Table-First Guarded Reasoning: Outperforming Larger MLLMs in Chart Question Answering

***Abstract*—Multimodal Large Language Models (MLLMs) have revolutionized visual understanding but frequently suffer from "visual hallucination" when extracting precise numerical values from charts. Existing solutions often mitigate this by scaling parameters to massive sizes (e.g., 70B+ models), creating significant computational barriers for resource-constrained environments. To address this, we introduce Table-First Guarded Reasoning (TFGR), a resource-efficient pipeline that decouples visual data extraction from logical reasoning. By integrating a dedicated chart-to-table extraction module (Chart2Table) with Zero-Shot Chain-of-Thought (CoT) prompting, we provide the model with structured tabular context to "guard" against perception errors. We evaluate our approach on the ChartQA benchmark using the Qwen2.5-VL-7B-Instruct model. Our pipeline achieves 93.96% accuracy, significantly outperforming the 88.00% baseline. Notably, our 8.87B-parameter architecture (when accounting for Chart2Table) surpasses the State-of-the-Art Qwen2-VL-72B (88.20%) [1] on the OpenVLM Leaderboard, demonstrating that architectural innovation can substitute for raw parameter scale. We further validate our findings through an extensive analysis of explorative experiments in fine-tuning, architectural alternatives, and complementary tools.**

***Index Terms*—Multimodal LLM, Chart Understanding, Chain-of-Thought, Table-First Reasoning, Resource-Efficient AI**

I. Introduction

Data visualization serves as a cornerstone of modern decision-making, transforming complex datasets into accessible insights. With the growing need of charts in financial reports, scientific literature, and business intelligence, the ability to automatically interpret these visual artifacts is increasingly critical. Multimodal Large Language Models (MLLMs) have emerged as powerful tools for this task, offering the potential to answer natural language queries about charts directly. However, despite their impressive capabilities in general image captioning, MLLMs frequently struggle when tasked with precise Chart Question Answering (CQA).

A pervasive challenge in this domain is "visual hallucination", where a model correctly identifies visual trends (e.g., "sales are increasing") but fabricates or misinterprets the specific numerical values required to prove the claim. For instance, a model might misread a bar chart value of "23.5" as "25" due to insufficient resolution or poor alignment between its vision encoder and language head. To mitigate these errors, the prevailing trend has been parameter scaling (e.g., Qwen2-VL-72B [1], GPT-4V [2]), which requires substantial computational resources. This reliance on scale creates a significant barrier of inaccessibility to rendering high-accuracy chart understanding or real-time applications for research environments with limited computing power.

This paper challenges the paradigm that "bigger is better" by introducing a resource-efficient alternative that prioritizes architectural modularity over parameter density. We propose Table-First Guarded Reasoning (TFGR), a pipeline designed to maximize the performance of smaller, open-source models, specifically Qwen2.5-VL-7B-Instruct[3]. Our core hypothesis is that the tasks of visual perception (reading numbers from a grid) and logical reasoning (computing averages or differences) should be decoupled by converting the visual chart into a structured textual representation (tabular data) via an external module. We effectively "guard" the LLM against its own visual hallucinations. This allows a 8.29B-parameter model to leverage its strong textual processing capabilities on ground-truth data, rather than relying on noisy visual embeddings.

Our approach integrates Chart2Table[4], a specialized extraction tool utilizing OCR and structural recognition, with Zero-Shot Chain-of-Thought (CoT) prompting. This combination forces the model to perform explicit lookups and arithmetic operations based on the extracted table, rather than approximating values from the image alone. The result is a system that not only rivals but surpasses the performance of models almost ten times its size.

The main contribution our paper provides is the proposed pipeline referenced above as a combination and augmentation of existing approaches. Its ability to achieve a higher accuracy in chart question answering (CQA) while using a fraction of the parameters through data tabulation and CoT prompting opens the room for more accessible MLLMs that specialize in CQA when the previous state of the art required over 72B parameters. Furthermore, usage of Chart2Table[4] in this application remained woefully under-documented until our work contributed to examining how it performs in place of DePlot [5].

