real Estate
- Travel and Tourism
- Crime and Safety
- International Relations
- Religion and Beliefs
- History and Heritage
- Labor and Employment
- Urban Development
- Sustainability and Green Initiatives
- Education Policy and Reform
- Healthcare Access and Equity
- Clean Energy Initiatives
- E-commerce Trends
- Poverty and Homelessness
- Immigration and Migration Patterns
- Internet and Social Media Usage
- Trends
- Decentralized Finance (DeFi)
- Trends
- Disaster Relief and Emergency Response
- LGBTQ+ Rights and Advocacy
- Regional Economic Disparities
- Automation and Job Displacement
- Branding and Brand Loyalty
- Subcultural Trends and Movements
- Social Entrepreneurship and Impact
- Investing
- Fair Trade and Ethical Consumption
- Import-Export Regulations and Tariffs

We provide a word cloud in the figure above to show the frequency of each word in the defined topic set. A comprehensive list of all the topics is also provided at the bottom of this figure.

F Examples of augmented QAs

F.1 JSON-only QA: example 1

Chart image



JSON & README

```
{  "title": "Box Office Revenue Share by Genre",
    "description": "This pie chart displays the distribution of box
office revenues among five major movie genres in 2021,
illustrating the diversity in consumer preferences and the strategic
positioning of movie studios.",
    "data": [{
            "category": "Action Blockbuster",
            "value": 350,
            "color": "#FF6347"
        },
        {
            "category": "Animated Feature",
            "value": 300,
            "color": "#3CB371"
        },
        {
            "category": "Gripping Thriller",
            "value": 250,
            "color": "#4682B4"
        },
        {
            "category": "Sci-Fi Adventure",
            "value": 200,
            "color": "#DAA520"
        },
        {
            "category": "Historical Drama",
            "value": 150,
            "color": "#A0522D"
        }
    ]
}
```

Title: The title of the pie chart, which gives an idea of the dataset's overall topic.

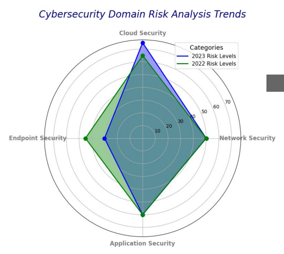
Description: A short summary explaining what the pie chart data represents and any additional information that might be useful to understand the context.

Data: An array of objects where each object represents a slice of the pie chart.
- category: A string representing the name of the category this slice of the pie chart is about.
- value: A numerical value indicating the size of the category in the dataset. This determines the size of the pie slice.
- color: A string indicating the color of this slice. It can be a HEX color code or a standard color name.

The JSON template and README file are designed to ensure that datasets for pie charts can be created with consistent structure and clarity, accommodating both simple and complex use cases. Make sure to replace placeholder text and values with actual data relevant to the pie chart you are creating.

F.2 JSON-only QA: example 2

Chart image



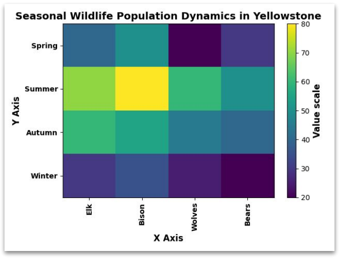
JSON & README

```
{
    "chartTitle": "Cybersecurity Domain Risk Analysis Trends",
    "datasets": [
        {
            "label": "2023 Risk Levels",
            "data": {
                "Network Security": 50,
                "Cloud Security": 75,
                "Endpoint Security": 30,
                "Application Security": 60
            }
        },
        {
            "label": "2022 Risk Levels",
            "data": {
                "Network Security": 50,
                "Cloud Security": 65,
                "Endpoint Security": 45,
                "Application Security": 60
            }
        }
    ],
    "options": {
        "scale": {
            "min": 0,
            "max": 100,
            "stepSize": 10
        }
    }
}
```

Chart Title: The title of the Radar Chart, which usually describes the overall data being represented.
Datasets: An array of objects, each representing a different dataset to be plotted on the Radar Chart.
Dataset Label: A descriptive name for the dataset. This could be the name of a product, an individual's name, or any other identifier relevant to the data.
Data: An object containing key-value pairs where the key is the label for the axis (e.g., 'Communication', 'Battery Life') and the value is the data point associated with that axis. The number of axes can vary depending on the use case.
Axis Label: Placeholders for the actual labels of each axis on the Radar Chart. Replace these placeholders with the appropriate criteria or variables for your specific use case.
Options: An object containing additional settings and configurations for the Radar Chart.

Scale: Defines the range and increment of the chart's axes.
Min: The minimum value of the scale (usually 0).
Max: The maximum value of the scale, which should be set according to the data's range.
StepSize: The interval between values on the scale.

