ChartArena: A Unified Benchmark with Atomic-Primitive Reasoning for Chart Parsing

Shangpin Peng[1, 4, 5, *] **Gengluo Li**[2, *] **Xingyu Wan**[1] **Chengquan Zhang**[1, †]
Hao Feng[1] Binghong Wu[1] Huawen Shen[1] Weinong Wang[1] Ziyi Cai[4]
Zhuotao Tian[4, ✉] Han Hu[1] Can Ma[2] Yu Zhou[3, ✉]

[1]Large Language Model Department, Tencent
[2]Institute of Information Engineering, Chinese Academy of Sciences
[3]Nankai University [4]Shenzhen Loop Area Institute
[5]Hong Kong University of Science and Technology

pspdada0808@gmail.com zhuotaotian@slai.edu.cn yzhou@nankai.edu.cn

Code: https://github.com/pspdada/ChartArena

Abstract

Charts convey dense quantitative and relational information, yet current general chart parsing remains challenging due to fragmented evaluation and limited structural reasoning. Existing benchmarks focus on narrow sub-tasks with inconsistent output formats (e.g., Markdown, CSV, JSON, SVG, code), hindering fair comparison and often overlooking real-world scenarios such as printed or hand-drawn photos. Meanwhile, end-to-end multimodal models tend to learn shallow pixel-to-text mappings without explicit structure understanding, making them fragile under visual variations. To address these issues, we present a unified framework that connects evaluation and modeling. On the evaluation side, we introduce **ChartArena**, a comprehensive bilingual benchmark covering eight chart families, including both numeric charts (bar, line, pie, radar, box plot, combination) and diagrammatic structures (flowchart, mind map). The dataset is constructed via a human-agent collaborative annotation pipeline, where model-assisted drafts are refined through multi-stage human verification to ensure structural consistency and annotation reliability. Furthermore, we design a format-agnostic evaluation protocol that maps different outputs into two canonical semantic spaces: a normalized triple view and a directed graph view, and evaluates them with structure-aware metrics. On the modeling side, we propose **Chart Atomic Primitives (CAP)**, a lightweight semantic scaffold that introduces structural inductive bias during supervised fine-tuning. By guiding the model to infer structure before generating outputs, CAP leads to more consistent and robust parsing behavior. Extensive experiments show that models trained with CAP mechanism achieve reliable performance on ChartArena.

1 Introduction

Charts serve as indispensable visual instruments for conveying quantitative and relational data. To unlock this information, chart parsing [1, 2] aims to convert these images into structured, machine-executable representations, serving as a foundation for downstream data analysis and reasoning. Recently, with the rise of multimodal large language models (MLLMs), the field has shifted from traditional modular pipelines [3, 4] to end-to-end generation approaches [5, 6]. Yet, despite this rapid

*Equal contribution. †Project leader. ✉Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

progress, building a truly general chart parser remains challenging due to two fundamental issues: *fragmented evaluation* and *limited structural reasoning*.

Unlike tasks such as table [7–12] or formula [13–15] parsing, which benefit from unified evaluation standards, chart understanding is hindered by severe fragmentation, presenting a major challenge for standardized assessment. This fragmentation manifests in three distinct ways. First, output formats are strictly siloed (Fig. 1); parsers emit results in disjoint syntactic forms (*e.g.,* Markdown, JSON, CSV), rendering direct cross-model evaluation intractable. Second, existing benchmarks isolate narrow sub-domains, failing to provide a unifying taxonomy across numeric and diagrammatic charts. Third, datasets sample pristine synthetic images, overlooking real-world visual perturbations [16, 17].

Beyond the evaluation bottleneck, current end-to-end parsers [6, 18, 19] also suffer from a critical architectural limitation. Most existing methods treat chart understanding as a black-box mapping from pixels directly to serialized text, relying on shallow sequence memorization instead of explicit structural reasoning. By bypassing intermediate semantics such as axis grounding and relational topology, these models lack a meaningful structural inductive bias and remain fragile under real-world presentation shifts. Taken together, these challenges raise a key question: *How can we unify chart parsing evaluation while equipping models with stronger structural reasoning?*

To tackle these challenges, we present a holistic framework that unifies both the evaluation paradigm and the parsing architecture. For evaluation, we introduce **ChartArena**, the most comprehensive chart parsing benchmark to date (Tab. 1). It spans eight families across numeric charts and diagrammatic structures, bilingual content, and three visual scenarios (digital renderings, printed photos, and hand-drawn photos). ChartArena is built through a human-agent collaborative annotation pipeline, where model-assisted drafts are iteratively refined through multi-stage human verification to ensure structural consistency and annotation reliability. We further design **a format-agnostic evaluation protocol** that deterministically maps heterogeneous model outputs into canonical semantic spaces. This enables direct cross-paradigm comparison with structure-aware metrics. With ChartArena, we evaluate a broad range of chart-specialized models as well as open-source and closed-source general MLLMs, revealing the current capability landscape and key failure modes in chart parsing.

To address the modeling gap, we explore reasoning-oriented architectures by proposing **Chart Atomic Primitives (CAP)**, which abstract the minimal semantic scaffold needed for robust chart comprehension. Integrated into the supervised fine-tuning stage, CAP encourages the model to first construct an explicit structural topology before generating the final sequence. Evaluated on a lightweight 1B-parameter model, our method delivers substantial parsing gains over direct modeling baselines, clearly demonstrating the benefit of explicit structural reasoning.

The primary contributions of this work are summarized as follows:

- **ChartArena benchmark.** We construct a unified benchmark spanning eight numeric and diagrammatic chart families across three visual scenarios. Featuring bilingual content and high-quality annotations, it faithfully captures the readability uncertainty inherent in real-world data.
- **Format-agnostic evaluation protocol.** We design a deterministic protocol that projects heterogeneous model outputs into shared canonical views. Leveraging this protocol, extensive evaluations across leading MLLMs reveal the current capability landscape and systemic vulnerabilities.
- **Reasoning-augmented supervision.** We introduce Chart Atomic Primitives to compel explicit structural reasoning before sequence generation. Evaluated on a 1B model, this approach outperforms direct modeling baselines, validating the fundamental advantage of structural abstraction.

2 Related Work

Evolution of chart parsing. The community initially approached chart parsing as a modular pipeline, combining optical character recognition with heuristic geometry [3, 4]. These cascaded systems suffered from compounding errors and struggled with real-world visual noise [20–23]. The paradigm shifted dramatically with the introduction of MLLMs [5]. Recent literature reformulates chart extraction as an end-to-end sequence generation problem [6, 18]. Despite remarkable progress, current architectures typically map raw pixels directly to final target strings (*e.g.,* Markdown) while neglecting explicit intermediate structural reasoning [1]. This direct mapping forces models to bypass the intermediate cognitive steps that humans instinctively use, such as anchoring axis semantics and localizing atomic data points. Consequently, existing parsers frequently fail to capture complex topological dependencies and lack interpretability, which severely impacts downstream analysis [2, 3].

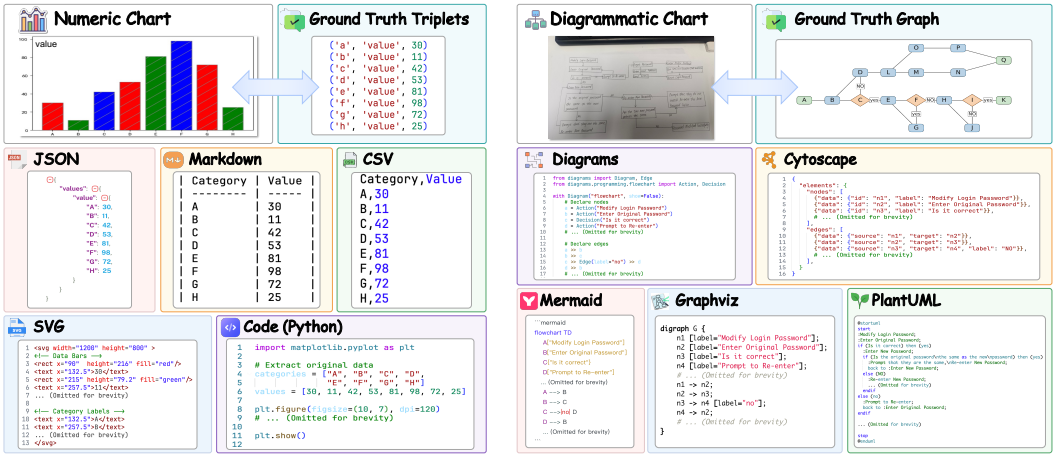


Figure 1: **Heterogeneous output formats.** Existing models parse charts into disparate formats, making direct cross-model evaluation difficult and motivating a unified, format-agnostic protocol.

Output paradigms and evaluation. Despite rapid advances in MLLMs for document understanding, chart parsing remains fragmented by a lack of unified standards. First, representational modalities are highly heterogeneous. Existing methods serialize extracted data from numeric charts into divergent formats, including Markdown [6, 18, 24, 25], SVG [26], Python code [24, 27–29], HTML table [26], CSV [1, 30, 31], or JSON structures [2, 32]. This structural diversity severely hinders cross-model comparisons and complicates the establishment of fair benchmarks. Second, current evaluation frameworks remain limited in scope and consistency. As shown in Tab. 1, recent evaluations typically rely on benchmarks that primarily target numeric charts [18], systematically ignoring diagrammatic structures like flowcharts and mind maps. Consequently, there is a pressing need for a unified evaluation framework that covers diverse chart types and normalizes heterogeneous output representations, thereby enabling systematic progress in the field.

Data synthesis and structural reasoning. The data engines driving recent chart parsers mainly rely on code-to-image synthesis. These pipelines inject semantic data into templating libraries to render millions of image and text pairs [33–35]. While highly successful in scaling data volume, this methodology optimizes for visual variety at the expense of structural depth. It inadvertently trains models to memorize surface layouts rather than comprehend atomic semantic primitives, such as coordinate mappings and legend interactions. In text-centric domains like mathematical reasoning [36–38] and code generation [39], chain-of-thought methodologies [40] have proven highly effective at mitigating shallow memorization by enforcing intermediate logical steps. Chart parsing requires a similar structural inductive bias. Nevertheless, the systematic integration of granular, atomic reasoning steps to explicitly map the topological structure of visual data remains a largely underexplored frontier in chart parsing. Further discussions of related work are in Sec. F.

3 Benchmark and Unified Evaluation

As discussed in Sec. 2, chart parsing remains fragmented in three ways: (i) models emit predictions in heterogeneous output formats, making direct cross-model evaluation difficult; (ii) existing benchmarks focus on isolated sub-domains and rarely unify numeric and diagrammatic charts under a common taxonomy; and (iii) most datasets rely on pristine synthetic renderings without covering real-world visual perturbations such as printed or hand-drawn photographs. To address these issues, we construct a unified benchmark together with a format-agnostic evaluation protocol that maps heterogeneous outputs into shared semantic views, enabling fair comparison across different parsing paradigms.

3.1 Benchmark Construction

Task coverage. Our benchmark is organized along three explicit axes of diversity: (a) *chart family*: eight types spanning both *numeric charts* (bar, line, pie, radar, box plot, and combination chart) and *diagrammatic charts* (flowchart and mind map); (b) *visual scenario*: clean digital renderings as well

Table 1: **Comparison of Chart Parsing Benchmarks.** Our **ChartArena** provides the most comprehensive coverage across chart types, visual scenarios, and languages, enabling realistic evaluation of chart parsing.

Benchmark	Release Date	Size	Chart Types								Image Styles			Languages	
			Bar	Line	Pie	Radar	Box Plot	Comb. Chart	Flow-chart	Mind Map	Digital Rendering	Printed Photo	Hand-drawn Photo	English	Chinese
PlotQA-SE [2, 41]	2019.09	33,657	✓	✓							✓			✓	
ChartQA-SE [2, 42]	2022.03	1,509	✓	✓	✓						✓			✓	
MMC-Bench [43]	2023.11	1,063	✓	✓	✓	✓					✓			✓	
ChartX-SE [1]	2024.02	1,152	✓	✓	✓	✓	✓				✓			✓	
ChartY [2]	2024.04	6,048	✓	✓	✓			✓			✓			✓	✓
VG-DCU [44]	2024.04	3,044	✓	✓	✓		✓	✓			✓			✓	
ChartP-Bench [32]	2026.02	1,200	✓	✓							✓			✓	
ParseBench [45]	2026.04	1,039	✓	✓	✓			✓			✓			✓	
ExChart-Bench [31]	2026.04	3,600	✓	✓	✓	✓					✓			✓	
ChartArena	2026.05	2,400	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

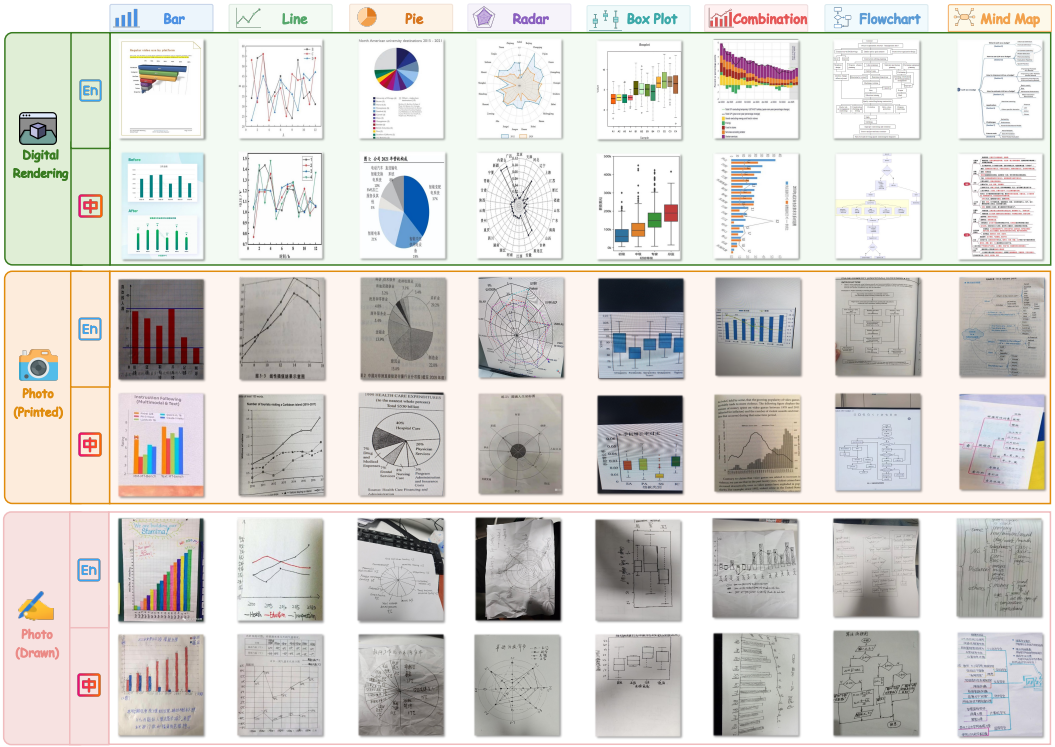


Figure 2: **Benchmark overview.** Our benchmark covers eight chart types spanning both numeric and diagrammatic categories. For each type, we include three visual scenarios (digital rendering, printed photo, and hand-drawn photo) and two languages (English and Chinese), with 50 samples per setting, resulting in a total of 2,400 charts for comprehensive and unified evaluation of chart parsing.

as real-world sources such as printed and hand-drawn photos; (c) *language*: bilingual Chinese and English content. As in Fig. 2, ChartArena explicitly stress-tests chart parsers on the combinations most commonly encountered in real-world applications yet missing from prior benchmarks [41, 42].

Data collection. Following recent studies [11, 46], we curate chart images from public document corpora, web sources, and in-house collections. We explicitly over-sample under-represented scenarios (*e.g.,* printed and hand-drawn charts) to avoid trivializing robustness. Details are shown in Sec. A.1.

Annotation pipeline. To balance efficiency and quality, we adopt a hybrid annotation strategy that combines model-assisted drafting with multi-stage human refinement on a shared human–agent co-annotation platform. For each chart, MLLMs first generate coarse annotations aligned with the chart type (*e.g.,* Markdown for numeric charts and Mermaid for diagrammatic charts), which substantially accelerates annotation. Human annotators then refine both structural elements, such as chart composition, node relations, and axis semantics, and semantic content, including labels and numerical values. When values are difficult to read, multiple annotators cross-check them to improve consistency and reliability. This collaborative pipeline produces high-quality annotations with strong structural consistency across diverse chart types and scenarios. Details are provided in Sec. A.2.

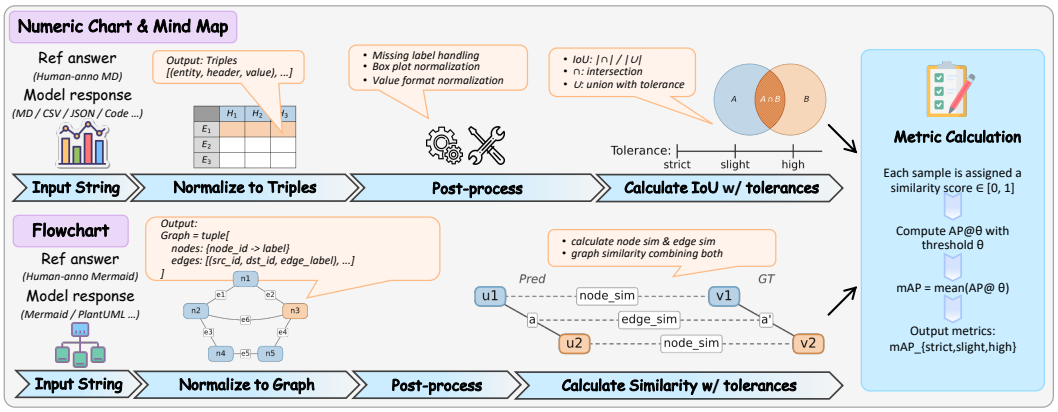


Figure 3: **Evaluation protocol.** We first normalize predictions and references into structured representations (triples for numeric charts and mind maps, and graphs for flowcharts), followed by a format-agnostic post-processing stage. We then compute tolerance-aware similarity (IoU for triples and graph similarity via node and edge matching), and finally aggregate results into unified scores.

3.2 Unified Evaluation Protocol

As discussed in Sec. 2, chart parsing models produce outputs in diverse formats and various diagram languages. These heterogeneous representations hinder direct comparison, as models often differ in *how* results are expressed rather than *what* is extracted. A unified evaluation protocol is therefore essential for fair and consistent assessment.

Format-agnostic normalization. As illustrated in Fig. 3, we first normalize both predictions and references into structured representations. Specifically, numeric charts are mapped into a *triple view* that captures values with associated entities and headers, following recent studies [1, 2]. Flowcharts are converted into a *graph view* with nodes and labeled edges. This design abstracts away surface formats and ensures that semantically equivalent outputs become directly comparable.

Structure-aware metrics with annotation-aware tolerance. For numeric charts, we follow StructChart's SCRM [47], using tolerance-aware triple matching and IoU-based similarity. We retain its three tolerance levels (*strict*, *slight*, *high*) and report Exact Match (EM) and mean Average Precision (mAP) over multiple thresholds. Our protocol adapts this scoring to heterogeneous output formats through canonical normalization and extends coverage to diagrammatic charts through graph and tree matching. For flowcharts, similarity combines node and edge matching. The inherited metric design and our format-specific adaptations are detailed in Sec. D.

4 Data Pipeline

Our data construction framework consists of three components: (i) a semantic-aligned synthesis pipeline that produces chart images with structured annotations (Sec. 4.1), (ii) a CAP extractor that extracts foundational semantics from corpora (Sec. 4.2), and (iii) a reasoning construction module that converts CAP into unified, reasoning-augmented supervision targets (Sec. 4.3).

4.1 Semantic-Aligned Chart Synthesis Pipeline

To endow the model with robust structural priors, we build a large-scale data synthesis pipeline that guarantees explicit semantic grounding across diverse chart families.

Numeric charts. For numeric domains, including bar, line, pie, radar, box plot, and combination charts, we use Qwen3-235B-A22B-Instruct [48] to synthesize both semantic corpora and chart template libraries. The semantic module generates structured key–value configurations, such as category names, series definitions, and numerical fields with coherent internal logic. The template library covers diverse visual styles. Each template fixes both the rendering recipe and the Markdown serialization format. By pairing semantic instances with templates and geometric layouts, we obtain million-scale image–annotation pairs for each chart family.