The remainder of the paper is organized as follows: Section

II delves into related work done on the subject; Section III explains the methodology proposed by the paper; Section IV discusses the various experiments we ran during the research period; and finally, Section V concludes the paper and offers insight into possible future work.

II. Related Work

Over the years, the problem of augmenting MLLMs capability of handling CQA was tackled by a variety of papers and researchers. Approaches ranged from changes in the end-to-end architecture to the plot-to-table extraction and CoT that inspired our approach.

A. Multimodal Large Language Models in Chart Understanding

The evolution of Chart Question Answering has shifted from heuristic-based parsers to end-to-end Multimodal Large Language Models. Early foundational models such as MatCha and UniChart attempted to unify visual and textual modalities by pre-training on massive chart-text pairs [6], [7]. MatCha enhanced mathematical reasoning by simulating visual language tasks, while UniChart utilized a unified encoder-decoder architecture.

However, as noted in recent studies [8], these end-to-end architectures often struggle with the "resolution gap" and dense information retrieval. Specifically, Zeng *et al.* [9] identified that models like UniChart frequently lose critical spatial and color information during the projection of visual features into the LLM's token space. Consequently, while they excel at general captioning, they are prone to "visual hallucinations" when tasked with precise numerical extraction from complex charts [5]. Furthermore, deploying massive proprietary models (e.g., GPT-4V [2]) remains computationally prohibitive for many applications, creating a need for efficient, open-source alternatives.

B. Intermediate Representations: Plot-to-Table Extraction

To mitigate the precision limitations of end-to-end models, recent research has explored Table-First approaches that decouple visual perception from reasoning. The primary strategy involves converting the non-sequential visual chart into a sequential, intermediate textual representation (i.e., a data table) [5].

DePlot [5] pioneered this "plot-to-table" translation, fine-tuning a visual language model (Pix2Struct [10]) to generate linearized tables from chart screenshots. This allows the downstream LLM to reason over text, a modality where it exhibits stronger performance. However, our explorative experiments and literature review suggest that DePlot suffers from significant domain sensitivity. When the target charts (e.g., ChartQA) differ stylistically from DePlot's synthetic training data, the model often generates structurally correct but factually erroneous tables.

In contrast, our work utilizes Chart2Table, a retrieval-based module leveraging Optical Character Recognition (OCR) and structural parsing (based on PaddlePaddle [11]). Unlike DePlot's generative approach, which can "hallucinate" data, Chart2Table provides a deterministic digitization of the chart content, offering superior robustness for numerical precision.

C. Prompting Strategies and the Reasoning Gap

Chain-of-Thought (CoT) prompting has become a standard technique for drawing out complex reasoning in LLMs [12], [13]. Research indicates that "Split CoT" strategies can further enhance performance by breaking down complex queries [14]. However, a critical gap remains: standard Multimodal CoT relies entirely on the quality of the initial visual encoding. If the vision encoder misinterprets a data point, the subsequent reasoning chain is grounded in false premises.

Existing literature highlights that pure CoT is insufficient for high-precision CQA without reliable data grounding [15]. Our TFGR pipeline addresses this by fusing the robustness of OCR-based extraction (Chart2Table) with the logical structuring of CoT. This "guards" the 8.29B-parameter model against perceptual errors, allowing it to outperform significantly larger models like Qwen2.5-VL-7B-Instruct that rely on end-to-end processing.

III. Methodology: Table-First Guarded Reasoning

The papers that influenced our approach the most were Liu *et al.*'s work on DePlot [5] and the work done on prompting strategies [12] [13] [14] [15]. We were not satisfied with merely copying previous work and wanted to test if the approaches would be compatible with one another to achieve greater improvements. So we found Chart2Table[4], which was only a few months old at that point and noted the lack of papers discussing it. We capitalized on the opportunity to test it in conjunction with our existing approach to push our pipeline to its limit.

