Code as Representation: A Compilable Parsing Paradigm for Academic Documents

Rihui Jin*
Jun Wang*
Southeast University
School of Computer Science and Engineering
Nanjing, Jiangsu, China
ari_king@seu.edu.cn
213211446@seu.edu.cn

Chengyuan Zhu
Liang Mingyu
Yue Gao
Li Yunxuan
Southeast University
School of Computer Science and Engineering
Nanjing, Jiangsu, China

Kuicai Dong
Nanyang Technological University
Singapore, Singapore
kuicai001@e.ntu.edu.sg

Guilin Qi†
Southeast University
School of Computer Science and Engineering
Nanjing University
State Key Laboratory for Novel Software Technology
Nanjing, Jiangsu, China
gqi@seu.edu.cn

Lin Ren
Yongrui Chen
Xinbang Dai
Jiaqi Li
Southeast University
School of Computer Science and Engineering
Nanjing, Jiangsu, China

Tongtong Wu
Gholamreza Haffari
Monash University
Clayton, Victoria, Australia
tongtong.wu@monash.edu
gholamreza.haffari@monash.edu

Abstract

Academic papers are a primary carrier of scientific knowledge, yet most of this knowledge remains locked in PDFs that are optimized for human reading rather than machine use. For Multimodal Large Language Models (MLLMs), the core challenge is not only perception, but representation: scientific pages interleave text with Structured Academic Elements (SAEs) such as tables, formulas, charts, and pseudocode, whose structure, data, and logic are poorly preserved by common surrogates like Markdown. We therefore propose Compilable Academic Document Parsing (CADP), a paradigm that reconstructs a full page as contextual LaTeX plus executable Python, so that structure-preserving elements and executable chart representations can be reconstructed, recompiled, and directly verified against the source page. To support this setting, we introduce CADP-Bench, an expert-verified benchmark of full academic pages containing tightly coupled text and multiple SAE types, evaluated through a re-injection compilation protocol. We further study current capabilities using SOTA MLLMs and an exploratory multi-agent baseline that incorporates common agentic techniques. Results show that even frontier models still struggle to produce high-fidelity executable reconstructions, highlighting substantial room for improvement in structure-aware scientific document parsing. CADP-Bench is released for future research. [1]

CCS Concepts

• **Computing methodologies** → **Computer vision**; **Natural language processing**; • **General and reference** → **Evaluation**.

Keywords

Multimodal Document Parsing, MLLMs

ACM Reference Format:
Rihui Jin, Jun Wang, Chengyuan Zhu, Liang Mingyu, Yue Gao, Li Yunxuan, Kuicai Dong, Guilin Qi, Lin Ren, Yongrui Chen, Xinbang Dai, Jiaqi Li, Tongtong Wu, and Gholamreza Haffari. 2026. Code as Representation: A Compilable Parsing Paradigm for Academic Documents. In *Proceedings of the 34th ACM International Conference on Multimedia (MM '26), November 10–14, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3767308.3836397

1 Introduction

Academic papers are a primary carrier of scientific knowledge, yet most of this knowledge remains locked within PDFs optimized for human reading rather than machine use [7, 13, 31]. As Multimodal Large Language Models (MLLMs) are increasingly deployed in retrieval, reasoning, and AI-for-science pipelines [1, 6, 18, 27], converting scientific pages into machine-actionable representations has become a central bottleneck [9, 10]. This challenge is especially acute for academic documents, where continuous prose is tightly interleaved with tables, formulas, charts, and pseudocode, which

*These authors contributed equally to this work.
†Corresponding author.

MM '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2213-4/2026/11
https://doi.org/10.1145/3767308.3836397

[1] https://github.com/AriKing11/CADP-Bench

we collectively term Structured Academic Elements (SAEs). Recovering such pages at full fidelity is therefore not only a perception problem, but fundamentally a representation problem.