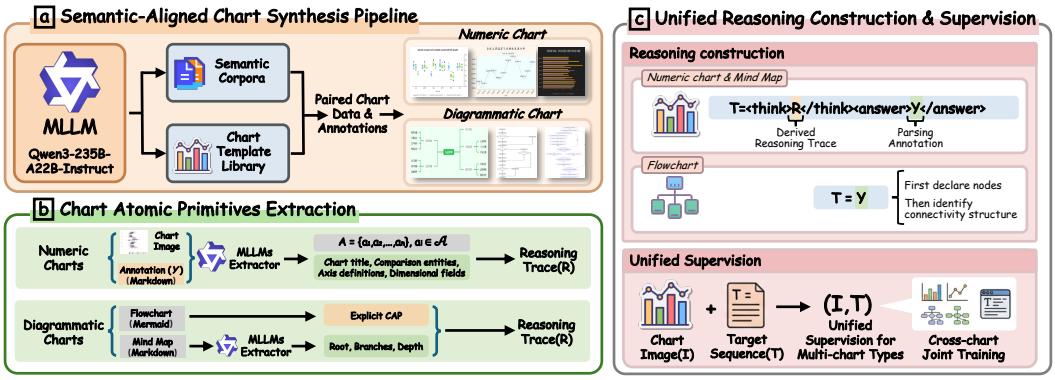


Figure 4: **Overview of data pipeline.** (a) LLM-based synthesis stage generates chart images I and annotations Y. (b) CAP extraction stage derives atomic semantics A from (I, Y). (c) Unified reasoning stage constructs targets T, enabling all chart types to share a single (I, T) training interface.

Diagrammatic charts. For diagrammatic domains (flowcharts and mind maps), the same LLM directly generates the structured target. For flowcharts, it emits executable Mermaid programs that explicitly encode node semantics and edge relations, which are then rendered into images; for mind maps, it produces hierarchical Markdown that captures the content organization. This guarantees that each chart image faithfully mirrors the underlying topology and the parsing target.

4.2 Chart Atomic Primitives

To bridge the semantic gap between raw pixels and serialized output, we introduce **Chart Atomic Primitives (CAP)** as a compact intermediate representation that isolates the minimal semantic scaffold needed for chart comprehension. Formally, given a chart image I and its target annotation Y, the corresponding CAP is defined as:

$$A = \{a_1, a_2, \ldots, a_n\}, \quad a_i \in \mathcal{A}, \tag{1}$$

where each primitive a_i denotes a distinct, chart-level semantic component. By decoupling instance-level specificities from the underlying topology, CAP extracts structural semantics such as chart titles, comparison entities, axis definitions, dimensional fields, and core statistical attributes.

Numeric charts. For numeric charts, CAP is materialized through schemas specifically tailored to the chart type. For instance, two-dimensional Cartesian charts inherently encode axis semantics and tick ranges, radar charts encapsulate dimensional nodes and radial axes, whereas combination charts capture intricate multi-axis alignments alongside dense statistical fields. Details are in Sec. B.2.

Diagrammatic charts. For diagrammatic charts, CAP aligns naturally with structured representations. In flowchart-like diagrams, Mermaid supports two common serialization styles: one that interleaves node definitions with edge relations, and another that separates node declarations from connectivity structure. To ensure consistency, we normalize all predictions into a unified form that explicitly separates nodes and edges. This normalized representation directly corresponds to atomic components and their interactions, and is used as the basis for CAP reasoning. For mind maps, where structure is often implicit, we further abstract high-level primitives such as the root topic, primary branches, hierarchical depth, and semantic domains, enabling consistent structural reasoning across formats.

4.3 CAP Extraction and Reasoning Construction

To convert CAP into supervision signals, we first extract atomic primitives A from (I, Y) for numeric charts and mind maps by prompting Qwen3-VL-235B-A22B-Instruct [49], and then serialize them into a structured reasoning trace. The extracted primitives focus on chart-level semantics such as titles, entities, axes, dimensions, and key statistics, while discarding low-level visual details. The reasoning trace is constructed as:

$$R = \mathcal{G}(A), \tag{2}$$

where $\mathcal{G}(\cdot)$ acts as a templating function that serializes the isolated atomic fields into an ordered sequence of semantic analysis steps. Ultimately, the unified supervision target is formulated as:

$$T = \texttt{<think>}\ R\ \texttt{</think>}\ \texttt{<answer>}\ Y\ \texttt{</answer>}\,. \tag{3}$$

This architectural decision injects a profound structural inductive bias into the training loop, forcing the model to articulate a topological scaffold before committing to a final parse.

For flowchart-like diagrams, we define $T = Y$ after normalizing the Mermaid representation into a canonical form that explicitly separates node definitions from edge relations. Only this structure-first representation aligns with CAP, as it exposes atomic components before their interactions.

Although the construction differs by chart type, all samples share the same (I, T) interface, enabling joint optimization across heterogeneous chart types within a single training pipeline.

5 Training Pipeline

Our training pipeline follows two stages: pretraining learns direct parsing ability, and supervised fine-tuning (SFT) introduces CAP reasoning while adapting the model to broader visual diversity.

5.1 Pretraining

A naive strategy that trains directly on reasoning-augmented targets forces the model to learn extraction and reasoning simultaneously, which may create competing supervision signals and hurt optimization efficiency and generalization [50–52]. We therefore use pretraining only for direct parsing supervision, so that the model learns the basic mapping from chart images to structured outputs and develops a stable initialization for later reasoning-based adaptation.

5.2 Supervised Fine-Tuning with CAP Reasoning

Building on this initialization, SFT introduces CAP reasoning. We augment the training targets with CAP-formatted supervision, encouraging the model to infer structure before generating the final output, thereby improving structural consistency during training.

At the same time, SFT broadens the data manifold along three axes: source heterogeneity, stylistic diversity, and semantic difficulty. We mix public chart understanding datasets with our human-annotated samples (Sec. 3) to cover more chart types, layout patterns, and semantic topics. We also apply rendering perturbations such as font variation and geometric warping to improve robustness against presentation shifts. As a result, the model retains the structural priors learned during pretraining while becoming more reliable on rendered charts, printed photos, hand-drawn photos, and bilingual inputs.

6 Experiments

We organize our experiments on **ChartArena** around four aspects: (i) experimental settings, including backbone, training recipe and baselines (Sec. 6.1); (ii) main comparison against expert chart parsers, general-purpose MLLMs, and document parsing MLLMs (Sec. 6.2); (iii) the adaptability of our unified evaluation protocol to diverse output formats (Sec. 6.3); and (iv) ablations on the training pipeline that isolate the effect of pretraining, SFT, and CAP reasoning (Sec. 6.4). Further analysis under different visual scenarios is shown in Sec. E.1.

6.1 Experimental Settings

Backbone and training recipe. We mainly adopt HunyuanOCR [6] as the vision–language backbone. Full-parameter fine-tuning is used throughout, and all experiments are conducted on 16 NVIDIA H20 GPUs. For *pretraining*, we combine 2M synthesized chart–target pairs from our pipeline with 2M samples from ChartGalaxy [33] and ChartNet [34], and train for 1 epoch with a learning rate of 4×10^{-5}. For *SFT*, we use the 60K human-annotated examples released by ChartNet together with our additional 20K in-house annotations, and train for 1 epoch with a learning rate of 2×10^{-5}. Detailed training settings are provided in Sec. B.1.

Baselines. We compare our method against representative baselines across three categories: (a) *General-purpose MLLMs*, capable of chart parsing through unified multimodal modeling; (b) *Document parsing MLLMs*, optimized for holistic document parsing; and (c) *Expert chart understanding*

Table 2: **Main results on ChartArena.** We report detailed performance for each chart type as well as the overall average. Each type includes two scores: **EN** (English) and **ZH** (Chinese), both averaged over three visual styles (digital renderings, printed photos, and hand-drawn photos). The **Average** columns are all-task means over all eight chart families. For each metric, we report $\mathbf{mAP_{high}}$. Within each model category, **bold** and underline denote the best and second-best results within each chart type.

Model Type	Model	Release Date	bar		line		pie		radar		box plot		comb. chart		flowchart		mind map		Average	
			EN	ZH	EN	ZH	EN	ZH	EN	ZH	EN	ZH	EN	ZH	EN	ZH	EN	ZH	EN	ZH
General Purpose MLLMs	GPT-4o [53]	2024.05	21.6	36.3	27.5	52.9	76.7	74.2	9.7	24.9	19.1	9.6	9.9	40.7	49.8	27.1	64.0	24.8	34.8	36.3
	GPT-5 [54]	2025.08	35.1	52.3	48.1	65.1	81.1	78.9	**32.0**	41.5	19.8	12.8	14.2	46.5	58.1	35.3	76.6	33.5	45.6	45.8
	InternVL3.5-8B [55]	2025.08	22.7	52.6	34.4	53.7	65.8	73.8	14.0	34.7	5.6	9.5	11.3	42.1	32.6	23.8	48.3	31.8	29.3	40.2
	InternVL3.5-241B-A28B [55]	2025.08	27.5	57.2	41.3	55.7	77.7	83.3	15.2	41.4	18.7	21.6	17.7	47.8	43.8	36.6	62.6	45.5	38.0	48.6
	Qwen2.5-VL-7B-Ins. [5]	2025.01	15.2	36.9	17.9	39.9	63.4	73.1	8.3	19.1	0.9	2.8	6.0	40.6	29.7	23.2	45.4	29.9	23.3	33.2
	Qwen2.5-VL-72B-Ins. [5]	2025.01	27.1	53.3	38.2	66.7	73.5	77.0	10.9	38.5	15.0	15.3	14.3	50.5	50.1	43.6	63.8	55.0	36.6	50.0
	Qwen3-VL-8B-Ins. [49]	2025.10	27.5	58.6	35.5	61.1	77.3	84.7	16.8	42.6	11.6	12.1	13.2	47.9	50.0	41.5	66.4	54.6	37.3	50.4
	Qwen3-VL-235B-A22B-Ins. [49]	2025.09	38.4	67.9	52.3	73.8	82.6	85.5	23.2	52.4	14.1	14.1	29.1	58.2	57.9	49.8	70.8	65.2	46.0	58.4
	Qwen3.5-35B-A3B (think) [56]	2026.02	46.2	65.3	60.3	77.6	89.7	88.4	25.2	57.8	42.2	50.6	31.5	56.9	62.5	56.5	75.1	70.9	54.1	65.5
	GLM-4.5V [57]	2025.08	33.5	61.4	51.7	70.5	81.2	83.1	19.7	43.1	32.4	37.4	21.2	52.5	44.7	39.6	66.2	43.7	43.8	53.9
	Seed1.8 (no-think) [58]	2025.12	29.1	59.7	46.0	72.5	84.7	88.0	22.0	45.9	16.1	17.5	15.0	59.7	47.8	50.3	76.5	69.1	42.2	57.8
	Seed2.0 Pro (no-think) [59]	2026.02	40.3	73.3	56.5	80.7	91.5	90.5	21.3	54.7	**44.5**	**55.2**	32.4	62.2	62.6	61.3	83.1	**85.8**	54.0	70.5
	Kimi K2.5 (no-think) [60]	2026.01	45.2	70.3	60.9	79.8	87.2	86.7	30.2	59.7	40.6	47.6	33.6	63.6	59.9	57.9	80.8	79.4	54.8	68.1
	MiMo-V2-Omni [61]	2026.03	31.1	56.9	41.5	66.4	87.0	85.8	19.7	46.1	19.1	30.3	19.4	54.7	57.1	51.0	76.6	64.6	43.9	57.0
	Gemini 2.5 Pro [62]	2025.03	46.0	76.5	56.5	77.6	88.6	87.3	17.5	53.0	10.2	22.1	28.7	57.6	62.1	57.8	71.7	67.1	47.7	62.4
	Gemini 3.1 Pro [63]	2026.02	**57.9**	**78.7**	**67.0**	**85.3**	**92.5**	**95.1**	31.8	**62.7**	32.5	45.2	**39.7**	**70.3**	**65.6**	**63.1**	**86.8**	85.2	**59.2**	**73.2**
Document Parsing MLLMs	dots.mocr (3B) [26]	2026.03	28.3	40.9	41.8	60.1	68.8	**78.3**	**20.3**	**43.1**	24.1	16.0	**26.9**	47.1	26.2	20.6	28.7	19.6	33.1	40.7
	PaddleOCR-VL (1B) [18]	2025.10	31.8	49.3	43.0	51.6	57.5	75.2	14.4	29.0	11.7	20.7	21.3	**54.0**	–	–	–	–	22.5	35.0
	HunyuanOCR (1B) [6]	2025.11	**32.4**	**58.9**	**48.6**	**68.0**	**70.8**	72.5	19.0	40.9	**43.9**	**43.9**	20.1	48.3	**38.3**	**35.5**	**55.0**	**46.6**	**41.0**	**51.8**
Expert Chart Understanding Models	ChartAst (13B) [25]	2024.01	5.2	–	4.2	–	0.3	–	1.5	–	0.3	–	0.0	–	–	–	–	–	1.4	–
	ChartVLM (8.3B) [1]	2024.02	11.2	5.3	11.5	4.3	12.9	8.2	2.1	5.0	0.7	0.4	4.1	4.4	–	–	–	–	5.3	3.5
	TinyChart (3B) [24]	2024.04	6.1	6.3	9.7	3.2	5.7	5.4	0.5	3.4	0.2	1.3	0.7	4.2	–	–	–	–	2.9	3.0
	ChartMoE (8B) [30]	2024.09	18.7	24.4	14.7	22.3	15.0	48.5	3.7	16.1	2.7	1.6	5.1	19.5	4.0	–	4.1	–	8.5	16.6
	ChartCoder (7B) [29]	2025.01	23.2	12.6	22.0	19.6	34.3	16.7	5.5	13.9	5.4	11.4	3.7	5.1	5.6	–	1.0	–	12.6	9.9
	RRVF (7B) [28]	2025.07	35.8	**66.5**	41.5	54.3	51.6	75.3	16.6	40.3	14.7	14.1	23.5	**61.2**	36.4	32.4	**68.4**	**63.8**	36.0	51.0
	MSRL (7B) [27]	2025.08	32.7	45.2	35.2	34.3	41.2	67.9	25.9	48.0	11.2	13.0	16.7	35.2	23.2	12.4	31.0	18.8	27.1	34.3
	Ours (1B)	2026.05	**43.2**	63.8	**57.8**	**73.5**	**74.3**	**82.0**	**28.1**	**54.7**	**54.3**	**59.1**	**26.3**	55.1	**50.8**	**51.3**	61.4	50.9	**49.5**	**61.3**

models, dedicated parsers explicitly designed for chart structure recognition and parsing. For each category, we select strong and representative models to ensure a comprehensive evaluation. Each baseline is run in its native output format, and outputs are normalized before scoring under the unified protocol of Sec. 3.2. Slices outside the declared capability of a model are reported as "–", yielding an per-model all-task *Average* columns over all eight chart families.

6.2 Main Results

Tab. 2 summarizes the main comparison on **ChartArena**, where we highlight two observations.

Best performance among expert chart understanding models. Our parser supports the broadest range of chart types and language settings while remaining the most lightweight model (only 1B parameters) in this category. Existing expert parsers are typically limited to a few numeric chart families or English-only data. Despite using 7–13 times fewer parameters, our model achieves the highest average score (49.5 EN / 61.3 ZH) and outperforms prior expert models on most chart families, including structurally complex cases such as box plots, radar charts, and flowcharts.

Competitive against general-purpose MLLMs. Compared with general-purpose MLLMs, our 1B parser already outperforms strong 8B baselines, including InternVL3.5-8B [55] and Qwen3-VL-8B-Instruct [49] on the overall average, and is competitive with much larger open-source systems on most chart families. A noticeable gap remains compared to frontier-scale models such as the open-source Kimi K2.5 [60] and the proprietary Gemini 3.1 Pro [63]. These models benefit from larger parameter scales and broader world knowledge. The advantage is most evident on the hardest ChartArena slices, which require inferring implicit semantics, units, or domain conventions from context. This gap also varies by chart type. General-purpose MLLMs struggle on box plots, where richer visual cues must be identified. In contrast, our 1B parser lags on mind maps, which require broader world knowledge. Both models perform poorly on radar charts, highlighting the difficulty of visual estimation.

6.3 Adaptability to Diverse Output Formats

A central design choice of ChartArena is that the proposed unified evaluation protocol (Sec. 3.2) is format-agnostic, outputs are normalized into canonical triples/graph views before scoring, and we can evaluate the same model under any of its supported output paradigms. We use two representative models, the proprietary Seed2.0 Pro [59] and the open-source Qwen3.5-35B-A3B [56], and report their performance under five numeric-chart formats and five flowchart formats in Tab. 3.

Table 3: **Adaptability to diverse output formats.** Our evaluation framework assesses model outputs across multiple structured formats. The left reports results on **numeric charts**, while the right reports **flowcharts**.

Model	Numeric Charts					Flowcharts				
	Output Format	EM	mAP_{strict}	mAP_{slight}	mAP_{high}	Output Format	EM	mAP_{strict}	mAP_{slight}	mAP_{high}
Seed2.0 Pro (no-think)	Markdown	16.3	22.0	38.7	54.9	Mermaid	4.0	32.4	51.1	58.3
	JSON	17.4	23.2	40.7	59.1	Cytoscape	5.3	35.7	55.5	62.0
	CSV	14.4	20.0	37.2	55.0	Diagrams	4.0	30.4	52.5	59.8
	Code	17.0	22.2	37.3	53.9	Graphviz	5.0	33.9	54.5	61.7
	SVG	8.0	14.0	28.0	40.0	PlantUML	1.0	12.9	24.3	33.8
Qwen3.5 35B-A3B (think)	Markdown	15.8	21.4	37.0	56.0	Mermaid	3.7	28.1	48.2	57.0
	JSON	5.9	9.4	23.0	46.9	Cytoscape	4.7	31.0	50.6	59.5
	CSV	15.0	20.0	38.5	56.6	Diagrams	3.7	28.9	49.0	57.1
	Code	14.3	19.3	32.2	46.2	Graphviz	5.0	28.3	48.0	57.0
	SVG	6.3	13.4	27.4	39.9	PlantUML	0.3	9.4	16.5	29.0

For numeric charts, Seed2.0 Pro is relatively stable across Markdown, JSON, CSV, and Code. Qwen3.5-35B-A3B drops from 56.0 mAP_{high} in Markdown to 46.9 in JSON and 46.2 in Code. Format-agnostic scoring therefore does not imply format-invariant model performance. SVG is weakest for both, suggesting difficulty recovering exact values from visual representations.

For flowcharts, Mermaid, Cytoscape, Diagrams, and Graphviz are all competent and roughly comparable, whereas PlantUML performs significantly worse: its code-like control statements struggle to encode complex graph topologies (e.g., cycles, multi-source/multi-sink subgraphs), which are frequent in the flowchart slice of ChartArena.

6.4 Ablation Study

Tab. 4 presents ablation results for pretraining, SFT, and CAP reasoning, further grouped by visual style. Pretraining (row (a)) mainly improves digital-rendering samples, with limited gains on printed and hand-drawn charts, reflecting the synthetic nature of the pretraining corpus. SFT (row (b)) substantially improves robustness on printed and hand-drawn samples for both numeric and diagrammatic charts. Adding CAP reasoning to either numeric or diagrammatic alone (rows (c)–(d)) consistently improves the corresponding subset without degrading the other. Enabling CAP on both subsets (row (e)) achieves the best overall performance and is used in Tab. 2.

Table 4: **Ablation study on training strategies.** We analyze the impact of pretraining and different SFT settings with or without CAP reasoning on numeric charts (1,800 samples) and diagrammatic charts (600 samples).

Setting	Pretrain	SFT		Numeric Charts			Diagrammatic Charts		
		Numeric	Flow	Digital	Printed	Drawn	Digital	Printed	Drawn
MLLM Backbone	–	–	–	56.3	49.7	37.4	55.0	50.4	26.0
(a) Pretrain	✓	–	–	62.5	50.2	38.1	58.9	51.8	27.7
(b) + SFT (w/o CAP Reasoning)	✓	direct	direct	63.1	57.8	44.9	59.6	55.8	37.9
(c) + SFT (Numeric w/ CAP Reasoning, Diagrammatic w/o)	✓	CAP	direct	64.2	60.8	47.8	60.1	56.3	38.5
(d) + SFT (Numeric w/o CAP Reasoning, Diagrammatic w)	✓	direct	CAP	62.9	58.2	44.3	62.3	57.9	41.2
(e) + SFT (w CAP Reasoning)	✓	CAP	CAP	64.1	60.4	48.6	62.5	58.1	40.9

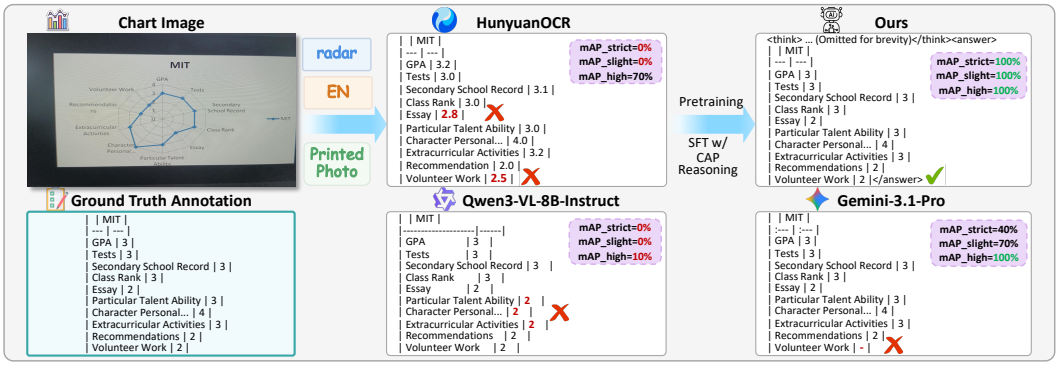


Figure 5: **Qualitative Comparisons on ChartArena.** Photograph samples are challenging due to visual noise. CAP training enables more accurate and complete recovery of structure and values.