A. Pipeline Architecture

Our architecture avoids the traditional end-to-end approach where a single model attempts to simultaneously parse visual features and compute answers. Instead, we implement a Chain-of-Models flow. The pipeline consists of three sequential stages:

1) Visual Extraction: A specialized module digitizes the chart image into a structured tabular format. It is represented by the Chart2Table module in Fig. 1.
2) Context Integration: The extracted table is injected into the prompt alongside the original image, creating a dual-modality context which is represented by the "Visual Tokens" and "Table Context" blocks in the "Guarded Prompting Zone" in Fig. 1.
3) Guarded Reasoning: The MLLM (Qwen2.5-VL-7B-Instruct) processes this enriched context (represented by the "Reasoning Engine" in Fig. 1.) using a specialized Zero-Shot CoT framework (represented by the "System Instruction" block in the "Guarded Prompting Zone" in Fig. 1.) to derive the final answer.

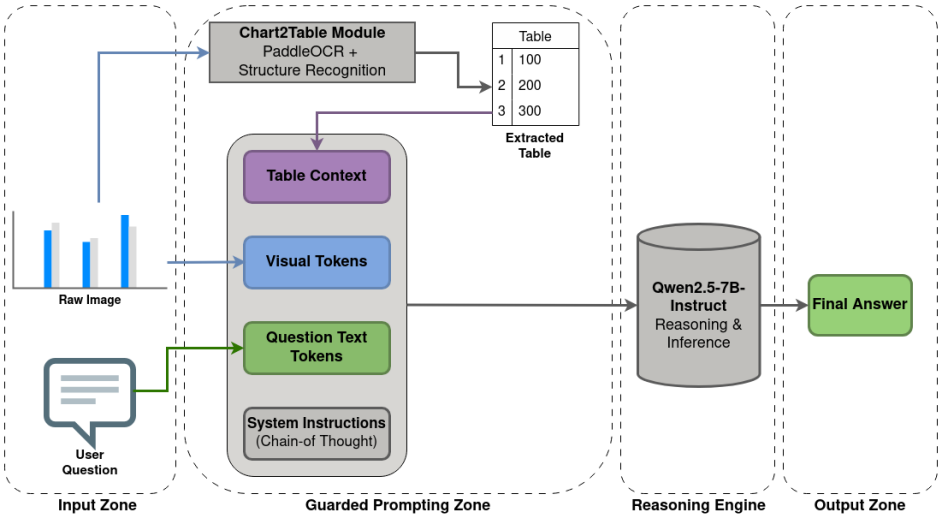

Fig. 1. TFGR Architecture

B. Visual Extraction: The Table-First Approach

A critical design decision in our methodology was the selection of the extraction backend. We evaluated two distinct approaches to digitize chart data: generative translation versus structural recognition.

1) Generative Translation (DePlot): We initially experimented with DePlot, a Pix2Struct-based model that treats extraction as a "plot-to-table" translation task. DePlot generates a linearized table directly from the image tokens. While DePlot improved our baseline accuracy (88.00%) to 92.84%, we observed some instability due to domain shifts. Because DePlot is a generative model, it occasionally "hallucinates" data values or table structures when the input chart style differs from its training distribution.
2) Structural Recognition (Chart2Table): To mitigate generative hallucinations, we adopted Chart2Table, a retrieval-based module built on the PaddlePaddle OCR framework. Unlike DePlot, Chart2Table employs a deterministic pipeline: it detects text via OCR and maps it to coordinates using table structure recognition.
   - Robustness: By relying on optical character recognition rather than token generation, Chart2Table offers higher fidelity for exact number retrieval.
   - Architecture: We deployed Chart2Table as a separate microservice (REST API), allowing it to run independently of the LLM running using pytorch, thereby preventing CUDA dependency version conflicts on our RTX 3090 hardware caused by PaddlePaddle using different versions.