Downstream AI systems rarely access a paper as editable source; they usually encounter only PDFs or screenshots [2, 5]. In this process, SAE structure becomes implicit: topology, alignment, nesting, and chart data are no longer available as symbolic objects that can be directly edited, executed, or verified. Practical pipelines therefore translate page images into machine-actionable surrogates, most commonly Markdown, because it is lightweight and LLM-friendly [14, 21]. However, Markdown is merely a convenient surface format, not a faithful representation of complex scientific pages. It inherently suffers from three critical limitations (see Fig. 1): *structural collapse*, as its flat syntax cannot cleanly encode merged tables [16], aligned equations, or complex pseudocode [23]; *chart opacity*, as charts are reduced to static cropped images that discard underlying data and rendering logic; and *non-verifiability*, as the parsed flat text cannot be recompiled to visually validate its fidelity against the original document. Thus, this motivates a different question: *can we recover from page pixels a representation that preserves structure, exposes chart data and rendering logic, and can be compiled back into a faithful page rendering, thereby serving as a machine-actionable substitute for the screenshot itself?*

We argue that it can, and advocate a compilable parsing paradigm that reframes parsing from pixels to programs. We instantiate this idea as the **Compilable Academic Document Parsing** (CADP) task: given a raw full-page screenshot and limited compilation context, the model jointly generates contextual LaTeX for continuous text and SAEs, and executable Python for the data and rendering logic behind charts. This dual-code formulation preserves nested structure, makes chart internals recoverable, and, crucially, is directly verifiable: generated code can be re-injected into the original document environment, compiled, rendered, and visually compared against the source page.

Despite growing interest in document parsing (DP), existing benchmarks cannot evaluate this paradigm. Markdown-based benchmarks [20, 36] inherit the ceiling of flat representations, while element-level code generation [29] evaluates isolated regions without page context, creating "semantic orphans" that lack cross-references and narrative grounding (see Table 1) [28, 38]. To bridge this gap, we introduce CADP-Bench, an expert-verified benchmark for compilable academic parsing. Each sample is a full academic page in which main text is tightly coupled with at least two SAE types, and evaluation follows a re-injection compilation protocol that compares the rendered page with the ground truth.

Using CADP-Bench, we evaluate SOTA MLLMs and an exploratory multi-agent baseline designed to probe their performance upper bounds. Results show that even with agentic scaffolding, current models still struggle with high-fidelity executable reconstruction, leaving substantial room for improvement.

In summary, the main contributions of this paper are as follows:

- We formalize CADP, a new paradigm reconstructing pages as contextual LaTeX and Python, shifting parsing to structure-preserving, dual-code generation under compilation constraints.
- We introduce CADP-Bench, an expert-verified benchmark operationalizing this paradigm with multi-SAE pages, rigorously evaluated via a novel re-injection compilation protocol.
- We benchmark SOTA MLLMs alongside an exploratory agentic baseline, and further demonstrate the dataset's broader utility for evaluating format-sensitive comprehension. Results expose key bottlenecks, confirming that high-fidelity reconstruction remains a formidable challenge.

2 Related Work

2.1 Multimodal Document Parsing

Recent advancements in document digitization have been driven by both robust industrial multi-stage pipelines and end-to-end generative models [19, 24, 26, 34, 35]. However, existing document parsing pipelines remain largely text-centric. Even when they incorporate localized HTML or LaTeX to represent complex elements, their primary output relies on flat Markdown [14, 20], treating non-textual graphical regions as inert raster crops. Consequently, much of the structural and semantic information encoded in these visuals is discarded, making the page reconstruction process inherently lossy and limiting the fidelity of the output.

Recognizing the limitations of character-level transcripts, a recent trend has shifted towards recovering document elements as structured, executable code. Specialized models now translate isolated mathematical formulas into strict and compilable LaTeX [15, 22], reverse-engineer hierarchical tables into verifiable code [8, 17, 23], or transpose data charts into programmatic Python scripts [3, 4, 12, 37]. Despite this progress, these code-generation approaches are severely constrained by an isolated cropping paradigm. When presented with a full screenshot of a complex academic paper, they fail to generate the comprehensive, compilable code necessary to faithfully reconstruct the original page.