F.3 Data-driven QA: example 1



An example of data extraction QAs

Can you please extract the raw data of this chart?

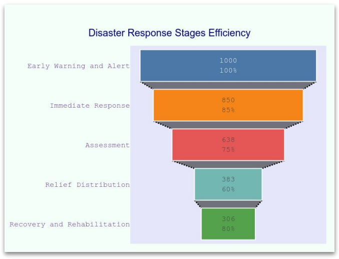
{ "title": "Seasonal Wildlife Population Dynamics in Yellowstone", "xAxis": { "title": "Species", "categories": ["Elk", "Bison", "Wolves", "Bears"] }, "yAxis": { "title": "Seasons", "categories": ["Spring", "Summer", "Autumn", "Winter"] }, "colorScale": { "minValue": 0, "maxValue": 100, "startColor": "#FFFFFF", "endColor": "#FF6347" }, "data": [{ "x": 0, "y": 0, "value": 40 }, { "x": 0, "y": 1, "value": 70 }, { "x": 0, "y": 2, "value": 60 }, { "x": 0, "y": 3, "value": 30 }, { "x": 1, "y": 0, "value": 50 }, { "x": 1, "y": 1, "value": 80 }, { "x": 1, "y": 2, "value": 55 }, { "x": 1, "y": 3, "value": 35 }, { "x": 2, "y": 0, "value": 20 }, { "x": 2, "y": 1, "value": 60 }, { "x": 2, "y": 2, "value": 45 }, { "x": 2, "y": 3, "value": 25 }, { "x": 3, "y": 0, "value": 30 }, { "x": 3, "y": 1, "value": 50 }, { "x": 3, "y": 2, "value": 40 }, { "x": 3, "y": 3, "value": 20 }] }

Which season generally shows the highest population densities across all species?

Long answer Reviewing the heatmap data, Summer has high values for all species: Elk (70), Bison (80), Wolves (60), and Bears (50). Thus, Summer generally shows the highest population densities.

Short answer Summer

F.4 Data-driven QA: example 2



An example of data extraction QAs

Can you please extract the raw data of this chart?

{ "chart": { "title": "Disaster Response Stages Efficiency", "description": "This funnel chart maps the stages of disaster response from the initial alert to recovery, highlighting the number of actions completed and the conversion rates between each stage. The cyclical trend observed suggests repeated challenges in transitioning efficiently from one stage to the next." }, "stages": [{ "stage": "Early Warning and Alert", "value": 1000, "conversionRate": "85%" }, { "stage": "Immediate Response", "value": 850, "conversionRate": "75%" }, { "stage": "Assessment", "value": 638, "conversionRate": "60%" }, { "stage": "Relief Distribution", "value": 383, "conversionRate": "80%" }, { "stage": "Recovery and Rehabilitation", "value": 306, "conversionRate": null }] }

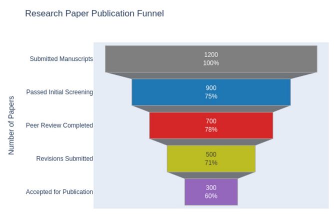
What is the sum of conversion rates for all stages where it is mentioned?

Long answer The sum of conversion rates for all stages mentioned is calculated by adding the rates: 85% (Early Warning and Alert) + 75% (Immediate Response) + 60% (Assessment) + 80% (Relief Distribution) = 300%.

Short answer 300%

G Examples of data prompting

G.1 Example 1



Without data extraction QA in the inference

What is the acceptance rate of manuscripts that have been revised?

50%

With data extraction QA in the inference

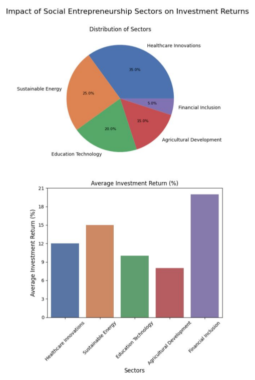
Retrieve the raw data from the image and organize it into a JSON dictionary.

{'chart': {'title': 'Research Paper Publication Funnel', 'xAxisLabel': 'Stage', 'yAxisLabel': 'Number of Papers'}, 'data': [{'stage': 'Submitted Manuscripts', 'count': 1200}, {'stage': 'Passed Initial Screening', 'count': 900}, {'stage': 'Peer Review Completed', 'count': 700}, {'stage': 'Revisions Submitted', 'count': 500}, {'stage': 'Accepted for Publication', 'count': 300}]}

What is the acceptance rate of manuscripts that have been revised?

60%

G.2 Example 2



Without data extraction QA in the inference

What proportion of the total distribution is made up by 'Sustainable Energy' and 'Healthcare Innovations' combined?