   Our experiments confirmed that Chart2Table provided the most robust grounding, elevating the pipeline accuracy to 93.96%. Consequently, Chart2Table became the foundational extraction layer for the TFGR pipeline.

C. Guarded Reasoning: Zero-Shot Chain-of-Thought

The "Guarded" aspect of our methodology refers to the use of structured prompting to constrain the LLM's reasoning process. We utilize a Zero-Shot Chain-of-Thought (CoT) strategy that forces the model to cross-reference the visual image with the extracted table before answering.

We define a 6-step reasoning framework (as seen in Fig. 2.) that acts as a cognitive guardrail:

1) State Chart Type: The model must explicitly state the type, for example, whether the image is a bar, line, or pie chart.
2) Find Relevant Table Columns: The model is instructed to find the relevant columns in the extracted table.
3) Cross-Verification: The prompt explicitly requires the model to "Cross-verify information between table and chart" to ensure the OCR output matches the visual trend.
4) Step-by-Step Arithmetic: All arithmetic operations must be shown step-by-step (e.g., "103.7 - 103.13 = 0.57").
5) Sanity Check: A verification step to ensure the calculated result aligns with visual intuition.
6) Output Tokens: The final answer's output must conclude with a specific formatting token ("Final Answer:") to facilitate automated extraction.

This approach transforms the role of our 8.29B model from a "guesser" to an "analyst", leveraging the extracted table as ground truth while using the image for visual context the table cannot provide such as colors.

D. Evaluation Function and Standardization

To ensure our findings are valid and comparable to the State-of-the-Art (SOTA), we implemented a rigorous evaluation algorithm strictly aligned with the OpenVLM ChartQA benchmarks.

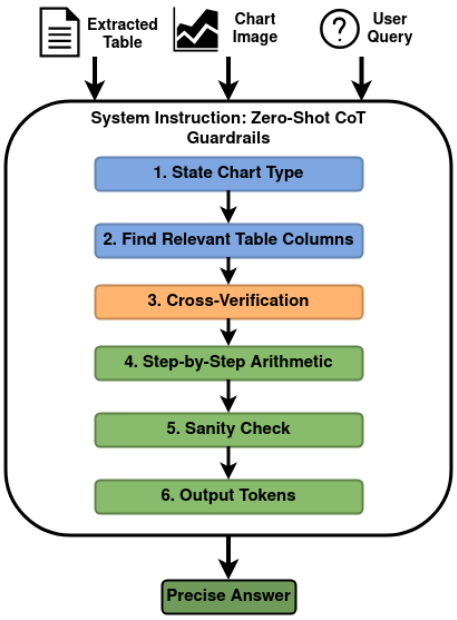

Fig. 2. Guarded Reasoning Framework

1) Metric Definition: We define accuracy based on a 5% relative error tolerance. Standard exact-match metrics are insufficient for floating-point calculations in charts (e.g., distinguishing between 1000 and 1001 is often impossible visually). The relative error is calculated as:

$$\text{Relative Error} = \frac{|\text{Predicted} - \text{Ground Truth}|}{|\text{Ground Truth}|} \quad (1)$$

If the relative error is less than or equal to 0.05, the answer is considered correct.

2) Multi-Stage Normalization: To handle the linguistic variance of LLMs, we implement a 5-stage normalization pipeline:
   - JSON & Pattern Extraction: We extract answers from structured JSON or regex patterns (e.g., "Final Answer: X").
   - Format Sanitization: We strip currency symbols, commas, and percentage signs (e.g., "$1,234" → "1234").
   - Word-to-Digit Conversion: Textual numbers are converted to digits (e.g., "twenty" → "20") to enable numerical comparison.
3) List and decimal Handling: A critical implementation detail involved handling the ChartQA dataset's list format, and our evaluation function that validates the prediction against the primary ground truth element. Furthermore, we implemented decimal protection to prevent truncation errors (e.g., ensuring "0.57" is not parsed as "0").