2.2 Benchmarks for Multimodal Document Parsing

Current DP benchmarks predominantly follow two paradigms(see Table 1, each constrained by systematic limitations. The first paradigm focuses on flat text extraction, typically adopting Markdown as the main target representation while relegating charts and complex tables to mere links referencing cropped screenshots [20, 25, 36]. By treating information-dense scientific charts as opaque pixels, these benchmarks fail to evaluate a model's capacity to recover underlying data arrays or logical rendering processes, resulting in a profound loss of computational utility. The second paradigm targets structured code generation for specific elements [29, 32, 33], yet it relies exclusively on localized, pre-cropped image inputs. Evaluating these elements in strict isolation creates "semantic orphans." This localized approach fails to assess how effectively a model grounds a chart or algorithm within the broader full-page context, nor does it measure the model's ability to resolve crucial in-text cross-references.

3 The CADP-Bench Benchmark

Below, we formalize the task (§3.1), detail dataset construction (§3.2, §3.3), describe the evaluation protocol and metrics (§3.4, §3.5), and present benchmark statistics and quality assurance (§3.6).

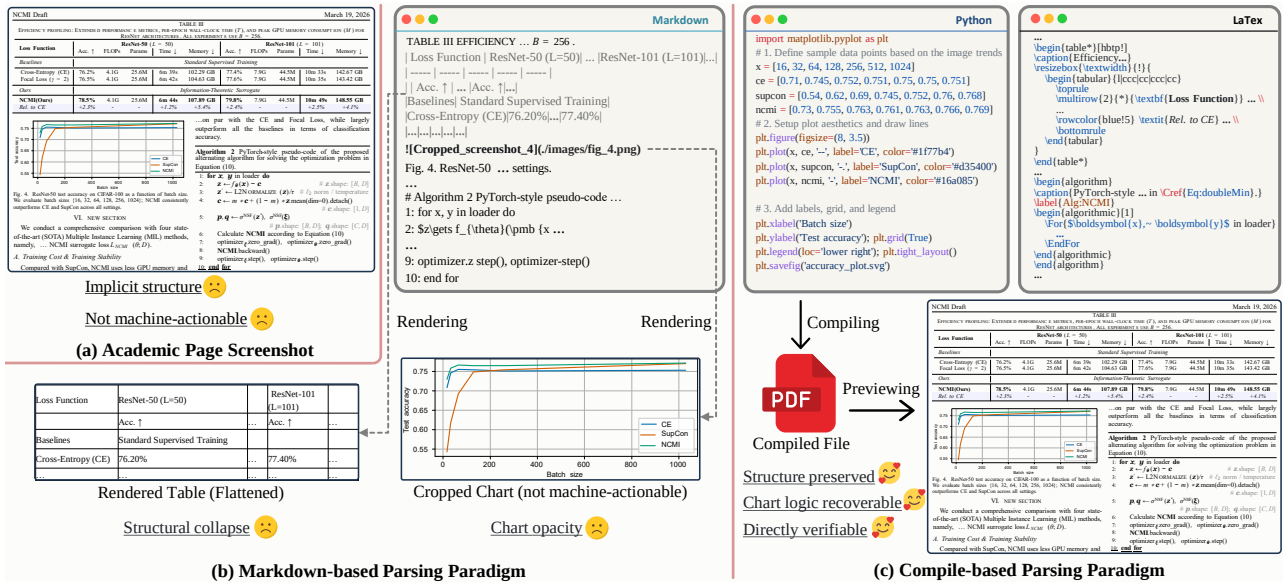

Figure 1: Comparison of three representations for machine reading of academic papers. (a) A raw page screenshot preserves the full visual layout but leaves structure implicit. (b) Markdown-based parsing linearizes the page into plain text, flattened tables, and raster chart crops, leading to structural collapse and opaque visual elements. (c) Our compilable parsing paradigm reconstructs the same page as contextual LaTeX plus executable Python, which can be compiled back into a preview PDF, preserving document structure, recovering chart logic, and enabling direct verification.