45%

With data extraction QA in the inference

Retrieve the raw data from the image and organize it into a JSON dictionary.

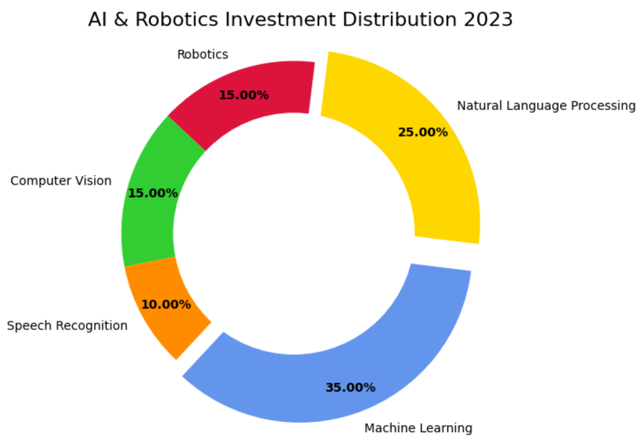
{'chart_title': 'Impact of Social Entrepreneurship Sectors on Investment Returns', 'x_axis_label': 'Sectors', 'y_axis_label_pie': 'Distribution of Sectors', 'y_axis_label_bar': 'Average Investment Return (%)', 'data': [{'category': 'Healthcare Innovation', 'value_pie': 35, 'value_bar': 12}, {'category': 'Sustainable Energy', 'value_pie': 25, 'value_bar': 15}, {'category': 'Education Technology', 'value_pie': 20, 'value_bar': 10}, {'category': 'Agricultural Development', 'value_pie': 15, 'value_bar': 8}, {'category': 'Financial Inclusion', 'value_pie': 5, 'value_bar': 20}]}

What proportion of the total distribution is made up by 'Sustainable Energy' and 'Healthcare Innovations' combined?

60%

H Examples from our benchmark

H.1 Example 1



JSON Data

{'chart': {'type': 'donut', 'title': 'AI & Robotics Investment Distribution 2023'}, 'data': {'labels': ['Machine Learning', 'Natural Language Processing', 'Robotics', 'Computer Vision', 'Speech Recognition'], 'datasets': [{'label': 'Investment Proportions', 'data': [35, 25, 15, 15, 10], 'backgroundColor': ['#6495ED', '#FFD700', '#DC143C', '#32CD32', '#FF8C00']}]}}

Literal Question

Question:	How much less investment did Speech Recognition receive compared to Natural Language Processing?
Long Answer:	Speech Recognition received 15% less investment than Natural Language Processing, with Speech Recognition at 10% and Natural Language Processing at 25%.
Short Answer:	15%

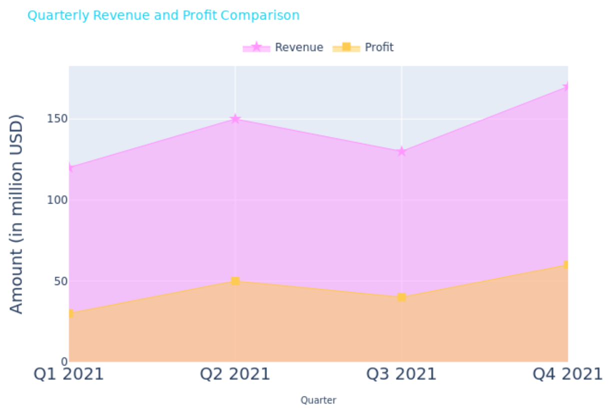
Inferential Question

Question:	What two sectors together make up half of the total investment?
Long Answer:	Machine Learning and Natural Language Processing together make up half of the total investment, with percentages of 35% and 25% respectively.
Short Answer:	Machine Learning and Natural Language Processing

Reasoning Question

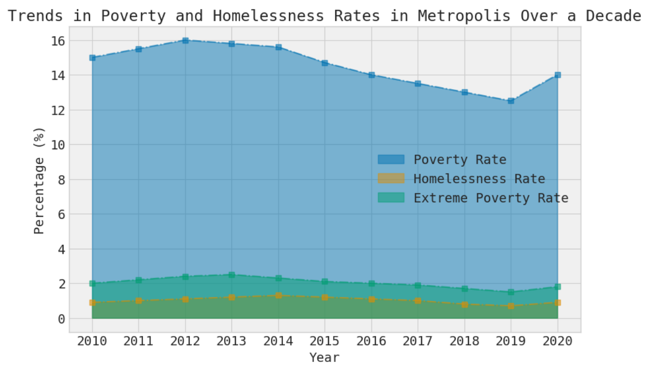
Question:	What fraction of the total investment is allocated to fields other than Machine Learning?
Long Answer:	Fields other than Machine Learning receive a combined total of 65% of the investment, which is equivalent to the fraction 65/100 or 13/20 of the total investment.
Short Answer:	13/20