By adhering to this strict evaluation protocol, we ensure that our reported accuracy of 93.96% is a genuine reflection of the model's capabilities and directly comparable to leaderboard standings.

IV. EXPERIMENTS AND RESULTS

Over the course of our research, we conducted a variety of experiments. We tested a variety of approaches to test their promise in our greater plan. The pipeline we settled on was tested on ChartQA's test split, and we decomposed the pipeline and tested parts separately to measure their contribution. However, there were also a variety of explorative experiments along the way.

A. Experimental Setup

All experiments were conducted on a workstation running Ubuntu 24.04.3 LTS Linux, equipped with a single NVIDIA GeForce RTX 3090 GPU (24GB VRAM). This hardware constraint serves as an environment and validation of our high-performance pipeline for resource-limited environments.

We utilized the ChartQA dataset [16], strictly adhering to the standard Train/Validation/Test splits, and utilizing the exact evaluation function employed by the OpenVLM ChartQA_TEST Leaderboard. This ensures that our results are directly comparable to existing models. The base model for our pipeline is Qwen2.5-VL-7B-Instruct, selected for its strong instruction-following capabilities and efficient parameter size.

B. Performance Analysis

We systematically evaluated the incremental contributions of each component in our proposed TFGR pipeline. The results are summarized in Table I by method where each method has a certain configuration of approaches noted in the "Configuration" column. The accuracy achieved and the percentage difference compared to the baseline are present in the last two columns of Table I.

TABLE I
ABLATION STUDY OF PIPELINE COMPONENTS ON CHARTQA TEST SET

Method	Configuration	Accuracy	Δ vs Baseline
Baseline	Qwen2.5-VL-7B-Instruct	88.00%	–
Reasoning	Baseline + Zero-Shot CoT	90.84%	+2.84%
Generative Extraction	Baseline + CoT + DePlot	92.84%	+4.84%
TFGR	Baseline + CoT + Chart2Table	93.96%	+5.96%

1) Baseline Performance: Qwen2.5-VL-7B-Instruct achieved 88.00%; frequent visual hallucinations persisted.
2) CoT Impact: Zero-Shot CoT raised accuracy to 90.84% (+2.84%) by improving multi-step reasoning.
3) Extraction Impact: Adding tabular grounding produced the largest gains. Chart2Table (93.96%) exceeded DePlot (92.84%) by avoiding generative hallucinations.

C. Inference Latency Analysis

We analyzed the computational cost of our approach. Table II details the inference latency per configuration with TFGR proving to be the slowest on average.

TABLE II
INFERENCE LATENCY COMPARISON

Method	Configuration	Avg. Latency (sec)
Baseline	Qwen2.5-VL-7B-Instruct	1.48
Reasoning	Baseline + Zero-Shot CoT	6.64
TFGR	Baseline + CoT + Chart2Table	9.38

D. Explorative Experiments

To ensure rigorous academic inquiry, we conducted extensive explorative experiments. Several approaches, despite their theoretical promise, yielded suboptimal results. We report these results to guide future research.