Benchmarks	Input Type					Output Type			Re-renderable Reconstruction	#Samples	Annotation Type
	Page	Table	Chart	Formula	Pseudocode	Markdown	Python	LaTeX			
TabLeX[8]	✗	✓	✗	✗	✗	✗	✗	✓	✓	4M+	Auto-extracted
Tab-To-Tex[17]	✗	✓	✗	✗	✗	✗	✗	✓	✓	40k+	Auto-extracted
TAB2LATEX[15]	✗	✓	✗	✓	✗	✗	✗	✓	✓	5,000	Auto-extracted
Table2LaTeX[23]	✗	✓	✗	✗	✗	✗	✗	✓	✓	1,211	Auto-extracted
Im2LaTeX-100K [22]	✗	✗	✗	✓	✗	✗	✗	✓	✓	100K	Auto-extracted
UniMER [30]	✗	✗	✗	✓	✗	✗	✗	✓	✓	20K+	Human
ChartMimic[33]	✗	✗	✓	✗	✗	✗	✓	✗	✓	4,800	Human
ChartX[32]	✗	✗	✓	✗	✗	✗	✓	✗	✓	6,000	LLM + Human
ChartEdit[37]	✗	✗	✓	✗	✗	✗	✓	✗	✓	1,405	LLM + Human
DaTikZ$_{v3}$[3]	✗	✗	✓	✗	✗	✗	✗	✓	✓	1,000	Auto + Human
OmniDocBench[25]	✓	✓	✓	✓	✗	✓	✗	√†	✗	1,355	LLM + Human
olmOCR-bench[26]	✓	✓	✓	✓	✗	✓	✗	✗	✗	1,402	LLM
READOC[20]	✓	✓	✓	✓	✗	✓	✗	✗	✗	3,576	Auto-extracted
CADP-Bench (Ours)	✓	✓	✓	✓	✓	✗	✓	✓	✓	1,630	LLM + Human

Table 1: Comparison of representative document parsing benchmarks across key dimensions. ✓: fully matches this category; √†: partially matches this category; ✗: does not match this category.

3.1 Task Formulation

3.1.1 Input Space and Contextual Environment. Let $\mathcal{D}$ denote a complete academic document. For a target page p in $\mathcal{D}$, let $I_p \in \mathbb{R}^{H\times W\times 3}$ denote its page image.

For re-injection compilation and evaluation, we retain a *Source Context* $C_{\setminus p}$, i.e., the original LaTeX source with the content of page p excised. It preserves the document-level environment, including the preamble, custom macros, references, bibliography, and surrounding text, and is used only to re-inject the generated content and verify compilation consistency.

Importantly, $C_{\setminus p}$ is not provided to the model during generation. The model receives only the target page image I_p and a limited compilation context C_{pkg} containing the available package declarations from the document template, while additional packages may be specified when necessary. Thus, CADP is a page-level reconstruction task under limited compilation context, rather than full source-code recovery.

3.1.2 Output Space: Dual-Code Generation. The objective is to learn $f_\theta : (I_p, C_{\text{pkg}}) \to (\mathcal{L}_p, \mathcal{P}_p)$, where the outputs jointly reconstruct the target page:

- **Contextual LaTeX Block ($\mathcal{L}_p$):** A LaTeX body block that reconstructs the page text and structured elements, including tables, formulas, and pseudocode.

- **Executable Python Programs ($\mathcal{P}_p$):** A set of Python programs $\mathcal{P}_p = \{P_1, P_2, \ldots, P_k\}$ for reconstructing the k charts on page p. Rather than recovering the original authors' plotting programs or unique raw data, these programs encode visually inferred quantitative values and rendering logic, providing an executable representation that can be rendered and verified.