6.5 Qualitative Result

As shown in Fig. 5, photograph-based charts are challenging due to visual distortions. After CAP training, our model recovers all values correctly and achieves the best scores. Gemini performs relatively well overall, but replaces uncertain entries with "–", leading to incomplete predictions. Further case studies are provided in Sec. I.

7 Conclusion

We introduced ChartArena, a unified benchmark and format-agnostic evaluation framework for general chart parsing across eight numeric and diagrammatic chart families. By covering bilingual content, real-world visual scenarios, and heterogeneous output formats, ChartArena reveals the limitations of existing parsers under realistic structural and visual variation. We further proposed CAP-based reasoning, which injects explicit structural priors into chart parsers and consistently improves robustness over direct generation baselines. Together, ChartArena and CAP provide a unified foundation for evaluating and building robust general-purpose chart parsers.

Limitations. Despite the broad coverage of ChartArena, two limitations remain. First, the current benchmark focuses on single-page chart images and does not yet cover multi-page charts, where the parser must aggregate visual elements, legends, or continuation tables across pages. Second, ChartArena does not include certain chart families that are particularly challenging for chart parsing models, such as scatter plots. These charts often require precise point localization and dense coordinate recovery, which introduce difficulties beyond the structural extraction tasks considered in this work. We leave these directions to future extensions of the benchmark.

References

[1] Renqiu Xia, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Botian Shi, Junchi Yan, and Bo Zhang. ChartX and ChartVLM: A versatile benchmark and foundation model for complicated chart reasoning. *IEEE Transactions on Image Processing*, 2025.

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

[3] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. ChartSense: Interactive data extraction from chart images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 2017.

[4] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. ReVision: Automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011.

[5] Shuai Bai, Keqin Chen, Xuejing Liu, et al. Qwen2.5-VL Technical Report. *arXiv preprint arXiv:2502.13923*, 2025.

[6] Hunyuan Vision Team, Pengyuan Lyu, Xingyu Wan, Gengluo Li, Shangpin Peng, Weinong Wang, Liang Wu, Huawen Shen, Yu Zhou, Canhui Tang, et al. HunyuanOCR Technical Report. *arXiv preprint arXiv:2511.19575*, 2025.

[7] Huawen Shen, Xiang Gao, Jin Wei, Liang Qiao, Yu Zhou, Qiang Li, and Zhanzhan Cheng. Divide Rows and Conquer Cells: Towards structure recognition for large tables. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 1369–1377, 2023.

[8] Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. Global Table Extractor (GTE): A framework for joint table identification and cell structure recognition using visual context. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 697–706, 2021.

[9] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. Image-Based Table Recognition: Data, model, and evaluation. In *Proceedings of the European Conference on Computer Vision*, pages 564–580, 2020.

[10] Gengluo Li, Shangpin Peng, Chengquan Zhang, Binghong Wu, Hao Feng, Weinong Wang, Pengyuan Lyu, Huawen Shen, Xingyu Wan, Zhuotao Tian, et al. StrucTab: A structured optimization framework for table parsing. In *Proceedings of the European Conference on Computer Vision*, pages 672–690, 2026.

[11] Zhibo Yang, Jun Tang, Zhaohai Li, Pengfei Wang, Jianqiang Wan, Humen Zhong, Xuejing Liu, Mingkun Yang, Peng Wang, Shuai Bai, et al. CC-OCR: A comprehensive and challenging OCR benchmark for evaluating large multimodal models in literacy. In *Proceedings of the IEEE International Conference on Computer Vision*, 2025.

[12] Weichao Zhao, Hao Feng, Qi Liu, Jingqun Tang, Wei Shu, Binghong Wu, Lei Liao, Yongjie Ye, Hao Liu, Wengang Zhou, et al. TabPedia: Towards comprehensive visual table understanding with concept synergy. In *Advances in Neural Information Processing Systems*, volume 37, pages 7185–7212, 2024.

[13] Bin Wang, Fan Wu, Linke Ouyang, Zhuangcheng Gu, Rui Zhang, Renqiu Xia, Botian Shi, Bo Zhang, and Conghui He. Image Over Text: Transforming formula recognition evaluation with Character Detection Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2025.

[14] Ye Yuan, Xiao Liu, Wondimu Dikubab, Hui Liu, Zhilong Ji, Zhongqin Wu, and Xiang Bai. Syntax-Aware Network for Handwritten Mathematical Expression Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[15] Bin Wang, Zhuangcheng Gu, Guang Liang, Chao Xu, Bo Zhang, Botian Shi, and Conghui He. UniMER-Net: A Universal Network for Real-World Mathematical Expression Recognition. *arXiv preprint arXiv:2404.15254*, 2024.

[16] An-Lan Wang, Jingqun Tang, Lei Liao, Hao Feng, Qi Liu, Xiang Fei, Jinghui Lu, Han Wang, Hao Liu, Yuliang Liu, et al. WildDoc: How far are we from achieving comprehensive and robust document understanding in the wild? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 23002–23012, 2025.

[17] Gengluo Li, Pengyuan Lyu, Chengquan Zhang, Huawen Shen, Liang Wu, Xingyu Wan, Gangyan Zeng, Han Hu, Can Ma, and Yu Zhou. Towards real-world document parsing via realistic scene synthesis and document-aware training. *arXiv preprint arXiv:2603.23885*, 2026.

[18] Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, et al. PaddleOCR-VL: Boosting multilingual document parsing via a 0.9B ultra-compact vision-language model. *arXiv preprint arXiv:2510.14528*, 2025.

[19] Hao Feng, Shu Wei, Xiang Fei, Wei Shi, Yingdong Han, Lei Liao, Jinghui Lu, Binghong Wu, Qi Liu, Chunhui Lin, et al. Dolphin: Document image parsing via heterogeneous anchor prompting. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21919–21936, 2025.

[20] Rujiao Long, Wen Wang, Nan Xue, Feiyu Gao, Zhibo Yang, Yongpan Wang, and Gui-Song Xia. Parsing table structures in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021.

[21] Saleem Ahmed, Bhavin Jawade, Shubham Pandey, Srirangaraj Setlur, and Venu Govindaraju. RealCQA: Scientific chart question answering as a test-bed for first-order logic. In *International Conference on Document Analysis and Recognition*. Springer, 2023.

[22] Muye Huang, Han Lai, Xinyu Zhang, Wenjun Wu, Jie Ma, Lingling Zhang, and Jun Liu. EvoChart: A benchmark and a self-training approach towards real-world chart understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[23] Maeve Hutchinson, Radu Jianu, Aidan Slingsby, Jo Wood, and Pranava Swaroop Madhyastha. Chart question answering from real-world analytical narratives. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, 2025.

[24] Liang Zhang, Anwen Hu, Haiyang Xu, Ming Yan, Yichen Xu, Qin Jin, Ji Zhang, and Fei Huang. TinyChart: Efficient chart understanding with Program-of-Thoughts learning and visual token merging. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1882–1898, 2024.

[25] Fanqing Meng, Wenqi Shao, Quanfeng Lu, Peng Gao, Kaipeng Zhang, Yu Qiao, and Ping Luo. ChartAssistant: 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*, 2024.

[26] Handong Zheng, Yumeng Li, Kaile Zhang, Liang Xin, Guangwei Zhao, Hao Liu, Jiayu Chen, Jie Lou, Jiyu Qiu, Qi Fu, et al. Multimodal OCR: Parse anything from documents. *arXiv preprint arXiv:2603.13032*, 2026.

[27] Lei Chen, Xuanle Zhao, Zhixiong Zeng, Jing Huang, Liming Zheng, Yufeng Zhong, and Lin Ma. Breaking the SFT plateau: Multimodal structured reinforcement learning for Chart-to-Code generation. *arXiv preprint arXiv:2508.13587*, 2025.

[28] Yang Chen, Yufan Shen, Wenxuan Huang, Sheng Zhou, Qunshu Lin, Xinyu Cai, Zhi Yu, Jiajun Bu, Botian Shi, and Yu Qiao. Learning Only with Images: Visual reinforcement learning with reasoning, rendering, and visual feedback. *arXiv preprint arXiv:2507.20766*, 2025.

[29] Xuanle Zhao, Xianzhen Luo, Qi Shi, Chi Chen, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ChartCoder: Advancing multimodal large language model for Chart-to-Code generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

[30] Zhengzhuo Xu, Bowen Qu, Yiyan Qi, Sinan Du, Chengjin Xu, Chun Yuan, and Jian Guo. ChartMoE: Mixture of diversely aligned expert connector for chart understanding. In *International Conference on Learning Representations*, 2025.

[31] Yuchen He, Peizhi Ying, Liqi Cheng, Kuilin Peng, Yuan Tian, Dazhen Deng, and Yingcai Wu. Making multimodal LLMs reliable chart data extractors: A benchmark and training framework. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, 2026.

[32] Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. Visual Self-Refine: A pixel-guided paradigm for accurate chart parsing. *arXiv preprint arXiv:2602.16455*, 2026.

[33] Zhen Li, Duan Li, Yukai Guo, Xinyuan Guo, Bowen Li, Lanxi Xiao, Shenyu Qiao, Jiashu Chen, Zijian Wu, Hui Zhang, et al. ChartGalaxy: A dataset for infographic chart understanding and generation. *arXiv preprint arXiv:2505.18668*, 2025.

[34] Jovana Kondic, Pengyuan Li, Dhiraj Joshi, Isaac Sanchez, Ben Wiesel, Shafiq Abedin, Amit Alfassy, Eli Schwartz, Daniel Caraballo, Yagmur Gizem Cinar, Florian Scheidegger, Steven I. Ross, Daniel Karl I. Weidele, Hang Hua, Ekaterina Arutyunova, Roei Herzig, Zihan Wang, Xinyue Yu, Yunfei Zhao, Sicong Jiang, Minghao Liu, Qunshu Lin, Aude Oliva, and Rogerio Feris. ChartNet: A million-scale, high-quality multimodal dataset for robust chart understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 15922–15932, 2026.

[35] Mengchen Liu, Qixiu Li, Dongdong Chen, Dong Chen, Jianmin Bao, and Yunsheng Li. SynChart: Synthesizing charts from language models. *arXiv preprint arXiv:2409.16517*, 2024.

[36] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[37] Shangpin Peng, Weinong Wang, Zhuotao Tian, Senqiao Yang, Xing Wu, Haotian Xu, Chengquan Zhang, Takashi Isobe, Baotian Hu, and Min Zhang. Uni-DPO: A unified paradigm for dynamic preference optimization of LLMs. *arXiv preprint arXiv:2506.10054*, 2025.

[38] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

[39] Jia Li, Ge Li, Yongmin Li, and Zhi Jin. Structured Chain-of-Thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology*, 2025.

[40] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-Thought prompting elicits reasoning in large language models. In *Proceedings of Advances in Neural Information Processing Systems*, pages 24824–24837, 2022.

[41] Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. PlotQA: Reasoning over scientific plots. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 1516–1525, 2020.

[42] Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2022.

[43] 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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024.

[44] Shuguang Dou, Xinyang Jiang, Lu Liu, Lu Ying, Caihua Shan, Yifei Shen, Xuanyi Dong, Yun Wang, Dongsheng Li, and Cairong Zhao. Hierarchically recognizing vector graphics and a new chart-based vector graphics dataset. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[45] Boyang Zhang, Sebastián G Acosta, Preston Carlson, Sacha Bron, Pierre-Loïc Doulcet, Daniel B Ospina, and Simon Suo. ParseBench: A document parsing benchmark for AI agents. *arXiv preprint arXiv:2604.08538*, 2026.

[46] Michał Turski, Tomasz Stanisławek, Karol Kaczmarek, Paweł Dyda, and Filip Graliński. CCpdf: Building a high quality corpus for visually rich documents from web crawl data. In *International Conference on Document Analysis and Recognition*, pages 348–365, 2023.

[47] Renqiu Xia, Haoyang Peng, Hancheng Ye, Mingsheng Li, Xiangchao Yan, Peng Ye, Botian Shi, Yu Qiao, Junchi Yan, and Bo Zhang. StructChart: On the schema, metric, and augmentation for visual chart understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

[48] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

[49] Shuai Bai, Yuxuan Cai, et al. Qwen3-VL Technical Report. *arXiv preprint arXiv:2511.21631*, 2025.

[50] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the International Conference on Machine Learning*, 2009.

[51] Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. Small models struggle to learn from strong reasoners. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

[52] Ahmet Ezgi and Aytuğ Onan. Task-structured curriculum learning for multi-task distillation: Enhancing step-by-step knowledge transfer in language models. *Computers, Materials & Continua*, 2026.

[53] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o System Card. *arXiv preprint arXiv:2410.21276*, 2024.

[54] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. OpenAI GPT-5 System Card. *arXiv preprint arXiv:2601.03267*, 2025.

[55] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

[56] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

[57] Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, et al. GLM-4.5V and GLM-4.1V-Thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*, 2025.

[58] Bytedance Seed. Seed1.8 Model Card: Towards generalized real-world agency. *arXiv preprint arXiv:2603.20633*, 2026.

[59] Bytedance Seed. Seed2.0 Model Card: Towards intelligence frontier for real-world complexity. *arXiv preprint arXiv:2607.00248*, 2026.

[60] Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

[61] Xiaomi Corporation. Xiaomi MiMo-V2-Omni: See, hear, act in the agentic era. `https://mimo.xiaomi.com/mimo-v2-omni`, 2026.

[62] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[63] Google. Gemini 3.1 Pro: A smarter model for your most complex tasks. `https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/`, 2026.

[64] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics. Doklady*, 1966.

[65] Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 Technical Report. *arXiv preprint arXiv:2607.02770*, 2026.

[66] Tianyu Yu, Zefan Wang, Chongyi Wang, Fuwei Huang, Wenshuo Ma, Zhihui He, Tianchi Cai, Weize Chen, Yuxiang Huang, Ranchi Zhao, et al. MiniCPM-V 4.5: Cooking efficient MLLMs via architecture, data, and training recipe. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11704–11715, 2026.

[67] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and PixMo: Open weights and open data for state-of-the-art vision-language models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2025.

[68] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Proceedings of Advances in Neural Information Processing Systems*, 2020.

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

[70] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

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

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

[73] Anthropic. The Claude 3 model family: Opus, Sonnet, Haiku, 2024. URL `https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf`.

[74] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 Technical Report. *arXiv preprint arXiv:2412.19437*, 2024.

[75] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[76] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct Preference Optimization: Your language model is secretly a reward model. In *Proceedings of Advances in Neural Information Processing Systems*, 2023.

[77] Yu Meng, Mengzhou Xia, and Danqi Chen. SimPO: Simple preference optimization with a reference-free reward. In *Proceedings of Advances in Neural Information Processing Systems*, 2024.

[78] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Proceedings of Advances in Neural Information Processing Systems*, 2022.

[79] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning*, 2021.

[80] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the International Conference on Machine Learning*, 2023.

[81] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. In *Proceedings of Advances in Neural Information Processing Systems*, pages 34892–34916, 2023.

[82] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. In *Proceedings of Advances in Neural Information Processing Systems*, 2023.

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

[84] Shangpin Peng, Senqiao Yang, Li Jiang, and Zhuotao Tian. Mitigating object hallucinations via sentence-level early intervention. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 635–646, 2025.

[85] 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.

[86] Gengluo Li, Shangpin Peng, Xingyu Wan, Chengquan Zhang, Hao Feng, Xin Xu, Pian Wu, Bang Li, Zengmao Ding, Yongge Liu, et al. Chronicles-OCR: A cross-temporal perception benchmark for the evolutionary trajectory of chinese characters. *arXiv preprint arXiv:2605.11960*, 2026.

[87] Yongxin Shi, Chongyu Liu, Dezhi Peng, Cheng Jian, Jiarong Huang, and Lianwen Jin. M5HisDoc: A large-scale multi-style chinese historical document analysis benchmark. In *Proceedings of the Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

ChartArena: A Unified Benchmark with Atomic-Primitive Reasoning for Chart Parsing

Supplementary Material

Overview

This material provides supplementary details to the main paper, including the following sections:

- (A) **Benchmark Details**
  - (A.1) Image Sources and Scenarios
  - (A.2) Annotation Protocol and Human Effort
  - (A.3) Fine-grained Data Statistics
  - (A.4) Benchmark Samples
- (B) **Training Details**
  - (B.1) Training Setup
  - (B.2) Training Data
- (C) **Further Analysis of CAP**
  - (C.1) Human Verification of CAP Targets
  - (C.2) Isolating the Contribution of CAP
  - (C.3) Attribution of Real-World Robustness to Training Data
  - (C.4) Scaling CAP to a Larger Backbone
- (D) **Evaluation Details**
  - (D.1) Format-Agnostic Routing and Normalization
  - (D.2) Triple-Based Scoring for Numeric Charts
  - (D.3) Graph- and Tree-Based Scoring for Diagrammatic Charts
  - (D.4) Aggregation into Exact Match and mAP
  - (D.5) Evaluation Setup
- (E) **Further Evaluation Results**
  - (E.1) Detailed Analysis under Different Visual Scenarios
  - (E.2) Analysis of the English–Chinese Performance Gap
  - (E.3) Fine-grained Analysis on Hard Samples
  - (E.4) Consistency Across Evaluation Metrics
  - (E.5) Node/Edge Weight Sensitivity in Flowchart Scoring
  - (E.6) Statistical Significance
  - (E.7) Additional Model Results
- (F) **Further Related Work**
- (G) **Broader Impact**
- (H) **LLM Usage Statement**
- (I) **Further Case Study**

A Benchmark Details

This section complements Sec. 3.1 with additional details on how ChartArena is collected and annotated. We describe the image sources under three real-world scenarios in Sec. A.1, the human–agent collaborative annotation protocol and the corresponding human-effort budget in Sec. A.2.

A.1 Image Sources and Scenarios

As outlined in Sec. 3.1, the design goal of ChartArena is to cover as broad a distribution of real-world chart images as possible, so that the benchmark presents a more challenging test bed for chart parsing models rather than favoring synthetic or templated image inputs. Images are primarily collected through web image search. When certain categories are under-represented, we further supplement the corpus by asking in-house annotators to *scan, photograph, or hand-draw* additional samples so that each category reaches the target size and the benchmark coverage remains complete.

All collected images must satisfy the following quality requirements: (i) all textual content must be human-readable; (ii) the chart must reflect *real-world data* rather than template-style placeholder data; and (iii) the image must be a single complete chart without heavy occlusion or cropping.

We organize the benchmark around three complementary visual scenarios, detailed below.

- **Digital rendering.** Digitally rendered charts taken directly from real-world documents, without any physical-capture distortion. Existing chart parsing benchmarks may be dominated by synthetically rendered images [1], and seldom stress-test chart parsers on the complex digital-native charts that appear in practice. To close this gap, we harvest charts from several sources such as academic papers, product launches, industry reports, financial disclosures, and analytical whitepapers, capturing the diversity of layouts, legends, and annotations found in real digital-native materials.
- **Printed photo.** Charts captured by camera from a printed page or an electronic screen. We first try to collect web-native photographs of printed charts; when such samples are scarce, annotators print or screen-display collected documents and re-capture them with a camera. We explicitly require the capture to reproduce real-world interference such as uneven illumination, reflections, and moiré patterns on screens. Annotators are instructed to keep all text visually legible, to focus on the chart region, and are allowed to introduce moderate perspective tilt to reflect casual handheld capture.
- **Hand-drawn photo.** Charts that are hand-drawn and photographed. On top of the real-world lighting conditions of the previous scenario, this category additionally introduces handwriting-specific variations: irregular fonts, non-uniform layout, strike-throughs, and edits. It is by far the hardest and most long-tailed scenario to collect. We first crawl high-quality handwritten charts that meet our quality bar via image search; for the remaining, annotators *re-draw* a subset of the complex digital-rendering and printed-photographed charts found during search, so that different annotators contribute diverse handwriting styles onto the same underlying chart semantics.

CAP Extraction Prompt Template

```
<|im_start|>user
<image>
Based on the chart image and markdown annotation, extract the chart's atomic
semantic information.
```
Only extract structural semantics. No trend analysis, no data reconstruction.