1) Fine-Tuning Limitations: We attempted to fine-tune Qwen2.5-VL-7B-Instruct directly on the ChartQA dataset using QLoRA. Surprisingly, this resulted in lower accuracy than the baseline. We hypothesize that fine-tuning on a relatively small, domain-specific dataset caused catastrophic forgetting of the model's general reasoning capabilities. Similarly, fine-tuning on Chart2Text datasets yielded no improvements. A similar phenomenon was noted when using standard 8-bit quantization in the introduction of Dettmers *et al.* [17].
2) Architectural Alternatives:
   - Phi-4-Multimodal: We initially experimented with Microsoft's Phi-4 [18]. Despite resolving initial Flash Attention errors and applying QLoRA quantization, the model's reasoning capabilities on complex charts did not match Qwen2.5.
   - Custom Projectors: We trained a custom architecture using DINOv2 [19] as a vision encoder projected via an MLP into Phi-4-Mini. The projector was trained on ChartQA, PlotQA [20], and Chart2Statista (taking 4-6 hours/epoch). However, this custom adapter failed to generalize as effectively as the integrated Qwen architecture.
3) Data Synthesis: Following literature suggesting data augmentation improves results [21], we incorporated open-source synthesized chart data. However, due to significant distribution shifts between the synthetic data and the realistic ChartQA samples, this actually degraded performance.
4) Advanced Reasoning Structures: We explored Graph of Thought (GoT) [22] which is a non-linear reasoning structure. The graph prompts introduced complexity that made it difficult for a 8.29B model, resulting in worse performance than the linear Chain-of-Thought approach.
5) External Solvers: We integrated a Python-based solver. We experimented with a regular expression triggered version, and another version that is triggered by semantic intent (using embedding cosine similarity). While effective for isolated calculations, it did not improve overall pipeline accuracy, as the TFGR pipeline was often already correct where the solver would have been invoked. Yet, it is worth noting that the solver improved accuracy in our experiments on weaker or non-tabular-guided models.
6) Chart Classifier: We developed a chart classifier using DINOv2 that achieved 99.93% classification accuracy. However, supplying the predicted chart type to the LLM yielded only negligible improvements, since Qwen2.5 already infers chart types implicitly. Incorporating the classifier into the TFGR pipeline also added latency that outweighed the minimal accuracy gains.

E. Error Analysis

Failure case analysis reveals two limitations:

1) Complex Arithmetic: In questions requiring multi-stage compound calculations (e.g., "What is the average of the difference between the top 3 and bottom 3 years?"), the CoT reasoning occasionally drifts.
2) Color Ambiguity: The vision encoder sometimes struggles to distinguish between visually similar colors in legends (e.g., dark blue vs. black), leading to incorrect column identification.

These findings suggest that while Table-First Guarded Reasoning solves the primary issue of data extraction, future work must focus on enhancing the LLM's arithmetic robustness and the vision encoder's color sensitivity.

V. CONCLUSION AND FUTURE WORK

This study challenges the prevailing "scale-is-all-you-need" paradigm in Multimodal Large Language Models. We presented TFGR, a modular pipeline that decouples visual perception from logical reasoning to achieve state-of-the-art results in Chart Question Answering.

By integrating Chart2Table, a robust retrieval-based extraction module, with Zero-Shot Chain-of-Thought prompting, we enabled a 8.29B-parameter model (Qwen2.5-VL-7B-Instruct) to achieve 93.96% accuracy on the ChartQA benchmark. Crucially, our approach outperforms the massive Qwen2-VL-72B [1], demonstrating that architectural innovation can effectively substitute for raw parameter density. Our findings validate that guarding the LLM with structured, extracted data significantly reduces visual hallucinations, making high-precision analytics accessible on consumer-grade hardware.

While our pipeline achieves outstanding accuracy, it introduces trade-offs that invite further investigation.

- Latency Optimization: The current pipeline operates at an average of 9.38 seconds per query, compared to 1.48 seconds for the baseline. Future research can focus on optimizing the Chart2Table API response time and exploring the opportunities of parallel execution and vision encoding to reduce end-to-end latency.
- Visual Sensitivity: Our error analysis revealed that the vision encoder occasionally struggles to distinguish between visually similar colors in legends. Techniques like

"visual cropping" or color-aware pre-processing steps can be investigated to enhance legend-to-data mapping.

- Hybrid Extraction: a meta-classifier can be implemented that dynamically selects between DePlot (generative) and Chart2Table (structural) based on chart complexity and desired inference time, aiming to combine the strengths of both extraction paradigms.

REFERENCES

[1] P. Wang et al., *Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution*, 2024. arXiv: 2409.12191 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2409.12191.

[2] OpenAI et al., *Gpt-4 technical report*, 2024. arXiv: 2303.08774 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2303.08774.

[3] J. Bai et al. "Qwen2.5-vl-7b-instruct: A versatile vision-language model." Hugging Face Model Card, Accessed: Nov. 27, 2025. [Online]. Available: https://huggingface.co/Qwen/Qwen2.5-VL-7B-Instruct.