3.2 Data Collection and Preparation

As illustrated in Fig. 2, the construction of CADP-Bench begins with large-scale data acquisition from the arXiv repository, covering a diverse set of academic disciplines. Both PDF documents and their corresponding LaTeX source files are collected to ensure comprehensive access to structural and textual information. To filter PDFs, we employ the Mineru framework [24], which extracts page-level layout information and identifies individual elements such as tables, charts, formulas, and pseudocode. Simultaneously, the collected LaTeX source fragments are consolidated into complete, compilable LaTeX files. Any LaTeX files that fail to compile are discarded to maintain dataset integrity.

To ensure that CADP-Bench presents a sufficiently challenging benchmark aligned with the multi-SAE coupling principle, we filter for pages containing multiple SAEs. Specifically, using the parsing results from Mineru, we retain only those pages in which at least two distinct SAE categories coexist. Pages that do not meet this criterion are excluded from the benchmark. If a single table spans two consecutive pages, we still collect this page pair as one sample even when a table is the only SAE type on those pages. This setting preserves long-range structural dependencies and evaluates whether models can recover complete tabular content under cross-page continuity.

Finally, for the selected pages, we extract the ground-truth LaTeX code corresponding to all included SAEs. This extraction leverages both regular expression matching and the Mineru parsing results, ensuring precise alignment between the visual elements in the PDF and their corresponding structural representations in LaTeX. The resulting dataset provides a robust foundation for evaluating full-page, multi-element reconstruction models.

3.3 Chart Reconstruction and Annotation

While tables, formulas, and pseudocode are inherently represented in LaTeX source files, charts present a unique challenge: the original LaTeX source typically does not contain any code that reproduces the chart, only a reference to a pre-rendered image file. This gap motivates the dual-code design of our paradigm—to achieve fully compilable reconstruction, chart data and rendering logic must be recovered as executable Python programs.

For each page retained in the benchmark, any embedded chart is first cropped and saved as a PNG image. This image is then input to a Chart2Code model—specifically, Gemini-3-Pro—which produces Python code capable of reproducing the chart. The generated Python code is subsequently compiled to render a reconstructed chart in SVG format. To ensure the accuracy and reliability of the dataset, the reconstructed chart replaces the original image only after expert verification. Human annotators review the rendered chart to confirm that it faithfully reflects the original visual information, mitigating potential discrepancies introduced by the automated Chart2Code generation. The verified Python code serves as the ground-truth annotation, and the reconstructed chart guarantees consistency between visual representation and executable logic.

Additionally, both automated and expert-guided annotation procedures are employed to assign difficulty levels—simple, medium, or hard—to each page and its constituent SAEs. The difficulty is determined based on factors such as the length of the LaTeX code and the structural complexity of the contained elements. Detailed criteria for this classification are provided in the Appendix.

3.4 Evaluation Protocol: Re-injection Compilation

The CADP enables an evaluation protocol that goes beyond surface-level text comparison: we test whether generated code can reproduce the original page through compilation. Concretely, the evaluation enforces strict structural, visual, and executable consistency through a *Re-injection Compilation* mechanism.

First, each generated Python program $P_i \in \mathcal{P}_p$ is executed in an isolated environment to synthesize a SVG file G_i:

$$G_i = \text{Execute}(P_i), \quad \forall i \in \{1, \ldots, k\} \tag{1}$$

The generated LaTeX block $\mathcal{L}_p$ must explicitly reference these generated assets (e.g., via `\includegraphics{...}`).

Subsequently, $\mathcal{L}_p$ is structurally re-injected into the excision point of the source context $C_{\setminus p}$, which provides the original preamble and surrounding document text. The complete, restored document is then processed by the LaTeX compiler Ω:

$$\hat{I}_p = \Omega\big(C_{\setminus p} \oplus \mathcal{L}_p,\ \{G_1, \ldots, G_k\}\big) \tag{2}$$

where $\oplus$ denotes in-place sequence concatenation, and $\hat{I}_p$ is the newly rendered target page. This process ensures that compiling the document reproduces the page content corresponding exactly to the input screenshot. Any hallucination of custom macros, failure to close nested environments, or incorrect data logic in Python will directly precipitate a compilation failure or severe visual misalignment, providing an inherently stringent fidelity check that is unique to the compilable parsing paradigm.