[Input Information]

- Chart image: title, axis labels, tick ranges
- Markdown: groups, series, fields

[Extraction Targets]

1. **chart_title**
   - Main title
2. **comparison_items**
   - Series or group names
   - Prefer markdown:
     - Multi-column → column names (except first)
     - Single metric → first column (groups)
   - No numerical values

3. **x_axis**
   - semantic (e.g., year, category)
   - tick_range: type, min/max, values
4. **y_axis**
   - semantic (e.g., value, index)
   - tick_range: type, min/max, values
5. **statistics_fields**
   - If present: min, Q1, median, Q3, max, outliers
   - Else: null
6. **caption**
   - Include additional global descriptive text if present

[Output Format]

```
{
"chart_title": string | null,
"comparison_items": [string],
"x_axis": { "semantic": string | null, "tick_range": {...} },
"y_axis": { "semantic": string | null, "tick_range": {...} },
"statistics_fields": [string] | null,
"caption": string | null
}
```

[Input]

```
markdown: {markdown}
<|im_end|>
<|im_start|>assistant
```

A.2 Annotation Protocol and Human Effort

We rely on a human–agent collaborative annotation pipeline with multi-round verification to guarantee high-quality ground truth of ChartArena. In the pre-annotation stage, multiple MLLMs independently generate candidate annotations, which are then cross-checked and merged into a higher-quality draft before human refinement. Human annotators subsequently verify and correct both the structural semantics and fine-grained content details. To further improve annotation reliability, two complementary cross-checking mechanisms are applied throughout: (i) *value-level cross-check*, where uncertain numeric entries identified during annotation are independently re-labeled by multiple annotators; and (ii) *batch-level review*, where, after one full annotation pass, a separate reviewer performs a unified quality check over the whole batch.

The pipeline organizes the $2{,}400$ images into two annotation streams according to the target format: a Markdown stream that covers the $1{,}800$ numeric charts together with the 300 mind maps, and a Mermaid stream that covers the 300 flowcharts. We describe the two streams in turn.

Numeric charts and mind maps. For these chart types, the ground truth is serialized as Markdown: numeric charts use a Markdown table, while mind maps use a nested Markdown unordered list. We define chart-type-specific annotation guidelines (column ordering, unit normalization, nesting rules, etc.) that annotators must strictly follow. During annotation, annotators use auxiliary aids such as reference grids and guidelines to read off values accurately. Since hand-drawn charts and low-resolution photographs often contain values that cannot be read unambiguously, multiple annotators *cross-check* all uncertain cells before finalizing the label.

Flowcharts. For flowcharts, the ground truth is serialized as Mermaid code. After annotation, annotators are required to render the produced Mermaid code with an online visualization tool and compare the rendered diagram against the source image, verifying that (i) the set of nodes, (ii) the connectivity of edges, and (iii) the overall logical flow all match. Flowchart annotation is substantially more demanding than the numeric stream, both because of the richer topology and because a single misrouted edge can break the semantics of the whole diagram.

Human-effort budget. The resulting per-image time cost and total human-effort budget (in person-days, 8 working hours per day) for both annotation and quality review are summarized in Tab. A.1.

Table A.1: **Human-effort budget for ChartArena annotation.** Rows are grouped by *annotation stream*: numeric charts and mind maps share a Markdown-based labeling workflow, whereas flowcharts use Mermaid-based workflow. Together, they comprise the 2,400-image benchmark.

Annotation Stream	#Images	Annotation		Quality Review	
		Per image (min)	Total (person-day)	Per image (min)	Total (person-day)
Markdown stream	2,100	19.7	86.2	8.9	38.9
Mermaid stream	300	46.3	28.9	17.1	10.7
Total	2,400	–	115.1	–	49.6

Table A.2: **Fine-grained statistics of ChartArena.** *Left*: per chart type and language, text density is mean ground truth characters, and structural complexity is mean triples (numeric charts), nodes (mind maps), and nodes plus edges (flowcharts). *Right*: per visual scenario and language, image-quality proxies are mean resolution (megapixels), compression ratio (bytes/pixel), and Laplacian variance (higher = sharper).

Chart type	Text density		Complexity	
	EN	ZH	EN	ZH
Bar	404.9	207.4	27.3	12.6
Line	290.0	271.8	20.5	21.2
Pie	241.4	157.1	7.6	7.3
Radar	389.5	301.4	25.0	23.7
Box plot	411.2	334.7	30.1	26.5
Comb. chart	695.5	356.6	54.9	27.7
Flowchart	1076.4	844.9	36.3	43.5
Mind map	537.0	522.8	21.5	28.2

Scenario	Resolution		Bytes / pixel		Sharpness	
	EN	ZH	EN	ZH	EN	ZH
Digital rendering	0.56	0.61	0.50	0.50	2067	2144
Printed photo	1.25	0.79	0.94	0.96	2617	1439
Hand-drawn photo	1.95	0.86	0.56	0.92	649	1001

A.3 Fine-grained Data Statistics

ChartArena is balanced by chart type, visual scenario, and language by design. To characterize the benchmark beyond these three axes and provide meaningful comparisons, we report four objectively computable dimensions, each crossed with language or visual scenario, in Tab. A.2 and Fig. A.1.

Text density. We count the characters in each ground truth sample. Chinese and English samples are comparable in some chart types. For instance, 523 versus 537 characters on mind maps and 272 versus 290 on line charts.

Structural complexity. We measure complexity with the same units as the evaluation backends, namely the number of triples for numeric charts, nodes for mind maps, and nodes plus edges for flowcharts. Averaged over chart families, the Chinese subset is simpler on numeric charts (20.2% fewer triples) and more complex on diagrammatic charts (24.6% more structural units).

Image quality. We use two proxies, the compression ratio in bytes per pixel and the Laplacian variance as a sharpness measure. Averaged over both languages, the Laplacian variance decreases from 2106 on digital renderings to 2024 on printed photos and drops sharply to 827 on hand-drawn photos. The English hand-drawn subset is the least sharp (649) even though it has the highest average resolution (1.95 megapixels), which shows that visual difficulty is driven mainly by the capture channel rather than by surface resolution or language.

Topics. We assign each chart to one of seven named topics or a general bucket. The shares are general (23.9%), economics and business (21.7%), science and engineering (18.4%), society, politics, and demographics (11.7%), health and medicine (10.2%), nature, environment, and geography (5.2%), education and academia (5.1%), and culture, sports, and entertainment (3.8%). The distribution across visual scenarios is shown in Fig. A.1(d).

Coverage of difficult real-world charts. Images are screened with the quality requirements of Sec. A.1: every text must be human-readable, the chart must carry real-world data, and the image must be a single complete chart without heavy occlusion or cropping. Images that fail these requirements are discarded because their ground truth cannot be recovered reliably by human annotators. Difficult samples on which annotators can still reach agreement, such as low-contrast photographs or irregular handwriting, are kept, and uncertain values in them are resolved by the multi-annotator cross-check described above.

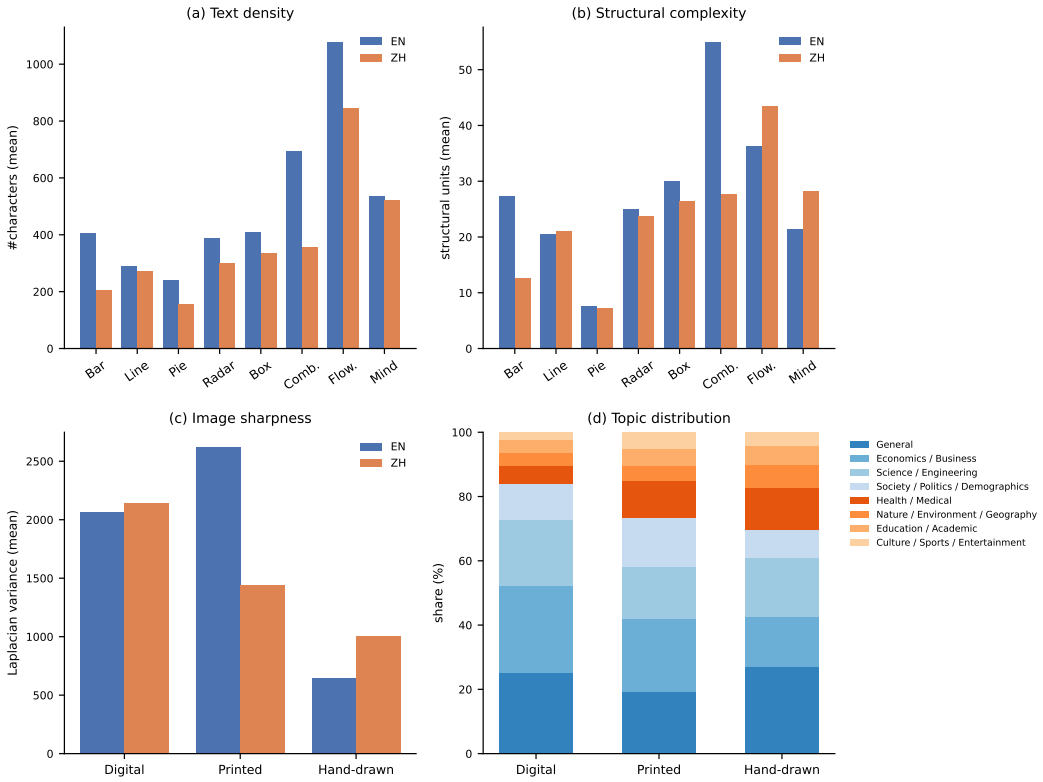
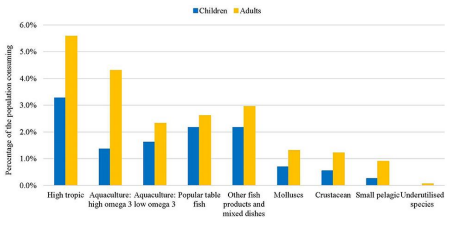


Figure A.1: **Fine-grained data distributions of ChartArena.** (a) Mean number of ground-truth characters per chart type and language. (b) Mean structural complexity per chart type and language, measured in triples, mind-map nodes, or flowchart nodes plus edges. (c) Mean Laplacian variance per visual scenario and language. (d) Topic distribution per visual scenario.

A.4 Benchmark Samples

To make the three scenarios introduced in Sec. A.1 more concrete, we show several representative samples in ChartArena. Each sample consists of the original image and its ground truth annotation.

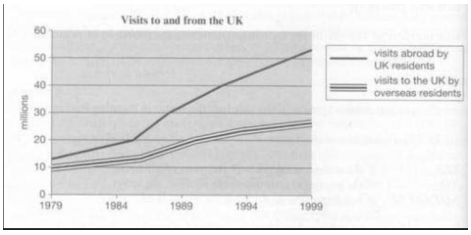


(a) **Bar chart** – digital.

GT annotation – Markdown table

```
| | Children-Percentage | Adults-Percentage |
| -- | -- | -- |
| High tropic | 3.3% | 5.6% |
| Aquaculture:  high omega 3 | 1.4% | 4.3% |
| Aquaculture:  low omega 3 | 1.6% | 2.4% |
| Popular table fish | 2.2% | 2.6% |
| Other fish | mixed dishes | 2.2% | 3.0% |
| Molluscs | 0.7% | 1.3% |
| Crustacean | 0.6% | 1.2% |
| Small pelagic | 0.3% | 0.9% |
| Underutilised species | | 0.1% |
```

Figure A.2: Representative sample: multi-series bar chart.

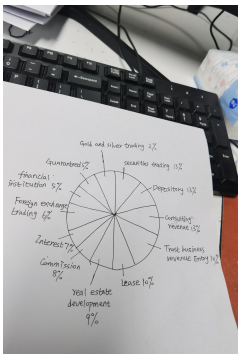


(b) **Line chart** – photo (printed).

GT annotation – Markdown table

```
### Visits to and from the UK
| | visits abroad by UK residents-millions | visits
to the UK by overseas residents-millions |
| -- | -- | -- |
| 1979 | 13.5 | 10 |
| 1984 | 18.5 | 12.5 |
| 1989 | 32.5 | 18.5 |
| 1994 | 43.5 | 24 |
| 1999 | 53 | 26.5 |
```

Figure A.3: Representative sample: line chart.

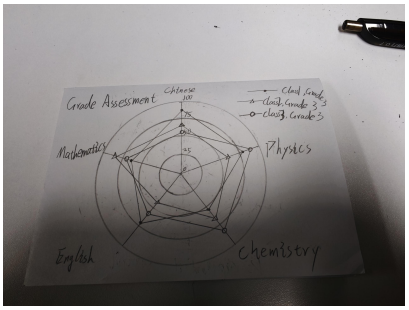


(c) **Pie chart** – hand-drawn.

GT annotation – Markdown table

```
| | |
| -- | -- |
| securities trading | 13% |
| Depository | 13% |
| consulting revenue | 13% |
| Trust business revenue Entry | 10% |
| Lease | 10% |
| real estate development | 9% |
| commission | 8% |
| Interest | 7% |
| Foreign exchange trading | 6% |
| financial institution | 5% |
| Guaranteed | 5% |
| Gold and silver trading | 2% |
```

Figure A.4: Representative sample: hand-drawn pie chart with 12 labelled slices.

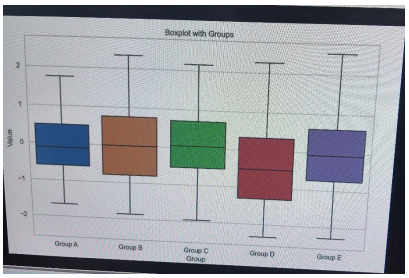


(d) **Radar chart** – hand-drawn, 3 series.

GT annotation – Markdown table

```
### Grade Assessment
| | Class1, Grade3 | Class2, Grade3 | Class3, Grade3 |
| -- | -- | -- | -- |
| Chinese | 87 | 65 | 55 |
| Physics | 79 | 60 | 90 |
| Chemistry | 54 | 67 | 83 |
| English | 72 | 42 | 58 |
| Mathematics | 64 | 85 | 70 |
```

Figure A.5: Representative sample: hand-drawn radar chart with overlapping rings.



(e) **Box plot** – photo (printed).

GT annotation – Markdown table

```
### Boxplot with Groups
| Group | Value-min | Value-Q1 | Value-median | Value-Q3 |
Value-max |
| -- | -- | -- | -- | -- | -- |
| Group A | -1.6 | -0.6 | -0.1 | 0.5 | 1.8 |
| Group B | -1.9 | -0.8 | 0 | 0.8 | 2.3 |
| Group C | -1.96 | -0.5 | 0.05 | 0.7 | 2.2 |
| Group D | -2.4 | -1.5 | -0.7 | 0.1 | 2.1 |
| Group E | -2.3 | -1.0 | -0.4 | 0.4 | 2.2 |
```

Figure A.6: Representative sample: box plot.

B Training Details

In this section, we provide additional training details, aiming to present the experimental setup more comprehensively and facilitate reproducibility for the community.

B.1 Training Setup

We summarize the detailed training configurations for the pretraining and supervised fine-tuning (SFT) stages in Tab. B.1. Full-parameter fine-tuning is applied in all stages with a maximum context length of 8192 tokens and a maximum image resolution of 2K.

Table B.1: **Training setup and hyperparameters for pretraining and SFT.**

	Setting	Pretraining	SFT
ViT	Max Resolution	2K	2K
	Tokens per Image	256-1280	256-1280
Data	Dataset	Synth + Public	Public + In house
	Raw Samples	4M	80K
	Training Samples	4M	80K
Model	Trainable Parameters	All	All
	Training Paradigm	Direct Parsing	CAP Reasoning
	Sequence Length	8192	8192
Training	Batch Size	128	128
	Epoch	1	1
	Learning Rate	4×10^{-5}	2×10^{-5}
	min-LR	2×10^{-6}	1×10^{-6}
	LR scheduler	Cosine	
	Optimizer	Adam	
	Hardware	16 $\times$ NVIDIA H20 GPUs	

B.2 Training Data

Pretraining data. Our pretraining corpus combines two sources. First, we select 0.8M samples from ChartGalaxy [33] and 1.2M samples from ChartNet [34], two high-quality synthetic chart corpora, for a total of 2M filtered image–annotation pairs. Since both sources cover only numeric chart families and do not include topologically richer diagrammatic charts such as flowcharts and mind maps, we further augment the corpus with another 2M image–text pairs synthesized by the data pipeline described in Sec. 4, where we not only fill the missing diagrammatic slice but also substantially expand the diversity of numeric chart coverage.

SFT data. Our SFT dataset has two parts. The first is 60K high-quality human-annotated samples released with ChartNet [34], which remain synthetic in their underlying image source. To complement this with real-world visual variation, we contribute an additional 20K in-house human-annotated samples spanning three visual styles: digital renderings, printed scans, and hand-drawn scans. The rendering perturbations of Sec. 5.2, namely font variation and geometric warping, are applied only to the ChartNet images and transform them in place, so the number of training samples is unchanged; the in-house images are used as captured. The contribution of each data source is analyzed in Sec. C.3.

CAP-based reasoning construction. Following Sec. 4.3, we first extract atomic primitives from each (I, Y) pair by prompting an MLLM; the full extraction prompt is shown in prompt 1. Notably, the prompt is slightly adapted per chart family to reflect the chart-type-specific schema of Sec. 4.2 (e.g., pie charts replace the x/y-axis slots with "names of slices"), but in all cases the extraction is restricted to chart-level semantics while discarding low-level visual details. Given the extracted atomic primitives A, we then assemble a reasoning trace following Eq. (2), whose templating logic is summarized in Algorithm 1: the atomic fields are serialized into an ordered sequence of semantic analysis steps (chart identification, data subject, structural analysis) and wrapped into the reasoning trace R. Finally, R is combined with the parsing answer Y via Eq. (3) to form the unified supervision target T used throughout SFT stage.

Table C.1: **Controls for the numeric-chart CAP target.** Numeric-chart mAP_{high} under three visual scenarios. The first two rows are rows (b) and (e) of Tab. 4. The three controls keep the training data, token budget, and hyperparameters of row (e) and change only the numeric-chart target.

Numeric-chart target	Digital	Printed	Drawn
Direct (w/o CAP), row (b) of Tab. 4	63.1	57.8	44.9
CAP, row (e) of Tab. 4	**64.1**	**60.4**	**48.6**
(i) Length-matched natural-language rewrite	60.1	52.2	41.8
(ii) CAP steps in random order	61.3	54.3	42.1
(iii) CAP with steps 2 and 3 swapped	63.5	58.2	46.9

C Further Analysis of CAP

This section complements the ablation study in Tab. 4 with further analyses. We verify the quality of the generated CAP targets in Sec. C.1, separate the contribution of CAP from target length and canonicalization in Sec. C.2, attribute the gains on printed and hand-drawn charts to the training data sources in Sec. C.3, and examine whether CAP carries over to a larger backbone in Sec. C.4.

C.1 Human Verification of CAP Targets

As in Sec. 4.3, the CAP reasoning targets are generated by an MLLM, so errors in axes, entity relations, or hierarchies could in principle enter the supervision. A full manual review of all samples is impractical. We therefore randomly sample 400 CAP targets and have four independent annotators verify them under the same quality-review protocol of the benchmark (Sec. A.2). The verified targets reach an accuracy of about 97%. This high reliability is due to two factors: primitives are extracted from human-verified ground-truth pairs, and the fixed CAP schema restricts extraction to a few chart-level fields, leaving little room for inconsistency.

C.2 Isolating the Contribution of CAP

The gain of CAP in Tab. 4 could in principle come from longer targets, a fixed template, or a canonical ordering rather than from the structural primitives themselves. Because CAP is carried by an explicit reasoning trace for numeric charts and by the target structure for flowcharts (Sec. 4.3), we design the controls separately for the two families. All control runs start from the same pretrained checkpoint and share the training data, token budget, and hyperparameters of row (e) of Tab. 4.

Numeric charts. We compare the direct SFT target of row (b) and the CAP target of row (e) with three variants of the numeric-chart target. (i) The length-matched rewrite keeps the content and length of the CAP trace but writes it as free-form natural language without the structured layout. (ii) The random-order variant shuffles the three CAP steps (chart identification, data subject, and structural analysis) independently for each sample. (iii) The swap variant keeps chart identification first and only exchanges the data-subject and structural-analysis steps. Results are reported in Tab. C.1.

At identical length and coverage, the natural-language rewrite falls 4.0, 8.2, and 6.8 points below CAP and even below the direct baseline. Longer targets are therefore not what drives the gain. A free-form reasoning path of the same length is hard for a 1B model to learn, whereas the structured layout of CAP is a much easier target to internalize. Random ordering loses 2.8, 6.1, and 6.5 points and also falls below the direct baseline, so a fixed template with an arbitrary ordering is not sufficient either. Swapping only the last two steps costs much less (0.6, 2.2, and 1.7 points) and stays above the direct baseline, which identifies chart identification as the critical anchor of the trace. The benefit of CAP thus comes from its structural primitives and their human-aligned ordering.