[4] PaddlePaddle. "Pp-chart2table: Paddlepaddle chart-to-table model." Hugging Face Model Card, Accessed: Nov. 27, 2025. [Online]. Available: https://huggingface.co/PaddlePaddle/PP-Chart2Table.

[5] F. Liu et al., *Deplot: One-shot visual language reasoning by plot-to-table translation*, 2023. arXiv: 2212.10505 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2212.10505.

[6] F. Liu et al., "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)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 12 756–12 770. DOI: 10.18653/v1/2023.acl-long.714. [Online]. Available: https://aclanthology.org/2023.acl-long.714/.

[7] A. Masry, P. Kavehzadeh, X. L. Do, E. Hoque, and S. Joty, *Unichart: A universal vision-language pretrained model for chart comprehension and reasoning*, 2023. arXiv: 2305.14761 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2305.14761.

[8] L. Y.-H. Lo and H. Qu, *How good (or bad) are llms at detecting misleading visualizations?* 2024. arXiv: 2407.17291 [cs.HC]. [Online]. Available: https://arxiv.org/abs/2407.17291.

[9] X. Zeng, H. Lin, Y. Ye, and W. Zeng, *Advancing multimodal large language models in chart question answering with visualization-referenced instruction tuning*, 2024. arXiv: 2407.20174 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2407.20174.

[10] K. Lee et al., *Pix2struct: Screenshot parsing as pretraining for visual language understanding*, 2023. arXiv: 2210.03347 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2210.03347.

[11] Y. Du et al., *Pp-ocr: A practical ultra lightweight ocr system*, 2020. arXiv: 2009.09941 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2009.09941.

[12] J. Wei et al., *Chain-of-thought prompting elicits reasoning in large language models*, 2023. arXiv: 2201.11903 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2201.11903.

[13] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, *Large language models are zero-shot reasoners*, 2023. arXiv: 2205.11916 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2205.11916.

[14] Z. Li, B. Jasani, P. Tang, and S. Ghadar, *Synthesize step-by-step: Tools, templates and llms as data generators for reasoning-based chart vqa*, 2024. arXiv: 2403.16385 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2403.16385.

[15] J. Hong, C. Seto, A. Fan, and R. Maciejewski, *Do llms have visualization literacy? an evaluation on modified visualizations to test generalization in data interpretation*, 2025. arXiv: 2501.16277 [cs.PF]. [Online]. Available: https://arxiv.org/abs/2501.16277.

[16] A. Masry, D. X. Long, J. Q. Tan, S. Joty, and E. Hoque, *Chartqa: A benchmark for question answering about charts with visual and logical reasoning*, 2022. arXiv: 2203.10244 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2203.10244.

[17] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, *Llm.int8(): 8-bit matrix multiplication for transformers at scale*, 2022. arXiv: 2208.07339 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2208.07339.

[18] Microsoft. "Phi-4." Hugging Face Model Card, Accessed: Nov. 27, 2025. [Online]. Available: https://huggingface.co/microsoft/Phi-4.

[19] M. Oquab et al., *Dinov2: Learning robust visual features without supervision*, 2024. arXiv: 2304.07193 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2304.07193.

[20] N. Methani, P. Ganguly, M. M. Khapra, and P. Kumar, *Plotqa: Reasoning over scientific plots*, 2020. arXiv: 1909.00997 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1909.00997.

[21] Y. Yang et al., *Effective training data synthesis for improving mllm chart understanding*, 2025. arXiv: 2508.06492 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2508.06492.

[22] M. Besta et al., "Graph of thoughts: Solving elaborate problems with large language models," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, pp. 17 682–17 690, Mar. 2024, ISSN: 2159-5399. DOI: 10.1609/aaai.v38i16.29720. [Online]. Available: http://dx.doi.org/10.1609/aaai.v38i16.29720.