3.5 Evaluation Metrics

Because code is non-unique representation, we evaluate both *code-level* correctness and *visual-level* fidelity. A full summary is given in Table 2. Metrics not redefined here follow Mineru [24]. Any sample that fails re-injection compilation is assigned a score of 0.

Pseudocode Reconstruction Score. We define *Pseudocode Reconstruction Score* (PRS) to jointly measure textual and control-flow fidelity. Ground-truth and predicted pseudocode are converted into ordered step sequences S^{gt} and S^{pred} (lengths n and m), with step tokens s_i^{gt} and s_i^{pred}. Preprocessing removes pseudocode tags (e.g., `\State`), normalizes case/whitespace, and strips trailing punctuation. Steps are aligned strictly by order: step i is compared only to step i; if $i > m$, set $s_i^{\text{pred}} = \emptyset$; predicted steps beyond n are ignored. Text fidelity is computed by normalized Levenshtein similarity:

$$R_{\text{text}} = \frac{1}{n} \sum_{i=1}^{n} \left(1 - \frac{\text{EditDist}(s_i^{\text{gt}}, s_i^{\text{pred}})}{\max(|s_i^{\text{gt}}|, |s_i^{\text{pred}}|)} \right). \tag{3}$$

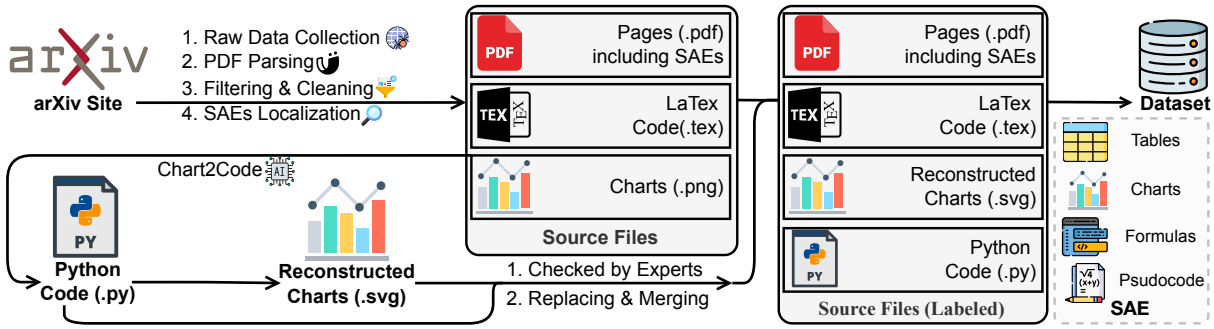

Figure 2: Construction workflow of CADP-Bench. We collect and filter arXiv source files to identify pages containing dense SAEs. Gemini-3-Pro generates candidate Python programs from raster charts, which are rendered as SVGs and retained only after expert verification of their visual content, data trends, and layout. The verified programs serve as executable chart references rather than the original plotting code. Together with the corresponding LaTeX source, they form the final benchmark.

For structure fidelity independent of wording, we parse each pseudocode block into a simplified AST: the root is `ALGORITHM`; control-flow commands (`\For`, `\While`, `\If`, `\Else`, `\Return`) are mapped to normalized structural nodes; and all other statements collapse into a generic `STEP`, ignoring variable names and conditions. Let T^{gt} and T^{pred} be the resulting trees. Structure fidelity is measured by normalized tree-edit similarity:

$$R_{\text{struct}} = \max\left(0,\ 1 - \frac{\text{TED}(T^{\text{gt}}, T^{\text{pred}})}{\max(|T^{\text{gt}}|, |T^{\text{pred}}|)}\right). \tag{4}$$