Flowcharts. Flowchart evaluation is inherently invariant to node and edge ordering because predictions are parsed into node/edge sets and scored via Hungarian matching (detailed in Sec. D.3). Any impact of target formatting thus comes from training rather than the metric. To separate canonicalization from the ordering direction, we train three models that differ only in how the flowchart target is arranged, while the numeric side is fixed to CAP. The three targets encode the same graph. (i) The interleaved target is the raw Mermaid style, in which a node is declared where an edge first refers to it; this is the direct setting of rows (b) and (c). (ii) The edge-first target is fully canonicalized but lists every edge before appending all node declarations. (iii) The node-first target applies the same canonicalization in the opposite direction and is the CAP setting used in the paper.

Table C.2: **Ordering ablation for the flowchart target.** mAP_{high} under three visual scenarios. The three SFT targets encode the same graph and differ only in where node declarations are placed; the numeric side is fixed to CAP. The interleaved and node-first rows correspond to the flowchart slice of rows (c) and (e) of Tab. 4.

Flowchart target	Digital	Printed	Drawn	Mean
Pretrain only	41.9	40.8	30.7	37.8
(i) SFT, interleaved (not canonicalized)	56.0	43.6	44.4	48.0
(ii) SFT, canonicalized, edge-first	55.5	42.1	42.4	46.7
(iii) SFT, canonicalized, node-first (CAP)	**56.8**	**48.3**	**47.8**	**51.0**

Table C.3: **Attribution of the SFT-stage gains to data sources.** mAP_{high} under three visual scenarios. Rows (a) and (b) are those of Tab. 4; rows (b_1)–(b_3) are variants of row (b) that change only the SFT data. All SFT runs start from the checkpoint of row (a), disable CAP, and are trained under the same token budget.

Setting	Numeric Charts			Diagrammatic Charts		
	Digital	Printed	Drawn	Digital	Printed	Drawn
MLLM Backbone	56.3	49.7	37.4	55.0	50.4	26.0
(a) Pretrain (synthetic only)	62.5	50.2	38.1	58.9	51.8	27.7
(b) + SFT, ChartNet + in-house (mixed)	63.1	**57.8**	**44.9**	59.6	**55.8**	**37.9**
(b_1) + SFT, mixed without augmentation	62.2	57.5	44.2	58.1	55.4	37.3
(b_2) + SFT, ChartNet only	**63.9**	51.2	39.9	58.3	52.1	28.1
(b_3) + SFT, in-house only	62.8	56.3	44.7	**60.2**	54.8	36.4

As shown in Tab. C.2, relative to the interleaved target, the edge-first layout loses 0.5, 1.5, and 2.0 points across scenarios. The node-first layout instead gains 3.0 points on average, yielding a 4.3-point gap attributable to ordering direction. Canonicalization alone is thus mildly harmful, with all benefits stemming from committing to the node set first. This aligns with CAP principles: nodes are flowchart primitives while edges define their relations, so a node-first target lets the model fix primitives before choosing edges. An edge-first target inverts this dependency, forcing models to emit relations for undeclared nodes. This pattern mirrors the numeric-chart controls where shuffled CAP steps underperformed. The gain is concentrated on photographed charts and minor on digital renderings (consistent with Sec. 6.4), matching the pooled diagrammatic columns in Tab. 4.

C.3 Attribution of Real-World Robustness to Training Data

The training data combines synthetic pretraining pairs, the ChartNet SFT set (with synthetic image sources), in-house annotations of real printed and hand-drawn charts, and rendering perturbations (Sec. B.2). We examine which component drives gains on printed and hand-drawn charts.

Constraints from the existing setup. Three facts constrain this attribution. First, row (a) of Tab. 4 is a synthetic-only control at scale: its entirely synthetic pretraining corpus lifts the four printed and hand-drawn cells by just 4.3 points (concentrating 70% of its gain on digital renderings), whereas the SFT stage adds 28.6 points—about 6.7 times as much. Second, ChartNet SFT images are synthetic, and similar samples were seen in much greater volume during pretraining. Third, rendering perturbations apply exclusively to ChartNet images, yielding only rendered charts. Real printed and hand-drawn photos exist solely in our in-house annotations. Moreover, because ChartNet only covers numeric charts, neither the synthetic set nor its perturbations touch diagrammatic charts.

Controlled ablations. We decompose the SFT stage into synthetic-only, in-house-only, augmentation-only, and mixed-data controls. All runs start from the pretrained checkpoint of row (a), keep CAP disabled so that data is the only variable, and are trained under the same token budget with the SFT configuration of the main paper. Results are reported in Tab. C.3, where rows (b_2), (b_3), (b) versus (b_1), and (b) instantiate the four controls, respectively.

Summing the four printed and hand-drawn cells, the gain over row (a) is only 3.5 points for (b_2) but 24.4 points for (b_3), seven times as much from the smaller in-house set and a $21\times$ difference once normalized per sample. The in-house real images alone thus account for about 85% of the total gain of 28.6 points. The perturbations are measured on a different contrast: removing them from the full mixture, (b) versus (b_1), costs 2.0 points on the same cells, a real but secondary effect. The contrast is clearest on flowcharts and mind maps, where (b_2) changes the three cells by only -0.6, $+0.3$, and $+0.4$ points while (b_3) changes them by $+1.3$, $+3.0$, and $+8.7$ points, as expected since ChartNet contains no diagrammatic charts.

Table C.4: **CAP on larger backbone Qwen3.5-9B.** *Top*: average mAP_{high} over all chart families, reported separately for English and Chinese. *Bottom*: mAP_{high} for the two chart categories under three visual scenarios.

Setting	Average (EN)	Average (ZH)
Qwen3.5-9B [56]	40.6	47.6
+ SFT (w/o CAP)	55.9	67.9
+ SFT (w/ CAP)	**58.8**	**71.0**

Setting	Numeric Charts			Diagrammatic Charts		
	Digital	Printed	Drawn	Digital	Printed	Drawn
Qwen3.5-9B	48.4	43.6	33.9	59.7	54.3	37.9
+ SFT (w/o CAP)	70.6	65.5	55.2	66.2	58.6	43.9
+ SFT (w/ CAP)	**72.1**	**67.7**	**60.8**	**67.4**	**61.0**	**48.2**

The two sources are complementary rather than redundant, which is why the full mixture is used in the main experiments. Row (b_2) reaches the best numeric digital score of the table (63.9, $+1.4$ over row (a)), whereas (b_3) stays nearly flat there ($+0.3$), so the synthetic set contributes rendering-level precision and the in-house set contributes real-world robustness. Relative to (b_2), adding the in-house set gives up 0.8 points on numeric digital renderings but gains 6.6 and 5.0 points on printed and hand-drawn numeric charts.

C.4 Scaling CAP to a Larger Backbone

To test whether CAP carries over beyond a 1B parser, we adopt Qwen3.5-9B [56], a newer and considerably larger general-purpose MLLM, as the backbone. Because it is already strong, we skip the pretraining stage and apply SFT only, so that the contrast between two SFT settings attributes it to CAP. All other settings, including the SFT data, the evaluation protocol, and the metric, follow the main experiments. We compare the backbone with SFT without and with CAP, whose configurations correspond to rows (b) and (e) of Tab. 4. Results are reported in Tab. C.4.

A stronger backbone learns chart parsing more effectively. The untrained Qwen3.5-9B scores below the HunyuanOCR backbone used in the paper (40.6/47.6 versus 41.0/51.8 on English/Chinese), which is expected since HunyuanOCR is specialized for OCR. With the same SFT data, however, Qwen3.5-9B with CAP reaches 58.8/71.0, well above the 49.5/61.3 of our 1B model in Tab. 2. We attribute this to the richer world knowledge and stronger general understanding of the larger backbone.

CAP remains beneficial at the 9B scale. Adding CAP on top of the same SFT recipe lifts the average by 2.9 points in English and 3.1 points in Chinese. The benefit of CAP does not vanish once the backbone becomes larger and more capable, indicating CAP as a scale-independent mechanism.

The gain concentrates on visually harder scenarios. The improvement grows as visual conditions degrade. On numeric charts the CAP gain goes from 1.5 points on digital renderings to 2.2 on printed photos and 5.6 on hand-drawn photos, and on diagrammatic charts it goes from 1.2 to 2.4 and then 4.3 points. This trend is consistent with Sec. 6.4 and confirms that making chart structure explicit mainly improves robustness under visually uncertain real-world inputs.

Algorithm 1 CAP-based Reasoning Trace Construction ($\mathcal{G}$)

Require: Dataset $\mathcal{D}_{\text{raw}} = \{(I_i, Y_i, \tau_i, A_i)\}_{i=1}^{N}$, where I_i denotes the chart image, Y_i the target annotation, τ_i the chart type, and A_i the extracted CAP atomic semantics (*e.g.,* title, axes)

Ensure: Supervised dataset $\mathcal{D}_{\text{sft}} = \{(I_i, T_i)\}$, where T_i is the structured target that encodes both CAP-based reasoning and the final output (see Eq. (3))

function CONSTRUCTSUPERVISIONTARGETS($\mathcal{D}_{\text{raw}}$)
  $\mathcal{D}_{\text{sft}} \leftarrow \emptyset$ ▷ Output buffer for unified supervision samples
  for each $(I, Y, \tau, A) \in \mathcal{D}_{\text{raw}}$ **do**
    # [Shortcut Branch] Types whose structure is already sequential
    # (e.g. flowcharts in Mermaid) need no extra reasoning scaffold
    if $\tau \in \mathcal{T}_{\text{short}}$ **then**
      $T \leftarrow$ "<answer>" $\oplus\ Y \oplus$ "</answer>" ▷ Directly wrap the annotation
      $\mathcal{D}_{\text{sft}} \leftarrow \mathcal{D}_{\text{sft}} \cup \{(I, T)\}$
      continue ▷ Skip reasoning construction for this sample
    end if

    # [Step 1] Chart Identification — high-level "what is this chart"
    # Extract chart type, title and caption as the opening statement
    $S_{\text{id}} \leftarrow \text{FORMATIDENTITY}(\tau, A.\text{title}, A.\text{caption})$

    # [Step 2] Data Subject — "what entities are being compared"
    # Different chart types expose different subject fields:
    # 2D chart → comparison_items / statistics_fields
    # radar → comparison_items / dimensions
    # mindmap → main_branches / semantic_domain
    # pie → categories / value_semantic
    $\mathcal{F}_{\text{subj}} \leftarrow \text{SELECTSUBJECTFIELDS}(\tau)$ ▷ Type-specific field selector
    $S_{\text{subj}} \leftarrow \text{FORMATFIELDS}(A, \mathcal{F}_{\text{subj}})$ ▷ Serialise selected fields to text

    # [Step 3] Structural Analysis — "how is the chart organized"
    # Captures axes semantics, tick ranges, tree depth, # sectors, etc.
    $\mathcal{F}_{\text{struct}} \leftarrow \text{SELECTSTRUCTFIELDS}(\tau)$ ▷ Type-specific structural fields
    $S_{\text{struct}} \leftarrow \text{FORMATFIELDS}(A, \mathcal{F}_{\text{struct}})$

    # [Step 4] Assemble the interleaved reasoning trace R
    # Format: <think> [3 sections] </think> <answer> Y **</answer>**
    $R \leftarrow$ "[Chart Identification]\n" $\oplus\ S_{\text{id}} \oplus$ "\n\n" ▷ Section 1
    $R \leftarrow R \oplus$ "[Data Subject]\n" $\oplus\ S_{\text{subj}} \oplus$ "\n\n" ▷ Section 2
    $R \leftarrow R \oplus$ "[Structural Analysis]\n" $\oplus\ S_{\text{struct}}$ ▷ Section 3
    $T \leftarrow$ "<think>" $\oplus\ R \oplus$ "</think>" ▷ Wrap reasoning trace
    $T \leftarrow T \oplus$ "<answer>" $\oplus\ Y \oplus$ "</answer>" ▷ Append ground-truth parse
    $\mathcal{D}_{\text{sft}} \leftarrow \mathcal{D}_{\text{sft}} \cup \{(I, T)\}$
  end for
  return $\mathcal{D}_{\text{sft}}$
end function

function FORMATFIELDS($A, \mathcal{F}$)
  # Generic serialiser: iterate over the selected fields of A **and**
  # emit "field: stringified_value" lines, skipping empty ones.
  $Lines \leftarrow []$
  for each field $f \in \mathcal{F}$ **do**
    $v \leftarrow A[f]$ ▷ Fetch value from atomic semantics
    if v is non-empty **then**
      $Lines$.Append("f: " $\oplus$ STRINGIFY(v)) ▷ e.g. list → comma-joined string
    end if
  end for
  return Join($Lines$, "\n") ▷ One field per line
end function

Algorithm 2 Routing a prediction to its canonical evaluation view.

Require: raw prediction $\hat{Y}$, reference Y, declared task τ
Ensure: canonical representation $X_{\hat{Y}}$ and view $v \in \{\text{triple}, \text{tree}, \text{graph}\}$

function RouteToCanonicalView($\hat{Y}, Y, \tau$)
  # [Step 1] Graph-based formats (primary path for diagrammatic charts)
  # Includes Mermaid, DOT, PlantUML, D2, Diagrams, Cytoscape, etc.
  if $\tau \in \mathcal{T}_{\text{graph}}$ **then**
    $X \leftarrow$ ParseGraph($\hat{Y}, \tau$) ▷ Dedicated DSL parser; failure ⇒ empty graph
    return (X, graph)
  end if

  # [Step 2] Tree-based formats (mind maps)
  # Identified by reference structure rather than surface format
  if Y is a Markdown bullet list **then**
    $X \leftarrow$ ToMarkdownTree($\hat{Y}, \tau$)
    return (X, tree)
  end if

  # [Step 3] Graph fallback (flowcharts expressed in Mermaid)
  if Y is Mermaid code **then**
    $X \leftarrow$ ToMermaid($\hat{Y}, \tau$)
    return (X, graph)
  end if

  # [Step 4] Numeric charts (default branch)
  # Covers bar, line, pie, radar, box plot, and combination charts
  $X \leftarrow$ ToInternalCSV($\hat{Y}, \tau$)
  return (X, triple)
end function

D Evaluation Details

This section provides a detailed description of the unified evaluation protocol in Sec. 3.2, and explains how ChartArena converts heterogeneous model outputs into a single, comparable score. We first describe how predictions are routed to one of two semantic views (Sec. D.1). We then formalize the two scoring backends on these views: a triple-based IoU for numeric charts (Sec. D.2) and a graph-matching score or tree-based score for diagrammatic charts (Sec. D.3). Finally, we describe how per-sample similarities are aggregated into the reported *EM* and *mAP* (Sec. D.4).

D.1 Format-Agnostic Routing and Normalization

For each evaluation sample, we start from the raw model output $\hat{Y}$ and the reference annotation Y. The evaluator first routes $\hat{Y}$ into a canonical representation based on two pieces of information: (i) the declared surface format of $\hat{Y}$, such as Markdown, CSV, JSON, code, or SVG; and (ii) the structural type of the reference Y, namely *numeric triples*, *hierarchical trees*, or *directed graphs*. The high-level routing procedure is shown in Algorithm 2.

Per-format adapters. Each surface format has a deterministic adapter that extracts the semantic content from the prediction and discards presentation details. For numeric charts, ToInternalCSV converts the prediction into a tabular view. For mind maps, ToMarkdownTree converts the prediction into a tree-structured Markdown view. For diagrammatic charts, the corresponding parser converts the prediction into a graph representation. These adapters do not attempt to correct semantic mistakes in $\hat{Y}$. Their only goal is to map different output languages into a shared canonical form.

Light-touch normalization on the canonical view. After routing, we apply only semantics-preserving normalization. This includes simple punctuation mapping, such as converting full-width symbols to half-width ones, removing symbols such as `$` and `%` from numeric cells, and rewriting equivalent box-plot headers into a unified suffix, such as mapping `lower quartile` and `Q1` to `-Q1`. We do not apply any task-specific heuristic correction. If a prediction cannot be normalized by these rules, we keep it unchanged for scoring.

Parse failure. If an adapter cannot produce a non-empty canonical representation, the sample is marked as `parse_failed`. Examples include syntactically broken Mermaid code or JSON outputs that contain no usable data payload. Such samples are assigned zero similarity under all thresholds.

D.2 Triple-Based Scoring for Numeric Charts

Our numeric-chart scoring follows the Structuring Chart-oriented Representation Metric (SCRM) proposed in StructChart [47]. We inherit its structured triples, tolerance-aware IoU matching, separate treatment of text through edit distance and numbers through relative error, three tolerance levels, and threshold-averaged mAP. Our adaptation routes heterogeneous surface formats into a common tabular view before matching. Graph and tree scoring extend the protocol to diagrammatic charts in Sec. D.3.

For numeric charts, both the prediction and the reference are first converted into a tabular CSV form, denoted as $X_{\hat{Y}}$ and X_Y. Here, $X_{\hat{Y}}$ is obtained from the model output $\hat{Y}$ through the routing step, and X_Y is the ground-truth table.

Triple construction. We convert each table into a set of unordered triples of the form (e, h, v), where e is the entity (e.g. row or column category), h is the header (e.g. series name), and v is the value. Given a table with R rows and C columns, this produces up to $R \times (C-1)$ triples.

To make the representation invariant to table orientation, the pair (e, h) is treated as an order-free tuple. As a result, transposing the table yields the same set of triples.

Pre-processing. Before matching, we apply several deterministic normalization steps: (i) all text fields e and h are lower-cased; (ii) values v are cast to floating-point numbers when possible; (iii) common header affixes in the reference (e.g. prefixes or suffixes in box plots) are aligned with the prediction; (iv) triples corresponding to outlier or scatter markers are removed if they are not present in the reference. These steps standardize representation but do not modify semantic content.

Tolerance-aware matching. Let $T_{\hat{Y}}$ and T_Y denote the triple sets from prediction and reference. Following recent studies [1, 2, 47], we define a tolerance-aware matching rule between two triples (e_1, h_1, v_1) and (e_2, h_2, v_2).

First, their text keys are compared by concatenating the entity and the header, and measuring the Levenshtein distance [64]:

$$\mathrm{Lev}(e_1 h_1, e_2 h_2) \le \epsilon_{\text{text}}. \tag{4}$$

Then, their values are compared as follows: if both v_1 and v_2 are non-numeric, we apply the same Levenshtein threshold; if both are numeric, we require the relative error to satisfy

$$\frac{|v_1 - v_2|}{|v_2| + \delta} \le \epsilon_{\text{num}}, \tag{5}$$

where δ is a small constant to avoid division by zero.

A pair of triples is considered a match only if both the text and value conditions are satisfied.

Final score. Using the above matching rule, we compute a tolerance-aware intersection $T_\cap$ and union $T_\cup$ between the two triple sets. The similarity score is then defined as

$$s = \frac{|T_\cap|}{|T_\cup|}, \tag{6}$$

which corresponds to an IoU-style metric. If $T_\cup = \emptyset$, we define $s = 0$.

Tolerance levels. The three evaluation levels (*strict*, *slight*, and *high*) correspond to different settings of $(\epsilon_{\text{text}}, \epsilon_{\text{num}})$, as listed in Tab. D.1.

D.3 Graph- and Tree-Based Scoring for Diagrammatic Charts

For diagrammatic charts, both the prediction $\hat{Y}$ and the reference Y are converted into structured representations. Depending on the chart type, we use either a graph representation for flowcharts and similar diagrams or a tree representation for mind maps.

Common graph representation. All supported diagram DSLs (*e.g.,* Mermaid, Graphviz DOT, PlantUML, D2, Cytoscape JSON, and Python-based diagram libraries) are parsed into a unified graph intermediate representation

Table D.1: **Tolerance parameters across the three views.** ϵ_{text} is the maximum Levenshtein edit distance on the concatenated text keys; ϵ_{num} is the relative numeric tolerance; for the tree view, θ_{path} is the minimum Levenshtein ratio between two root-to-node paths; for the graph view, θ_{item} is the minimum per-node / per-edge similarity allowed to enter the Hungarian matching.

Tolerance	**Triple ϵ_{text}**	**Triple ϵ_{num}**	**Tree θ_{path}**	**Graph θ_{item}**
strict	0	0	1.00	1.00
slight	2	0.05	0.85	0.85
high	5	0.10	0.60	0.60

$$\mathcal{G} = (V, \mathcal{L}_V, E), \tag{7}$$

where V is the set of nodes, $\mathcal{L}_V : V \to \Sigma^\star$ maps each node to its text label, and $E \subseteq V \times V \times \Sigma^\star$ is the set of directed edges with optional labels.

For Python-style outputs, parsing is performed via static AST analysis without executing the code. This ensures safety and avoids non-determinism.