H.2 Example 2



JSON Data	
{'chart': {'title': 'Quarterly Revenue and Profit Comparison', 'xAxisLabel': 'Quarter', 'yAxisLabel': 'Amount (in million USD)'}, 'data': [{'category': 'Revenue', 'values': [{'x': 'Q1 2021', 'y': 120}, {'x': 'Q2 2021', 'y': 150}, {'x': 'Q3 2021', 'y': 130}, {'x': 'Q4 2021', 'y': 170}]}, {'category': 'Profit', 'values': [{'x': 'Q1 2021', 'y': 30}, {'x': 'Q2 2021', 'y': 50}, {'x': 'Q3 2021', 'y': 40}, {'x': 'Q4 2021', 'y': 60}]}]}	
Literal Question	
Question:	What was the Revenue in Q3 2021?
Long Answer:	In Q3 2021, the Revenue was 130 million USD as shown on the chart.
Short Answer:	130 million USD
Inferential Question	
Question:	Which quarter had the highest ratio of Profit to Revenue?
Long Answer:	To determine the highest ratio of Profit to Revenue, we compare the ratios for each quarter. The highest ratio is in Q2 2021, with Profit at 50 million USD and Revenue at 150 million USD, giving a ratio of 1:3.
Short Answer:	Q2 2021
Reasoning Question	
Question:	If Profit in Q1 2022 is expected to be 20% higher than Q4 2021, what would be the expected Profit?
Long Answer:	If Profit in Q1 2022 is expected to be 20% higher than Q4 2021's 60 million USD, the expected Profit would be 60 * 1.20, which is 72 million USD.
Short Answer:	72 million USD

H.3 Example 3



JSON Data

{'chart': {'title': 'Trends in Poverty and Homelessness Rates in Metropolis Over a Decade', 'xAxisLabel': 'Year', 'yAxisLabel': 'Percentage (%)'}, 'data': [{'category': 'Poverty Rate', 'values': [{'x': '2010', 'y': 15.0}, {'x': '2011', 'y': 15.5}, {'x': '2012', 'y': 16.0}, {'x': '2013', 'y': 15.8}, {'x': '2014', 'y': 15.6}, {'x': '2015', 'y': 14.7}, {'x': '2016', 'y': 14.0}, {'x': '2017', 'y': 13.5}, {'x': '2018', 'y': 13.0}, {'x': '2019', 'y': 12.5}, {'x': '2020', 'y': 14.0}]}, {'category': 'Homelessness Rate', 'values': [{'x': '2010', 'y': 0.9}, {'x': '2011', 'y': 1.0}, {'x': '2012', 'y': 1.1}, {'x': '2013', 'y': 1.2}, {'x': '2014', 'y': 1.3}, {'x': '2015', 'y': 1.2}, {'x': '2016', 'y': 1.1}, {'x': '2017', 'y': 1.0}, {'x': '2018', 'y': 0.8}, {'x': '2019', 'y': 0.7}, {'x': '2020', 'y': 0.9}]}, {'category': 'Extreme Poverty Rate', 'values': [{'x': '2010', 'y': 2.0}, {'x': '2011', 'y': 2.2}, {'x': '2012', 'y': 2.4}, {'x': '2013', 'y': 2.5}, {'x': '2014', 'y': 2.3}, {'x': '2015', 'y': 2.1}, {'x': '2016', 'y': 2.0}, {'x': '2017', 'y': 1.9}, {'x': '2018', 'y': 1.7}, {'x': '2019', 'y': 1.5}, {'x': '2020', 'y': 1.8}]}]}

Literal Question

Question:	In which year did the Poverty Rate reach its lowest value?
Long Answer:	According to the chart data, the Poverty Rate reached its lowest value in 2019 at 12.5%.
Short Answer:	2019

Inferential Question

Question:	Did any category show a consistent decline over the entire decade?
Long Answer:	No category showed a consistent decline over the entire decade. While Poverty Rate and Extreme Poverty Rate generally declined until 2019, they both increased in 2020, and the Homelessness Rate fluctuated throughout the decade.
Short Answer:	No

Reasoning Question

Question:	What is the average annual decrease in the Poverty Rate from 2010 to 2019?
Long Answer:	From 2010 to 2019, the Poverty Rate decreased from 15.0% to 12.5%. This is a total decrease of 2.5 percentage points over 9 years, which gives an average annual decrease of about 0.278 percentage points per year.
Short Answer:	Approximately 0.278 percentage points per year