The final PRS averages the two components and rescales to $[0, 100]$:

$$\text{PRS} = 50\,(R_{\text{text}} + R_{\text{struct}})\,. \tag{5}$$

Visual Reconstruction Fidelity (VRF). For holistic visual quality, we introduce an MLLM-as-a-judge [11] metric termed *Visual Reconstruction Fidelity* (VRF) to evaluate layouts, tables, charts, and pseudocode (formulas are evaluated by separate code-level metrics). Each target type is evaluated via a tailored prompt with K dimensions (e.g., completeness, structure, alignment, and visual fidelity), and each dimension is scored on a five-level ordinal scale $s_k \in \{0, 1, 2, 3, 4\}$. Gemini-3-Pro serves as the evaluator, having achieved an 86% exact agreement rate against human ratings on 100 validation samples. To further assess evaluator dependence, we re-evaluate Chart VRF-A using GPT-5.5. The resulting scores differ from the Gemini-3-Pro-based evaluation by less than 3%, while preserving the overall performance trend and Gemini-3-Pro as the best-performing model.

To enforce rigorous evaluation and prevent partial scores from overly inflating results on inherently structured elements, we employ a strict variant, *VRF-S*, for tables and pseudocode. A sample scores 100 if and only if all dimensions receive a perfect mark; otherwise, it scores 0:

$$\text{VRF-S}(x) = \begin{cases} 100, & \text{if } s_k = 4 \text{ for all } k \in \{1, \ldots, K\} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Conversely, for charts and global layout—where deterministic correctness is less binary—we define a continuous variant, *VRF-A*, by averaging the individual dimensions:

$$\text{VRF-A}(x) = 25 \cdot \frac{1}{K} \sum_{k=1}^{K} s_k \tag{7}$$

Type	Metric	Notes
Code	Exec. Rate	Layout/Chart: re-injected LaTeX and Python must be able to be compiled and rendered.
	TEDS	Table: structure&content similarity.
	PRS	Pseudocode: combines text and structure fidelity.
Visual	Reading Order	Layout: reading-order error (lower is better).
	PageIoU	Layout: page-level overlap between prediction and ground truth.
	CDM	Formula: character-level matching score.
	VRF	Layout/Table/Charts/Pseudocode: an MLLM-as-a-judge based score.
	Pixel Sim.	Global pixel similarity between rendered and ground-truth pages.

Table 2: Summary of the evaluation metrics.

The final benchmark score for a given category is obtained by averaging either VRF-S(x) or VRF-A(x) over all samples in the dataset, effectively guaranteeing precise evaluation while retaining flexibility where continuous matching is necessary.

Pixel Similarity. Let G be the ground-truth target page and B the blank template. If the predicted PDF renders to T pages, denoted by $\hat{P}^{(t)}$ for $t = 1, \ldots, T$, we extend the single-page ground truth to

$$\tilde{G}^{(1)} = G, \qquad \tilde{G}^{(t)} = B \text{ for } t = 2, \ldots, T. \tag{8}$$

For each page t, we ignore co-background pixels and define

$$\mathcal{V}^{(t)} = \{(i, j) : \tilde{G}_{ij}^{(t)} \neq B_{ij} \ \vee \ \hat{P}_{ij}^{(t)} \neq B_{ij}\}. \tag{9}$$

Then

$$\Delta^{(t)} = \frac{1}{3|\mathcal{V}^{(t)}|} \sum_{(i,j)\in\mathcal{V}^{(t)}} \sum_{c\in\{R,G,B\}} \left|\tilde{G}_{ijc}^{(t)} - \hat{P}_{ijc}^{(t)}\right|, \tag{10}$$

$$\text{MAD}^{(t)} = \begin{cases} 0, & |\mathcal{V}^{(t)}| = 0, \\ \Delta^{(t)}, & |\mathcal{V}^{(t)}| > 0, \end{cases} \qquad \text{PS}^{(t)} = 1 - \frac{\text{MAD}^{(t)}}{255}. \tag{11}$$