Graph matching. Given a predicted graph $\mathcal{G}_{\hat{Y}}$ and a reference graph $\mathcal{G}_Y$, we compute similarity by matching nodes and edges separately.

Node similarity is defined using the Levenshtein ratio between node labels. Edge similarity is defined as the average of three components: source node, target node, and edge label. If both edge labels are empty, they are treated as a perfect match.

We construct two similarity matrices: one for nodes and one for edges. Each matrix is then matched using the Hungarian algorithm, which finds the optimal one-to-one assignment.

After matching, only pairs with similarity above a threshold θ_{item} are kept. The node and edge scores are computed as the average similarity over the matched pairs, normalized by the larger set size:

$$\text{Match}_V = \frac{1}{\max(|V_1|, |V_2|)} \sum S^V_{ij}, \quad \text{Match}_E = \frac{1}{\max(|E_1|, |E_2|)} \sum S^E_{ij}. \tag{8}$$

The final graph similarity is a weighted sum:

$$s = w_E \cdot \text{Match}_E + w_V \cdot \text{Match}_V, \quad (w_E, w_V) = (0.6, 0.4), \tag{9}$$

which places more emphasis on edge correctness, as topology errors are more critical in practice.

Tree scoring for mind maps. For mind maps, we use a simpler tree-based formulation. Each Markdown bullet list is converted into a set of root-to-node paths. Specifically, every prefix of every leaf path is treated as an individual path.

Two paths are compared by computing the Levenshtein ratio between their string representations (joined by `" -> "`). We then construct a similarity matrix between the predicted and reference path sets and apply Hungarian matching.

The final score is obtained by averaging the similarities of matched path pairs that exceed a threshold θ_{path}. This formulation rewards partial structural correctness, such as correctly identifying top-level branches even if some leaf nodes are incorrect.

D.4 Aggregation into Exact Match and mAP

For each evaluation sample i, the previous sections produce a similarity score $s_i^{(t)} \in [0, 1]$ under each tolerance level $t \in \{\text{strict}, \text{slight}, \text{high}\}$. These similarities are the only inputs to the final metrics.

Exact Match (EM). Exact Match measures the fraction of samples that are perfectly recovered under the strict setting:

$$\text{EM} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[s_i^{(\text{strict})} = 1\right]. \tag{10}$$

A sample contributes to EM only if its prediction matches the reference exactly, without any tolerance.

AP at a fixed threshold. To capture partial correctness, we evaluate whether each sample exceeds a similarity threshold θ. For any threshold $\theta \in [0,1]$ and tolerance level t, we define

$$\mathrm{AP@}\theta^{(t)} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[s_i^{(t)} \geq \theta\right], \tag{11}$$

which is the fraction of samples whose quality is above the bar θ.

Mean Average Precision (mAP). Instead of fixing a single θ, our primary evaluator averages $\mathrm{AP@}\theta$ over a dense grid of thresholds to obtain a threshold-agnostic measure. For each tolerance level t, we compute

$$\mathrm{mAP}^{(t)} = \frac{1}{10}\sum_{\theta \in \{0.50, 0.55, \ldots, 0.95\}} \mathrm{AP@}\theta^{(t)}. \tag{12}$$

This follows standard practice in detection benchmarks and reflects overall performance across quality levels; it is the metric backing all headline numbers in this paper (main results, ablations, and the aggregated tables in the appendix all report $\mathrm{mAP}_{\mathrm{high}}$, which is Eq. (12) with $t = \mathrm{high}$).

Reporting single-threshold slices for diagnostics. For finer-grained diagnostics we also report $\mathrm{AP@}\theta^{(t)}$ at three representative thresholds, $\theta \in \{0.5, 0.75, 0.90\}$. These are simply instantiations of the AP definition at three fixed cut-offs; they are not an alternative aggregation of mAP but three points sampled from the same $\mathrm{AP@}\theta$ curve. In particular, the ten thresholds used in Eq. (12) for aggregation and the three thresholds used here for point reporting serve different purposes and are consistent by construction; the paper does not aggregate mAP over $\{0.5, 0.75, 0.90\}$, nor does it report AP at all ten grid thresholds.

Final reporting. For each tolerance level t, the evaluator therefore emits (i) EM, (ii) the aggregate $\mathrm{mAP}^{(t)}$ from Eq. (12), and (iii) three point-wise $\mathrm{AP@}\theta^{(t)}$ values at $\theta \in \{0.5, 0.75, 0.90\}$. All headline comparisons in the main text use $\mathrm{mAP}^{(\mathrm{high})}$; EM and the AP slices are used only as complementary diagnostics.

Interpretation. Reporting both EM and mAP provides complementary signals, where EM reflects strict, exact recovery, while mAP captures graded similarity. For example, a model may achieve high mAP but low EM if it produces structurally correct outputs with small numerical errors. Conversely, a model may have non-trivial EM but low mAP if it occasionally outputs perfectly correct results but fails on most samples.

D.5 Evaluation Setup

In this section, we introduce the detailed evaluation setup used in ChartArena. Our evaluation pipeline is strictly aligned with prior works [1, 2, 47] to ensure fairness and reproducibility.

Inference prompt. For document parsing MLLMs and expert chart understanding models, we use their officially recommended prompts for inference and configure the corresponding output formats according to each evaluation target. In particular, for our training backbone, we directly adopt the official parsing prompt released by HunyuanOCR[1] as the direct parsing prompt. For CAP reasoning, we additionally append the following instruction to the original prompt: *"Please first analyze the chart information step by step, and then provide the final parsing result."*

For general-purpose MLLMs, we use a *unified prompt template* per chart family, shown in prompts 2 to 4. The template is not tuned per model or on the test set. Instead, it is consolidated from two sources, namely the parsing prompts used in prior chart-parsing works [1, 2, 47] and the officially recommended prompts released by each model vendor for structured extraction, and then *frozen* before benchmark evaluation. The same frozen template is applied to every general-purpose MLLM.

Cross-model inference protocol. To make the cross-model comparison as transparent as possible, we detail the exact inference protocol used for every model in Tab. 2:

- **Single-pass direct parsing.** Every score corresponds to a single forward pass per image. We do not use self-refinement, multi-turn feedback, retrieval, or tool use. The model receives the image and the frozen prompt and emits one response, from which the structured output is parsed.
- **Decoding parameters.** We call each model through its native OpenAI-compatible or vendor API with default decoding: temperature is left unset, no explicit `top_p` or `max_tokens` override is applied, and no logit bias is used. We keep decoding uniform rather than per-model tuned to avoid confounding language ability with generation-parameter engineering.

[1] `https://github.com/Tencent-Hunyuan/HunyuanOCR`

- **Retry policy.** A sample is retried when the API returns a transport-level error (timeout, HTTP 5xx, or empty content). Retries reuse the same prompt and the same image.
- **Call and token budget.** For each of the 2,400 benchmark images we issue *one* inference call under the above protocol, yielding 2,400 calls per general-purpose model in the main table. No token budget or cost cap is imposed at inference time.
- **Scope of the fairness claim.** We report cross-model numbers strictly under this fixed, single-pass, frozen-prompt setting, and interpret the resulting comparison as reflecting each model's ability to directly parse charts. Systems built on top of an MLLM with heavier inference-time machinery (self-refinement, tool use, agentic pipelines) are outside the scope of this comparison.

Numeric Chart Parsing Prompt

Please parse the chart content in the image and extract the data into a structured Markdown **table** format.

Requirements:

1. Focus only on the chart itself and ignore unrelated elements such as decorations, backgrounds, logos, and watermarks;
2. If both category labels and numerical units are present (e.g., axis labels), merge them into the table header using the format "category-unit";
3. Preserve all category labels exactly as they appear in the chart without translation or rewriting;
4. Preserve the original semantics and numerical precision of all values.

Mind Map Parsing Prompt

Please parse the chart content in the image and extract the data into a structured Markdown **multi-level unordered list** format.

Requirements:

1. Use unordered lists beginning with '-', where each node text is represented as a list item;
2. Determine the hierarchy according to the connection relationships between nodes, where parent nodes correspond to higher-level list items and child nodes correspond to nested list items;
3. Fully extract all text contained in each node or box while preserving the original language and punctuation.

Flowchart Parsing Prompt

Please carefully analyze the following **flowchart** image and fully transcribe it into Mermaid flowchart code.

Requirements:

1. Use Mermaid `flowchart` or `graph` syntax (preferably `flowchart TD` or `flowchart LR` according to the actual direction of the diagram);
2. Strictly preserve all node text, including the original language and punctuation, without translation, rewriting, or simplification;
3. Correctly reconstruct all node connections and edge directions. If conditional branches exist, preserve branch labels such as "Yes/No";
4. Ignore decorations, backgrounds, logos, watermarks, and other irrelevant visual elements;
5. The output must strictly contain two stages:
   - First, declare all nodes line by line without edges;
   - Then, declare all relationships line by line without repeating node text.

E Further Evaluation Results

E.1 Detailed Analysis under Different Visual Scenarios

In this section, we further analyze model performance under three visual scenarios in Tab. E.1, including digital rendering, printed photo, and hand-drawn photo.

Impact of visual scenarios. Compared with digitally rendered charts, printed photos introduce additional visual difficulty caused by camera noise, illumination variation, blur, perspective distortion, and printing artifacts, leading to a consistent performance drop across nearly all models and chart categories. The hand-drawn photo scenario is substantially more challenging, since it additionally contains irregular strokes, imperfect structures, inconsistent layouts, and ambiguous visual boundaries. As a result, models generally exhibit a much larger degradation under hand-drawn settings, revealing limited robustness to severe visual uncertainty and distribution shift.

Numeric charts vs. diagrammatic charts. Diagrammatic charts suffer much heavier degradation than numeric charts when moving from digital renderings to printed or hand-drawn settings. While numeric charts mostly depend on extracting geometric and numerical values, diagrammatic charts require recovering complex topological structures, like node-link relationships, directional arrows, and hierarchical layouts. Because these structures are fragile under real-world visual perturbations, robustly reconstructing diagrammatic charts is significantly harder for current models.

Analysis across model categories. Across categories, the general trends align with Tab. 2. Among general-purpose MLLMs, Gemini 3.1 Pro achieves top performance and overall robustness. On numeric charts, Seed2.0 Pro ranks second for digital and printed formats, while Kimi K2.5 takes second on hand-drawn photos; Seed2.0 Pro also leads the runner-up spot for diagrammatic charts.

Among expert models, our approach outperforms competitors across nearly all scenarios. Its strong performance under printed and hand-drawn conditions demonstrates that our training strategy improves both structural comprehension and resilience to real-world perturbations—underscoring that explicitly modeling chart semantics is essential for robust understanding beyond clean synthetic data.

Document parsing models on numeric and diagrammatic charts. Document parsing MLLMs handle several numeric chart families reasonably well but remain weaker on diagrammatic structures. In Tab. 2, HunyuanOCR scores 70.8 EN / 72.5 ZH on pie charts, compared with 38.3 EN / 35.5 ZH on flowcharts and 55.0 EN / 46.6 ZH on mind maps. On the two diagrammatic families, it trails Gemini 3.1 Pro by 27.3 EN / 27.6 ZH and 31.8 EN / 38.6 ZH points, respectively. The pattern also appears in dots.mocr, whose flowchart and mind-map scores range from 19.6 to 28.7, while PaddleOCR-VL does not support these families in this evaluation. These results indicate that numeric extraction capability does not necessarily transfer to recovering node relations and hierarchies.

Remaining robustness gap on hand-drawn numeric charts. Zooming in on the digital-to-hand-drawn performance drop for numeric charts clarifies CAP's exact contribution. Compared to the HunyuanOCR backbone, which drops -18.9 points ($56.3 \rightarrow 37.4$), our SFT-with-CAP model (row e, Tab. 4) reduces the gap to -15.5 ($64.1 \rightarrow 48.6$)—effectively absorbing about 18% of the loss. By forcing the model to infer atomic primitives (axes, entities, series, units) before parsing, CAP decouples numeric extraction from pixel-level recognition, improving transfer to hand-drawn images.

However, our remaining drop (-15.5) still lags behind frontier MLLMs like Gemini 3.1 Pro (-6.6, $67.0 \rightarrow 60.4$). We attribute this residual gap to visual prior capacity tied to model scale rather than structural reasoning limits: parsing irregular strokes, uneven lighting, and non-uniform layouts is a general vision problem whose ceiling depends on the encoder's pre-trained prior. A 1B specialized backbone simply cannot match the pixel-level robustness of massive MLLMs.

Ultimately, structural reasoning and parameter scale offer complementary paths to robustness: CAP offsets backbone-level degradation, while closing the remaining gap will require scaling the visual encoder or adding targeted hand-drawn augmentations.

E.2 Analysis of the English–Chinese Performance Gap

A recurring pattern in Tab. 2 is that most general-purpose models score higher on the Chinese subset than on the English one, while several chart-specialized models show the opposite trend. A natural question is whether this reflects genuine differences in language ability, or differences in the intrinsic difficulty of the two subsets. We investigate this by measuring the structural complexity of each subset and comparing it against the observed score gap.

Table E.1: **Main results on two chart categories under three visual scenarios.** We report the average scores for numeric charts and diagrammatic charts under digital rendering, printed photo, and hand-drawn photo scenarios. The red subscript shows the difference from the first visual scenario within each chart category.

Model Type	Model	Numeric Charts			Diagrammatic charts		
		Digital Rendering	Printed Photo	Hand-drawn Photo	Digital Rendering	Printed Photo	Hand-drawn Photo
General Purpose MLLMs	GPT-4o [53]	36.0	34.3$_{-1.7}$	30.5$_{-5.5}$	48.9	45.0$_{-3.9}$	30.9$_{-18.0}$
	GPT-5 [54]	45.2	43.6$_{-1.6}$	43.1$_{-2.1}$	60.2	52.2$_{-8.0}$	40.9$_{-19.3}$
	InternVL3.5-8B [55]	39.3	34.1$_{-5.2}$	31.7$_{-7.6}$	40.5	38.1$_{-2.4}$	24.0$_{-16.5}$
	InternVL3.5-241B-A28B [55]	45.1	40.7$_{-4.4}$	40.5$_{-4.6}$	55.6	49.6$_{-6.0}$	36.4$_{-19.2}$
	Qwen2.5-VL-7B-Ins. [5]	31.1	27.6$_{-3.5}$	22.4$_{-8.7}$	37.5	36.6$_{-0.9}$	22.0$_{-15.5}$
	Qwen2.5-VL-72B-Ins. [5]	43.0	39.4$_{-3.6}$	37.8$_{-5.2}$	59.9	58.7$_{-1.2}$	40.7$_{-19.2}$
	Qwen3-VL-8B-Ins. [49]	49.5	48.3$_{-1.2}$	46.6$_{-2.9}$	63.1	61.4$_{-1.7}$	47.5$_{-15.6}$
	Qwen3-VL-235B-A22B-Ins. [49]	51.9	49.7$_{-2.2}$	46.3$_{-5.6}$	69.4	65.9$_{-3.5}$	47.4$_{-22.0}$
	Qwen3.5-35B-A3B (think) [56]	61.8	57.0$_{-4.8}$	55.8$_{-6.0}$	74.9	69.1$_{-5.8}$	53.4$_{-21.5}$
	GLM-4.5V [57]	52.8	47.9$_{-4.9}$	46.3$_{-6.5}$	56.8	52.1$_{-4.7}$	37.0$_{-19.8}$
	Seed1.8 (no-think) [58]	47.8	46.8$_{-1.0}$	44.6$_{-3.2}$	66.5	64.6$_{-1.9}$	51.7$_{-14.8}$
	Seed2.0 Pro (no-think) [59]	63.7	59.0$_{-4.7}$	55.5$_{-8.2}$	81.8	75.7$_{-6.1}$	63.3$_{-18.5}$
	Kimi K2.5 (no-think) [60]	61.6	58.2$_{-3.4}$	56.6$_{-5.0}$	76.0	73.5$_{-2.5}$	58.9$_{-17.1}$
	MiMo-V2-Omni [61]	49.3	45.6$_{-3.7}$	44.7$_{-4.6}$	71.5	65.0$_{-6.5}$	50.6$_{-20.9}$
	Gemini 2.5 Pro [62]	54.4	51.1$_{-3.3}$	50.0$_{-4.4}$	70.5	67.2$_{-3.3}$	56.0$_{-14.5}$
	Gemini 3.1 Pro [63]	**67.0**	**62.3**$_{-4.7}$	**60.4**$_{-6.6}$	**83.0**	**78.3**$_{-4.7}$	**64.2**$_{-18.8}$
Doc Parsing MLLMs	dots.mocr (3B) [26]	45.1	41.0$_{-4.1}$	32.3$_{-12.8}$	27.2	23.6$_{-3.6}$	18.2$_{-9.0}$
	PaddleOCR-VL (1B) [18]	50.2	42.3$_{-7.9}$	25.7$_{-24.5}$	–	–	–
	HunyuanOCR (1B) [6]	**56.3**	**49.7**$_{-6.6}$	**37.4**$_{-18.9}$	**55.0**	**50.4**$_{-4.6}$	**26.0**$_{-29.0}$
Chart Experts	ChartAst (13B) [25]	2.6	2.1$_{-0.5}$	1.0$_{-1.6}$	–	–	–
	ChartVLM (8.3B) [1]	9.1	5.6$_{-3.5}$	2.9$_{-6.2}$	–	–	–
	TinyChart (3B) [24]	4.8	3.7$_{-1.1}$	3.6$_{-1.2}$	–	–	–
	ChartMoE (8B) [30]	20.7	15.8$_{-4.9}$	12.2$_{-8.5}$	2.9	2.5$_{-0.4}$	1.7$_{-1.2}$
	ChartCoder (7B) [29]	17.7	13.8$_{-3.9}$	12.0$_{-5.7}$	2.5	1.8$_{-0.7}$	0.6$_{-1.9}$
	RRVF (7B) [28]	46.2	38.3$_{-7.9}$	36.1$_{-10.1}$	59.5	**59.0**$_{-0.5}$	38.5$_{-21.0}$
	MSRL (7B) [27]	36.6	35.6$_{-1.0}$	31.6$_{-5.0}$	24.4	21.4$_{-3.0}$	18.6$_{-5.8}$
	Ours (1B)	**64.1**	**60.4**$_{-3.7}$	**48.6**$_{-15.5}$	**62.5**	58.1$_{-4.4}$	**40.9**$_{-21.6}$

Measuring subset difficulty. For every ground-truth annotation we compute an objective structural-complexity measure that matches the evaluation backend: the number of triples (rows $\times$ (cols $-$ 1)) for numeric charts, and the number of nodes and edges for diagrammatic charts. For each chart type we then compute the relative difficulty gap between the two subsets, $(\mathrm{ZH} - \mathrm{EN})/\mathrm{EN}$, where a positive value means the Chinese subset is structurally harder. As summarized in Tab. E.2, the two subsets are not balanced in difficulty and, importantly, the imbalance runs in opposite directions for the two chart families. The Chinese numeric charts are markedly simpler, with 53.9% fewer triples on bar charts and 49.4% fewer on combination charts, whereas the Chinese diagrammatic charts are more complex, with 18.3% more edges on flowcharts and 30.9% more nodes on mind maps.

Table E.2: **Language score gap versus subset difficulty, per chart type.** *Difficulty gap* is the relative structural-complexity difference, $(\mathrm{ZH} - \mathrm{EN})/\mathrm{EN}$ (measured by #triples for numeric charts, #nodes/#edges for diagrammatic charts; positive means Chinese is harder). *Score gap* is ZH $-$ EN in $\mathrm{mAP}_{\mathrm{high}}$ points averaged per model group (positive means Chinese scores higher). These quantities correlate strongly and negatively ($r = -0.80$ for general-purpose models): models score higher on the structurally easier language subset.

Chart type	Difficulty gap $(\mathrm{ZH} - \mathrm{EN})/\mathrm{EN}$ (%)	Score gap (ZH $-$ EN)		
		General	Doc	Expert
Bar	-53.9	+25.8	+18.9	+5.4
Line	+3.2	+21.5	+15.4	+0.6
Pie	-4.0	+2.2	+9.6	+10.2
Radar	-5.0	+25.0	+19.8	+12.1
Box plot	-11.9	+3.8	+0.3	+1.1
Comb. chart	-49.4	+32.2	+27.0	+12.6
Flowchart	+18.3	-7.2	-4.2	-7.4
Mind map	+30.9	-13.0	-8.8	-8.4

The score gap tracks the difficulty gap rather than language ability. When we place the score gap next to the difficulty gap, the two align almost perfectly in direction. Numeric charts, where the Chinese subset is easier, are exactly where Chinese scores are higher (a general-purpose gap of $+18.4$ points on average), while diagrammatic charts, where the Chinese subset is harder, are where Chinese

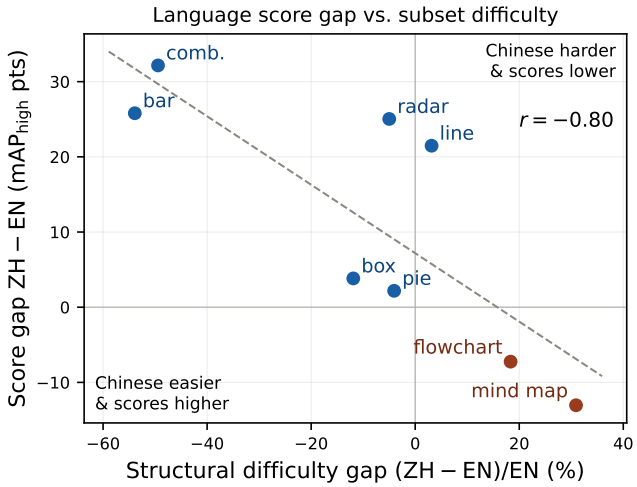


Figure E.1: **Language score gap tracks subset difficulty.** Each point represents a chart type. The x-axis shows the structural difficulty gap, $(\mathrm{ZH} - \mathrm{EN})/\mathrm{EN}$ (positive = Chinese is harder), while the y-axis shows the performance gap, $\mathrm{ZH} - \mathrm{EN}$ in $\mathrm{mAP}_{\mathrm{high}}$ (positive = Chinese scores higher). Numeric charts (blue) cluster where Chinese is easier and scores higher; diagrammatic charts (orange) drop where Chinese is harder and scores lower. The strong negative correlation ($r = -0.80$) confirms this gap is mostly driven by subset difficulty asymmetry.

scores are lower (-10.1 points on average). Across the eight chart types, the general-purpose score gap and the structural difficulty gap are strongly negatively correlated, with a Pearson correlation of $r = -0.80$ (Fig. E.1): models score higher precisely on whichever language subset happens to be structurally easier. This indicates that the English–Chinese gap is largely an artifact of the difficulty asymmetry between the two subsets rather than a pure language-ability effect.

Why specialized models differ. This view also explains the contrasting trends among specialized models. Because general-purpose models cover all chart families, they fully inherit a large performance advantage on easier Chinese numeric charts, which dominate the benchmark. In contrast, chart-specialized models are mostly restricted to numeric charts and trained on English-centric corpora; their margin on the Chinese numeric subset is much smaller ($+7.0$ points vs. $+18.4$ for general models), which can invert the residual gap when accounting for weaker Chinese language priors. The apparent disagreement between the two model groups thus stems from a single underlying cause: the difficulty asymmetry of the subsets, modulated by each model's language prior.

E.3 Fine-grained Analysis on Hard Samples

Aggregate benchmark scores may understate the benefits of CAP reasoning because a large fraction of samples are already well-solved after SFT, yielding identical scores for both settings. To provide a finer-grained view, we isolate a *hard-sample subset* of ChartArena comprising $1{,}055$ samples whose per-sample $\mathrm{mAP}_{\mathrm{high}}$ falls below 0.5 under our SFT baseline *without* CAP (row (b) of Tab. 4). This subset removes saturated samples to isolate cases where CAP reasoning can genuinely make a difference. Following the main evaluation protocol, we assess the SFT-with-CAP setting (row (e)) on both the full and hard subsets, reporting per-sample average $\mathrm{mAP}_{\mathrm{high}}$ in Tab. E.3.

CAP gain grows by roughly $7\times$ on the hard subset. While CAP yields a modest $+2.0$-point average gain across the full benchmark, it boosts performance on the hard subset from 7.1 to 21.5, a net jump of $+14.3$ points. This trend holds across all chart families: numeric charts rise from $+1.1$ to $+14.7$, mind maps from $+2.8$ to $+13.6$, and flowcharts from $+6.4$ to $+13.3$. The modest overall gain simply reflects the SFT baseline saturating on easy samples; once evaluated where headroom remains, CAP reasoning provides substantial, uniform improvements across families.

Implication. This decomposition reveals that the value of CAP concentrates on structurally challenging cases rather than spreading uniformly, explaining why aggregate margins appear modest: they are diluted by easy samples where baseline parsers already succeed. This finer-grained view confirms that lightweight structural reasoning is key to closing the remaining gap on complex real-world charts.

Table E.3: **Fine-grained analysis on hard samples.** Aggregate scores can mask CAP's benefits since easy samples are well solved after SFT. Isolating the *hard-sample subset* of ChartArena (1,055 samples with $\text{mAP}_{\text{high}} < 0.5$ under standard SFT), adding CAP reasoning boosts the baseline from 7.1 to 21.5, a +14.3-point net gain compared to just +2.0 points on the full benchmark. This gain is roughly 7× larger, with every chart family improving more on hard samples (ranging from ∼ 2× for flowcharts to over 13× for numeric charts).

Subset	#Samples	mAP_{high} (×100)		
		SFT w/o CAP	SFT w/ CAP	Δ (CAP gain)
All	2,400	53.42	55.40	+2.0
	Hard (1055)	7.12	21.45	+14.3
Numeric charts	Full	55.75	56.86	+1.1
	Hard (762)	6.06	20.75	+14.7
Diagrammatic charts	Full	46.27	50.91	+4.6
	Hard (293)	9.86	23.28	+13.4
Mind map	Full	53.96	56.74	+2.8
	Hard (124)	10.16	23.71	+13.6
Flowchart	Full	38.97	45.37	+6.4
	Hard (169)	9.64	22.96	+13.3

E.4 Consistency Across Evaluation Metrics

The main paper adopts mAP_{high} as the headline metric. Here we examine whether this choice biases the conclusions by reporting all four metrics in Tab. E.4, and by measuring the pairwise rank correlations among these metrics in Fig. E.2. This analysis directly addresses whether our unified evaluation is robust to the specific tolerance setting used for aggregation.

Model ordering is stable across metrics. The relative ordering of models remains highly consistent across metrics. Every metric correlates strongly with mAP_{high}, showing Spearman correlations of 0.91 (EM), 0.93 ($\text{mAP}_{\text{strict}}$), and 0.96 ($\text{mAP}_{\text{slight}}$) on numeric charts, and 0.95, 0.99, and 0.99 on diagrammatic charts. Agreement is similarly tight within general-purpose models, with all correlations at or above 0.90. The top of the leaderboard is virtually metric-invariant: Gemini 3.1 Pro ranks first across all metrics for both categories, and Seed2.0 Pro remains the leading general-purpose runner-up under tolerance-based metrics. This confirms that our core conclusions are independent of mAP_{high}, and that alternative, cheaper-to-report metrics convey the same qualitative findings.

EM decays faster on diagrammatic charts, while graded metrics decay faster on numeric charts. While model rankings remain stable, metric ratios highlight a clear asymmetry between chart categories. Using metric-to-mAP_{high} ratios to measure sensitivity to stricter criteria, exact match (EM) retains 0.24 of mAP_{high} on numeric charts but only 0.14 on diagrammatic charts. EM penalizes diagrammatic charts far more heavily because a complete flowchart or mind map (including every node label and edge direction) must be recovered perfectly.

Conversely, relaxed metrics decay much more slowly on diagrammatic charts: the $\text{mAP}_{\text{slight}}$-to-$\text{mAP}_{\text{high}}$ ratio is 0.74 compared to 0.55 for numeric charts, a pattern that also holds for $\text{mAP}_{\text{strict}}$ (0.41 vs. 0.30). This shift stems from fundamentally different failure modes. Numeric values are scored element-wise, so tiny per-cell inaccuracies quickly compound under strict tolerances. Graph matching, on the other hand, awards partial credit for recovered subgraphs, letting scores degrade far more gracefully as long as the overall skeleton is intact.

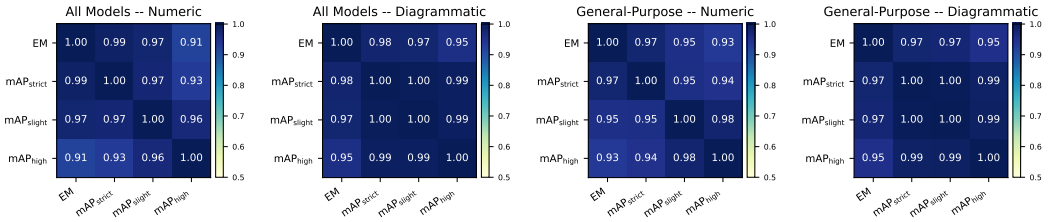
Implications for metric choice. Together, these results justify using mAP_{high} as the main metric: it preserves model rankings while offering a wider dynamic range to better separate models. Meanwhile, divergent decay patterns highlight the value of strict metrics as a complementary signal, exposing a structural recovery gap that is consistently larger for diagrammatic charts than for numeric ones.

E.5 Node/Edge Weight Sensitivity in Flowchart Scoring

The flowchart score in Sec. D.3 combines a node-matching term and an edge-matching term through a weighted sum with the default $(w_E, w_V) = (0.6, 0.4)$. To verify that this choice does not bias the reported comparisons, we re-evaluate all flowchart-capable models under five weightings, $(w_E, w_V) \in \{(0.2, 0.8), (0.4, 0.6), (0.5, 0.5), (0.6, 0.4), (0.8, 0.2)\}$, keeping the rest of the protocol fixed and scoring every model on its Mermaid output. The per-weighting scores and their rank correlations with the default setting are reported in Tab. E.5, and visualized in Fig. E.3.

Table E.4: **Results on two chart categories under four evaluation metrics.** We report **EM** and **mAP** at three tolerance levels (strict, slight, high) for numeric and diagrammatic charts, aggregated over all visual scenarios. Models that lack diagrammatic capability are marked "–". Within each model category, **bold** and underline denote the best and second-best results per column.

Model Type	Model	Numeric Charts				Diagrammatic Charts			
		EM	$\text{mAP}_{\text{strict}}$	$\text{mAP}_{\text{slight}}$	mAP_{high}	EM	$\text{mAP}_{\text{strict}}$	$\text{mAP}_{\text{slight}}$	mAP_{high}
General Purpose MLLMs	GPT-4o [53]	8.5	10.8	19.1	33.6	4.3	15.5	30.3	41.4
	GPT-5 [54]	11.8	14.6	24.9	44.0	7.9	22.7	41.4	50.9
	InternVL3.5-8B [55]	8.1	10.5	19.0	35.0	3.2	10.2	22.1	34.1
	InternVL3.5-241B-A28B [55]	13.2	16.2	26.2	42.1	6.2	17.5	34.7	47.1
	Qwen2.5-VL-7B-Ins. [5]	6.6	8.3	13.6	27.0	3.6	10.5	20.1	32.0
	Qwen2.5-VL-72B-Ins. [5]	8.6	10.6	22.2	40.0	5.6	20.4	39.7	53.1
	Qwen3-VL-8B-Ins. [49]	13.8	16.7	25.8	40.7	6.6	21.9	39.7	53.1
	Qwen3-VL-235B-A22B-Ins. [49]	17.9	21.2	33.6	49.3	9.1	27.6	47.6	60.9
	Qwen3.5-35B-A3B (think) [56]	17.1	21.1	37.5	57.6	9.7	30.9	54.5	66.2
	GLM-4.5V [57]	14.1	17.9	31.9	49.0	6.8	20.1	37.8	49.6
	Seed1.8 (no-think) [58]	14.5	17.5	28.8	46.4	9.3	28.0	49.7	60.9
	Seed2.0 Pro (no-think) [59]	17.8	21.0	38.0	57.7	14.2	41.3	64.8	71.9
	Kimi K2.5 (no-think) [60]	17.6	21.6	37.4	58.8	10.9	36.1	59.9	69.5
	MiMo-V2-Omni [61]	13.5	16.4	28.6	46.5	9.0	28.1	51.4	62.3
	Gemini 2.5 Pro [62]	17.6	20.9	34.5	51.8	11.7	31.2	54.8	64.7
	Gemini 3.1 Pro [63]	**24.5**	**28.7**	**45.4**	**63.2**	**15.8**	**42.8**	**67.6**	**75.2**
Document Parsing MLLMs	dots.mocr (3B) [26]	6.9	**10.6**	21.0	41.6	1.5	6.1	9.9	23.8
	PaddleOCR-VL (1B) [18]	6.3	9.1	17.8	38.3	–	–	–	–
	HunyuanOCR (1B) [6]	**7.4**	10.3	**24.5**	**47.3**	**7.1**	**19.3**	**34.2**	**43.8**
Expert Chart Understanding Models	ChartAst (13B) [25]	0.0	0.0	0.6	1.9	–	–	–	–
	ChartVLM (8.3B) [1]	0.2	0.6	1.8	5.8	–	–	–	–
	TinyChart (3B) [24]	0.1	0.3	0.9	3.9	–	–	–	–
	ChartMoE (8B) [30]	0.7	1.8	6.6	16.0	0.1	0.4	0.8	2.3
	ChartCoder (7B) [29]	1.6	2.7	5.8	14.5	0.0	0.1	0.2	1.6
	RRVF (7B) [28]	**7.8**	**10.8**	**18.7**	**41.3**	**5.9**	**18.4**	**32.7**	**50.3**
	MSRL (7B) [27]	2.6	4.2	9.8	33.9	1.2	4.2	7.8	21.3



(a) All models, numeric (b) All models, diagrammatic (c) General-purpose, numeric (d) General-purpose, diagrammatic

Figure E.2: **Cross-metric rank-correlation heatmaps.** Each panel displays the 4×4 Spearman rank correlation among the four evaluation metrics (EM, $\text{mAP}_{\text{strict}}$, $\text{mAP}_{\text{slight}}$, and mAP_{high}) computed across all models or the general-purpose subset for numeric and diagrammatic charts separately. All off-diagonal correlations are high (≥ 0.90), confirming that model rankings are largely invariant to metric choice.

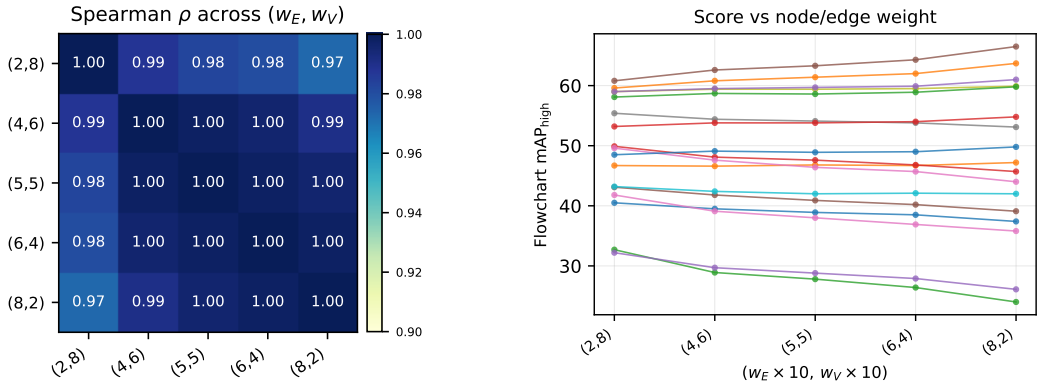
Model ranking is stable across all node/edge weightings. The relative ordering of models barely changes as the weight moves from node-dominated to edge-dominated matching. Every column correlates strongly with the default $(0.6, 0.4)$ ranking, with Spearman $\rho \geq 0.98$ and Kendall $\tau \geq 0.91$ even at the two extremes, and $\rho \geq 0.99$ for all intermediate settings; the pairwise Spearman correlation among any two weightings never drops below 0.97. Gemini 3.1 Pro remains the single best model under all five weightings, and the strong and weak groups stay clearly separated throughout. The choice of (w_E, w_V) therefore does not alter any conclusion drawn from the flowchart results.

The score differences reveal that edges are harder and more discriminative. Although rankings remain stable, score magnitudes shift systematically with weighting based on model capability. Moving weight toward edges raises scores for strong models, such as Gemini 3.1 Pro ($60.8 \rightarrow 66.5$) and Seed2.0 Pro ($59.6 \rightarrow 63.7$), while lowering them for weaker ones, like Qwen2.5-VL-7B ($32.7 \rightarrow 24.0$). Across models, edge-induced gains (the difference between $(0.8, 0.2)$ and $(0.2, 0.8)$ weightings) correlate strongly with average scores ($r = 0.87$). This indicates that recovering edge topology (i.e., directed connectivity in flowcharts) is harder than recognizing isolated nodes, and that edge reconstruction capability is precisely what separates strong parsers from weak ones.

This pattern supports weighting edges more heavily by default. Because edge correctness is both harder to achieve and more discriminative of true parsing ability, a metric that emphasizes edges is more faithful to the practical difficulty of flowchart parsing, where a single misrouted edge can

Table E.5: **Node/edge weight sensitivity on flowcharts.** Flowchart mAP_{high} under five node/edge weightings $(w_E \times 10,\ w_V \times 10)$. The paper default is $(w_E, w_V) = (0.6, 0.4)$. The bottom rows report the Spearman and Kendall rank correlation of each column against the default ranking.

Model	$(2,8)$	$(4,6)$	$(5,5)$	$(6,4)$	$(8,2)$
GPT-4o [53]	40.5	39.5	38.9	38.5	37.4
GPT-5 [54]	46.7	46.6	46.8	46.7	47.2
Qwen2.5-VL-7B-Ins. [5]	32.7	28.9	27.8	26.4	24.0
Qwen2.5-VL-72B-Ins. [5]	49.9	48.1	47.6	46.8	45.7
InternVL3.5-8B [55]	32.2	29.7	28.8	27.9	26.1
InternVL3.5-241B-A28B [55]	43.1	41.8	40.9	40.2	39.1
Qwen3-VL-8B-Ins. [49]	49.6	47.6	46.4	45.7	44.0
Qwen3-VL-235B-A22B-Ins. [49]	55.4	54.4	54.1	53.8	53.1
Qwen3.5-35B-A3B [56]	59.0	59.4	59.4	59.5	59.9
GLM-4.5V [57]	43.2	42.4	42.0	42.1	42.0
Seed1.8 [58]	48.5	49.1	48.9	49.0	49.8
Seed2.0 Pro [59]	59.6	60.8	61.4	62.0	63.7
Kimi K2.5 [60]	58.1	58.7	58.6	58.9	59.8
MiMo-V2-Omni [61]	53.2	53.8	53.8	54.0	54.8
Gemini 2.5 Pro [62]	59.0	59.5	59.7	59.9	61.0
Gemini 3.1 Pro [63]	60.8	62.6	63.3	64.3	66.5
HunyuanOCR [6]	41.8	39.1	38.0	36.9	35.8
Spearman ρ vs. $(6,4)$	0.980	0.995	0.998	1.000	0.998
Kendall τ vs. $(6,4)$	0.908	0.971	0.985	1.000	0.985



(a) Rank correlation across weightings

(b) mAP_{high} vs. node/edge weight

Figure E.3: **Node/edge weight sensitivity for flowchart scoring.** (a) 5×5 Spearman rank correlation among five (w_E, w_V) settings over the same 17 models as Tab. E.5; all off-diagonal values are ≥ 0.97. (b) Flowchart mAP_{high} as a function of the weighting for the same models; the curves rarely cross, confirming a stable ordering, while stronger models improve as more weight shifts to edges.

invalidate the logic of the whole diagram. Our default $(w_E, w_V) = (0.6, 0.4)$ reflects this by placing more weight on edges than on nodes, while remaining well within the stable region identified above. The sensitivity analysis thus confirms both that the reported rankings are robust to the weighting and that the default weighting is a well-motivated design choice rather than an arbitrary one.

E.6 Statistical Significance

To quantify how much of the reported differences could be explained by sampling noise, we run a sample-level bootstrap on the per-sample mAP_{high} scores and report the results in Tab. E.6.

Protocol. To ensure a fair comparison, we enforce a single output format per chart family (Markdown for numeric charts and mind maps; Mermaid for flowcharts) across all 2,400 ChartArena samples. We compute 95% confidence intervals via bootstrapping ($B = 1{,}000$ with replacement, seed 42) using the 2.5th and 97.5th percentiles. For paired comparisons, we apply identical resampled indices to both models to cancel out sample-difficulty variance.

Sampling noise is small relative to the model gaps. The 95% intervals are about 3 to 4 points wide, far narrower than the gaps between performance tiers. Gemini 3.1 Pro forms a tier of its own, and its interval does not overlap that of Kimi K2.5. Our model is statistically indistinguishable from Gemini 2.5 Pro, and its interval clearly separates from that of the HunyuanOCR backbone.

Table E.6: **Capability-conditioned averages and bootstrap 95% confidence intervals.** Capability-conditioned averages exclude chart families marked "–" from the denominator. We report mAP_{high} averaged over all samples, along with 95% intervals from 1,000 sample-level bootstrap resamples. The bottom block presents paired bootstrap intervals on ΔmAP_{high} using identical resampled indices. Bootstrap point estimates use one fixed output format per chart family (Sec. E.6). Bold paired intervals exclude zero.

Model Type	Model	Average (capability)		Bootstrap mAP_{high}	95% CI
		EN	ZH		
General Purpose MLLMs	GPT-4o [53]	34.8	36.3	35.63	[33.97, 37.32]
	GPT-5 [54]	45.6	45.7	45.79	[43.91, 47.60]
	InternVL3.5-8B [55]	29.3	40.2	34.75	[33.05, 36.46]
	InternVL3.5-241B-A28B [55]	38.1	48.6	43.35	[41.60, 45.13]
	Qwen2.5-VL-7B-Ins. [5]	23.3	33.2	28.31	[26.64, 29.92]
	Qwen2.5-VL-72B-Ins. [5]	36.6	50.0	43.27	[41.63, 45.02]
	Qwen3-VL-8B-Ins. [49]	37.3	50.4	43.84	[42.11, 45.69]
	Qwen3-VL-235B-A22B-Ins. [49]	46.0	58.4	52.17	[50.48, 53.98]
	Qwen3.5-35B-A3B (think) [56]	54.1	65.5	59.80	[58.08, 61.51]
	GLM-4.5V [57]	43.8	53.9	48.96	[47.26, 50.80]
	Seed1.8 (no-think) [58]	42.1	57.8	50.00	[48.23, 51.80]
	Seed2.0 Pro (no-think) [59]	54.0	70.5	62.25	[60.55, 63.92]
	Kimi K2.5 (no-think) [60]	54.8	68.1	61.44	[59.64, 63.12]
	MiMo-V2-Omni [61]	43.9	57.0	50.43	[48.60, 52.16]
	Gemini 2.5 Pro [62]	47.7	62.4	55.06	[53.34, 56.83]
	Gemini 3.1 Pro [63]	**59.2**	**73.2**	66.15	[64.59, 67.76]
Document Parsing MLLMs	dots.mocr (3B) [26]	33.1	40.7	–	–
	PaddleOCR-VL (1B) [18]	29.9	46.6	–	–
	HunyuanOCR (1B) [6]	**41.0**	**51.8**	46.51	[44.71, 48.36]
Expert Chart Understanding Models	ChartAst (13B) [25]	1.9	–	–	–
	ChartVLM (8.3B) [1]	7.1	4.6	–	–
	TinyChart (3B) [24]	3.8	4.0	–	–
	ChartMoE (8B) [30]	8.5	22.1	–	–
	ChartCoder (7B) [29]	12.6	13.2	–	–
	RRVF (7B) [28]	36.1	51.0	–	–
	MSRL (7B) [27]	27.1	34.4	–	–
	Ours (1B)	**49.5**	**61.3**	55.44	[53.59, 57.16]

Paired comparison	ΔmAP_{high}	95% CI
Ours (1B) − HunyuanOCR (1B) [6]	+8.93	**[+7.31, +10.29]**
Ours (1B) − SFT w/o CAP, row (b) of Tab. 4	+1.99	**[+0.74, +3.29]**
Ours (1B) − Gemini 3.1 Pro [63]	-10.71	**[-12.52, -9.07]**
Ours (1B) − Kimi K2.5 (no-think) [60]	-6.00	**[-7.72, -4.24]**
Ours (1B) − Gemini 2.5 Pro [62]	+0.37	[-1.53, +2.04]
Ours (1B) − Qwen3-VL-235B-A22B-Ins. [49]	+3.27	**[+1.55, +4.93]**
Gemini 3.1 Pro [63] − Kimi K2.5 (no-think) [60]	+4.71	**[+3.39, +6.22]**

The core claims hold under paired resampling. Our model outperforms the HunyuanOCR backbone by $+8.93$ points ($[+7.31, +10.29]$), while CAP adds $+1.99$ points ($[+0.74, +3.29]$) over the SFT baseline. Because neither interval includes zero, these gains are robust against sampling noise. Meanwhile, the performance gaps relative to Gemini 3.1 Pro and Kimi K2.5 remain significant, aligning with the parameter-scale differences discussed in Sec. E.1.

E.7 Additional Model Results

To improve transparency and broaden comparability across model scales, we evaluate additional models in Tab. E.7. These models include multiple size variants from InternVL3.5 [55] and Qwen3.5 [56], as well as Gemma 4-31B-IT [65], MiniCPM-V 4.5 [66], and the Molmo series [67]. All deployment and evaluation details follow exactly the same setup as the main experiments described in Sec. 6.

F Further Related Work

This section extends the discussion in Sec. 2 by providing background on large language models, specialized chart parsers, and evaluation conventions in neighboring document parsing tasks.

F.1 Large Language and Multimodal Models

The rapid progress of large language models (LLMs) [48, 53, 68–75] has substantially reshaped the landscape of artificial intelligence. A key driver behind this progress is the combination of large-scale pre-training on web-scale text corpora with subsequent supervised fine-tuning (SFT) and preference

Table E.7: **Additional results on ChartArena.** We provide additional results across various model groups and parameter scales to ensure transparency and enable broader comparisons.

Model Group	Model	Release Date	bar		line		pie		radar		box plot		comb.		flowchart		mind map		Average	
			EN	ZH	EN	ZH	EN	ZH	EN	ZH	EN	ZH	EN	ZH	EN	ZH	EN	ZH	EN	ZH
InternVL 3.5	InternVL3.5-1B [55]	2025.08	22.9	40.5	24.5	36.6	51.1	73.3	4.2	18.1	0.5	1.0	4.3	32.7	8.5	4.7	14.3	3.1	16.3	26.2
	InternVL3.5-2B [55]	2025.08	**23.1**	**54.2**	27.3	46.7	63.1	**79.8**	7.4	**35.3**	6.2	7.0	6.9	37.8	16.3	12.2	11.2	2.6	20.2	34.5
	InternVL3.5-4B [55]	2025.08	20.0	50.1	31.5	52.5	65.6	66.9	11.4	34.3	**13.4**	**10.9**	**11.4**	40.0	29.9	**23.8**	**50.1**	**33.8**	29.2	39.0
	InternVL3.5-8B [55]	2025.08	22.7	52.6	**34.4**	**53.7**	**65.8**	73.8	**14.0**	34.7	5.6	9.5	11.3	**42.1**	**32.6**	23.8	48.3	31.8	**29.3**	**40.2**
Qwen2.5 VL	Qwen2.5-VL-3B-Ins. [5]	2025.01	20.4	51.7	32.9	50.7	62.8	78.3	6.0	24.9	1.6	2.7	11.7	41.0	9.0	12.4	26.5	16.9	21.4	34.8
	Qwen2.5-VL-7B-Ins. [5]	2025.01	15.2	36.9	17.9	39.9	63.4	73.1	8.3	19.1	0.9	2.8	6.0	40.6	29.7	23.2	45.4	29.9	23.3	33.2
	Qwen2.5-VL-32B-Ins. [5]	2025.03	**27.4**	**60.1**	25.3	53.4	72.5	**84.7**	**14.0**	**39.2**	8.9	**16.4**	10.6	47.3	41.7	40.1	54.0	40.6	31.8	47.7
	Qwen2.5-VL-72B-Ins. [5]	2025.01	27.1	53.3	**38.2**	**66.7**	**73.5**	77.0	10.9	38.5	**15.0**	15.3	**14.3**	**50.5**	**50.1**	**43.6**	**63.8**	**55.0**	**36.6**	**50.0**
Qwen3 VL	Qwen3-VL-2B-Ins. [49]	2025.10	22.1	45.9	34.1	45.2	70.4	79.1	9.2	35.3	1.9	2.3	10.8	40.0	23.3	20.1	44.1	21.0	27.0	36.1
	Qwen3-VL-2B-Think. [49]	2025.10	21.1	45.9	29.7	48.0	69.1	65.6	8.0	32.1	3.1	1.1	7.3	39.7	30.1	19.1	43.9	18.6	26.5	33.8
	Qwen3-VL-4B-Ins. [49]	2025.10	25.9	52.7	26.3	57.0	75.4	**85.5**	12.3	28.5	15.3	20.0	7.8	47.1	44.7	35.9	57.6	38.1	33.2	45.6
	Qwen3-VL-4B-Think. [49]	2025.10	24.1	49.8	34.3	57.9	73.5	73.7	10.4	35.5	22.1	**29.0**	11.3	44.0	39.1	29.0	62.4	35.7	34.7	44.3
	Qwen3-VL-8B-Ins. [49]	2025.10	27.5	**58.6**	35.5	61.1	77.3	84.7	**16.8**	**42.6**	11.6	12.1	13.2	**47.9**	**50.0**	**41.5**	**66.4**	**54.6**	37.3	**50.4**
	Qwen3-VL-8B-Think. [49]	2025.10	**32.1**	56.5	**42.0**	**64.9**	**79.4**	84.2	14.6	39.3	**22.4**	25.9	**16.0**	46.2	45.7	38.1	64.8	42.3	**39.6**	49.7
Qwen 3.5	Qwen3.5-0.8B [56]	2026.03	16.1	26.0	9.1	21.3	35.6	49.2	3.4	19.3	0.0	0.2	1.8	10.0	6.2	4.3	5.9	0.5	9.8	16.3
	Qwen3.5-2B [56]	2026.03	16.7	33.2	19.6	31.3	46.7	62.7	5.4	20.3	1.5	1.5	2.9	20.1	17.7	11.7	18.7	7.7	16.2	23.6
	Qwen3.5-4B [56]	2026.03	24.2	35.3	35.1	50.1	76.9	**79.6**	14.1	38.0	9.3	8.1	8.8	46.7	39.3	30.3	60.2	39.6	33.5	41.0
	Qwen3.5-9B [56]	2026.03	**32.5**	**45.1**	**45.5**	**54.1**	**82.6**	76.9	**22.0**	**44.8**	**15.3**	**18.1**	**16.8**	**49.5**	**45.5**	**38.1**	**64.2**	**54.5**	**40.6**	**47.6**
Other General MLLMs	Molmo-7B-D [67]	2024.09	2.9	3.3	2.9	0.8	3.6	9.1	1.4	2.7	0.0	0.0	0.0	0.9	0.0	0.0	2.3	0.0	1.6	2.2
	Molmo-72B [67]	2024.09	3.5	2.6	2.5	0.7	6.3	5.5	1.5	6.5	0.7	0.1	0.1	0.7	0.3	0.0	2.1	0.0	2.1	2.0
	MiniCPM-V 4.5 [66]	2025.08	23.3	52.2	34.7	53.5	73.3	79.3	13.0	34.7	**15.3**	**18.0**	14.2	41.1	34.3	29.0	57.4	31.2	33.2	42.4
	Gemma 4-31B-IT [65]	2026.04	**43.7**	**68.7**	**53.4**	**76.4**	**88.9**	**87.2**	**20.9**	**55.5**	9.3	8.9	**22.7**	**59.3**	**59.6**	**44.0**	**76.2**	**61.0**	**46.8**	**57.6**

alignment [37, 76–78], which together endow LLMs with strong reasoning, instruction-following, and emergent generalization abilities [68, 74, 75]. These developments have steadily expanded the range of real-world tasks that language models can reliably handle.

Building on this foundation, vision-language models (VLMs) extend such capabilities to the visual domain by aligning image and text representations in a shared semantic space [79, 80]. More recently, multimodal large language models (MLLMs) integrate a visual encoder with a powerful LLM backbone through cross-modal connectors and visual instruction tuning, achieving strong perception and reasoning over images, documents, and other modalities [5, 6, 49, 55, 57, 81–83]. This convergence of language and vision has turned MLLMs into general-purpose interfaces for visually grounded understanding [84], and motivates their use as the dominant paradigm for chart parsing studied in this work.

F.2 Specialized MLLMs for Chart Parsing

Building on the end-to-end formulation reviewed in Sec. 2, a growing body of specialized chart parsers further adapts general MLLMs to this task, either by instruction tuning on synthetic chart corpora or by introducing chart-specific representations and training objectives [1, 2, 24, 25, 27–30, 85]. Despite these advances, most parsers still learn a direct pixel-to-string mapping and emit results in idiosyncratic output formats, which complicates fair comparison and leaves diagrammatic structures such as flowcharts and mind maps largely underexplored. These observations directly motivate the unified benchmark and format-agnostic protocol introduced in this work.

F.3 Evaluation for Document and Chart Parsing

Standardized evaluation has been a major catalyst of progress in structured document parsing. Closely related tasks have largely converged toward unified evaluation conventions, including table parsing [7–11] and mathematical formula parsing [13–15], where shared output formats and metrics enable direct cross-model comparison. Beyond contemporary documents, parsing historical and ancient materials introduces additional challenges from degraded media, archaic characters, and evolving character forms, and has correspondingly motivated dedicated benchmarks [86, 87].

Chart parsing, by contrast, remains comparatively fragmented, as summarized in Tab. 1. Along the chart-type axis, coverage has expanded only incrementally over time. Early benchmarks such as PlotQA-SE [41] and ChartQA-SE [42] focus on the three most common numeric types (bar, line, and pie), and subsequent efforts add only a few more, such as radar in MMC-Bench [43] and ExChart-Bench [31], box plots in ChartX-SE [1] and VG-DCU [44], or combination charts in ChartY [2] and ParseBench [45]. Crucially, none of these benchmarks include diagrammatic charts such as flowcharts

and mind maps, even though such structures are pervasive in real documents and demand explicit topological reasoning beyond value extraction. The visual-style and language axes are equally narrow: almost all existing benchmarks consist solely of clean digital renderings and overlook real-world conditions such as printed or hand-drawn photos, and only ChartY [2] offers bilingual content while the rest are English-only. Compounded by inconsistent output formats across methods, these gaps make it difficult to assess chart parsers fairly or to evaluate numeric and diagrammatic charts within a single framework. As shown in Tab. 1, ChartArena is the first benchmark to jointly cover all eight chart families, three visual scenarios, and both languages, and is explicitly designed to close this gap under one unified, format-agnostic evaluation protocol.

G Broader Impact

This work contributes to the advancement of general chart parsing and evaluation. By introducing a unified benchmark and evaluation protocol, we aim to support more reliable assessment of multimodal models on structured visual reasoning tasks. While chart understanding may be applied in domains such as scientific analysis, business intelligence, and document accessibility, we have not identified any broader societal impacts that warrant particular concern at this time.

H LLM Usage Statement

LLMs and MLLMs play two distinct roles in this work, and we disclose both.

As part of the method. Model usage that forms part of the research is described in the corresponding method sections. In benchmark construction, multiple MLLMs generate candidate annotations that are merged into a draft and then verified and corrected by human annotators (Sec. 3.1 and Sec. A.2); every released label is human-verified. In training data construction, Qwen3-235B-A22B-Instruct synthesizes the semantic corpora, chart templates, and diagrammatic targets of the synthetic pretraining data (Sec. 4.1), and Qwen3-VL-235B-A22B-Instruct extracts the atomic primitives from which the CAP reasoning traces are built (Sec. 4.3); a human audit of these targets is reported in Sec. C.1. All MLLMs evaluated on ChartArena are listed with their inference setup in Sec. D.5.

In manuscript preparation. LLMs were used only as auxiliary writing tools. Their role was limited to improving language quality, including grammar correction and wording refinement.

I Further Case Study

We provide additional qualitative comparisons in Figs. I.1 and I.2 to further analyze the behavior of different models under challenging visual scenarios. The examples include hand-drawn photos, introducing substantial visual uncertainty for chart parsing.

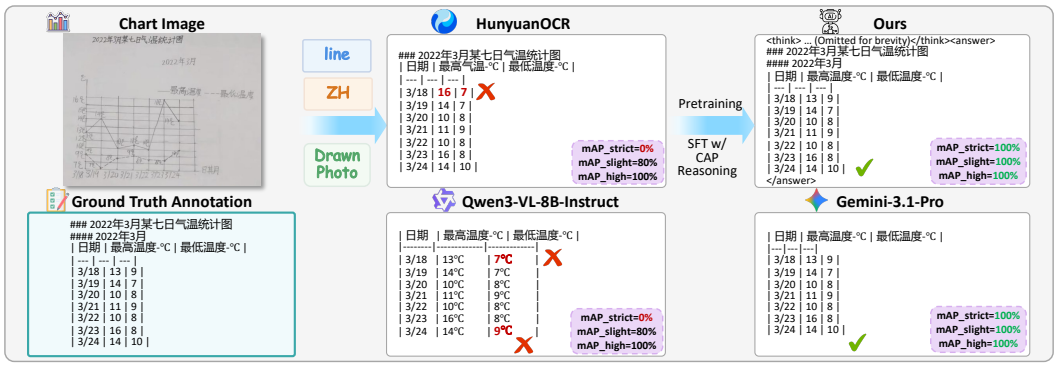


Figure I.1: **Case on hand-drawn line chart.** CAP reasoning improves robustness under challenging visual conditions, enabling more faithful numerical recovery.

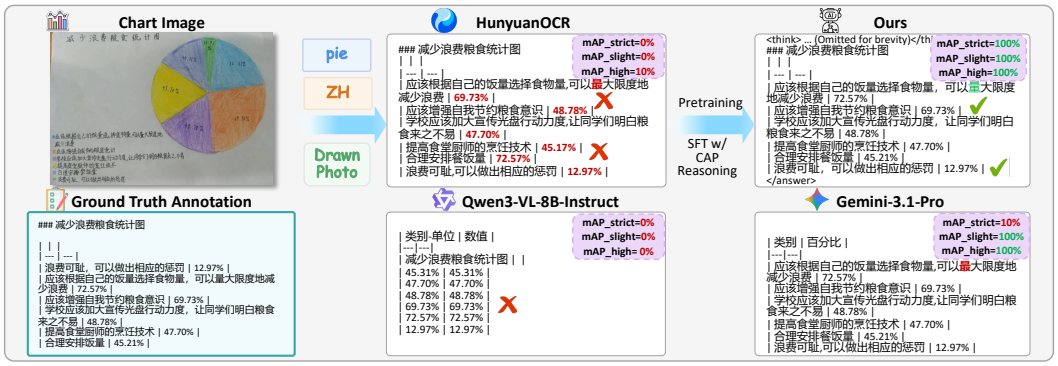


Figure I.2: **Case on hand-drawn pie chart.** CAP reasoning improves robustness under challenging visual conditions, enabling more faithful numerical recovery. Notably, the strong baseline Gemini 3.1 Pro [63] suffers from hallucination, incorrectly recognizing a visually present character as a more plausible alternative, which subsequently leads to parsing errors.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and the contribution list in Sec. 1 state the three contributions (the ChartArena benchmark, the format-agnostic evaluation protocol, and CAP). Each claim is supported by the experiments in Sec. 6, and the analyses in Sec. C and Sec. D test the robustness of these claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sec. 7 discusses the scope of the benchmark (single-page charts, missing chart families such as scatter plots). The remaining robustness gap of our 1B model on hand-drawn numeric charts is analyzed in Sec. E.1.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This work does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The training recipe is given in Sec. 6.1 and Sec. B, including hyperparameters (Tab. B.1), data composition, the CAP extraction prompt, and the reasoning-trace construction. The evaluation protocol, tolerance parameters, and inference prompts are specified in Sec. 3 and Sec. D.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The benchmark and the evaluation code are released at `https://github.com/pspdada/ChartArena`, and the appendix documents the evaluation protocol and the inference setup needed to reproduce the reported results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training data, epochs, learning rates, and hardware are given in Sec. 6.1 and Tab. B.1. Baseline inference settings, prompts, and output formats are described in Sec. D.5, and every ablation states which settings are held fixed.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Sec. E.6 reports sample-level bootstrap 95% confidence intervals ($B = 1{,}000$) for the per-model averages and paired bootstrap intervals for the core comparisons, including our model versus the backbone and the CAP ablation. The sources of variability, the resampling procedure, and the models covered are stated there.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All training and evaluation runs are conducted on 16 NVIDIA H20 GPUs (Sec. 6.1 and Tab. B.1). Proprietary baselines are queried through their official APIs (Sec. D.5).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the Code of Ethics, and the research conforms to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Sec. G discusses the positive uses of reliable chart parsing, such as scientific analysis, business intelligence, and document accessibility, and notes that we have not identified impacts that warrant particular concern.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The released benchmark contains chart images and structured annotations, and the model is a 1B chart parser; neither poses a high risk of misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All existing assets are cited and we use them for non-commercial research in accordance with their licenses and terms of use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new assets are the ChartArena benchmark and the evaluation code. Their construction, annotation protocol, statistics, and evaluation protocol are documented in Sec. 3, Sec. A, and Sec. D, and usage instructions are provided in the public repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Annotation and quality review are carried out by in-house annotators as part of their regular work (Sec. A.2); no crowdsourcing platform is used and no research is conducted on human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs and MLLMs are part of the method: they draft candidate benchmark annotations that are then verified by humans, synthesize the pretraining data, and extract the atomic primitives used to build CAP targets. These uses are described in Sec. 3.1, Sec. 4.1, and Sec. 4.3, and are summarized in Sec. H.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.