The sample-level score is

$$\text{PS} = \frac{1}{T} \sum_{t=1}^{T} \text{PS}^{(t)}. \tag{12}$$

Sub-domain	Computer Science	Physics	Economics	Quantitative Biology	Statistics	Total
Count	1047	251	45	222	65	**1630**
Element	**Simple**	**Medium**		**Hard**		**Count**
Table	803	644		395		1842
Formula	340	536		147		1023
Pseudocode	40	71		27		138
Chart	316	975		200		1491

Table 3: Statistical overview of CADP-Bench, including total scale, per-sub-domain sample counts, and element-level difficulty distribution. Difficulty levels are annotated as *Simple*, *Medium*, and *Hard*.

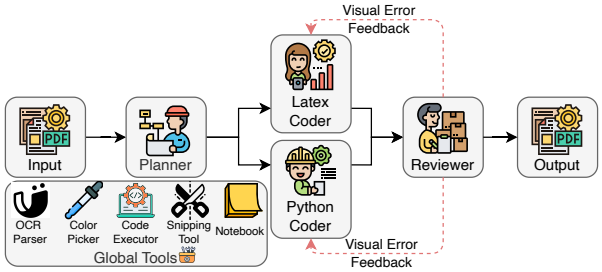

Figure 3: An exploratory multi-agent system used to study common agentic techniques on CADP-Bench.

3.6 Benchmark Statistics and Quality Assurance

We summarize CADP-Bench from four complementary perspectives: overall scale, sub-domain coverage, difficulty-aware element distribution, and annotation quality.

Scale and Coverage. As shown in Table 3, we report the total number of samples, the sample counts across five sub-domains, and the counts of four core SAEs under three difficulty levels. Every sample contains at least two co-occurring SAE types, ensuring consistent layout complexity across the benchmark.

Annotation Quality Control. To ensure high-fidelity ground truth, annotation and quality control were conducted by four CS undergraduate co-authors proficient in LaTeX and Python. Each sample was independently checked by two annotators, with disagreements adjudicated by a CS PhD candidate. For each element type, we report the reviewer agreement rate to reflect inter-annotator consistency; the overall agreement rate is 88%. Samples that do not satisfy strict alignment between the visual PDF content and the corresponding LaTeX/Python ground truth are revised and re-checked before inclusion.

4 Experiments & Analysis

4.1 Experimental Setup

4.1.1 Evaluated Models. We evaluate a set of SOTA MLLMs that represent leading approaches to full-page scientific document understanding and parsing.

4.1.2 Exploratory Multi-Agent Baseline. To study whether commonly used agentic techniques are helpful on CADP-Bench, we implement an exploratory multi-agent system. As shown in Fig. 3, it combines four common ingredients in agentic workflows: task decomposition, role specialization, shared tools, and iterative feedback. The system contains four agents: a Planner, a LaTeX Coder, a Python Coder, and a Reviewer. Given a page image, the Planner first parses the global layout and decomposes reconstruction into text, formula, table, and chart sub-tasks; the LaTeX Coder writes the page layout, text, and math; the Python Coder reconstructs chart data and plotting logic; and the Reviewer compares the compiled output with the source page and returns targeted visual feedback for another round of revision. All agents share a small tool set commonly used in document-centered agents: an OCR Parser for text and boxes, a Color Picker for chart colors, a Code Executor for LaTeX/Python validation, a Snipping Tool for local crops, and a shared Notebook for intermediate notes and parameters.

4.1.3 Implementation Details. To reduce variability, each sample is evaluated three times, and the reported metrics are averaged across runs. All models are accessed through their officially provided APIs. The model temperature is set to 0.7. All LaTeX compilations are performed using the pdfLaTex engine to ensure consistent rendering of complex structures and non-standard fonts.

4.2 Overview

- **RQ1 (End-to-End Performance):** How effectively do current SOTA MLLMs perform on CADP-Bench?
- **RQ2 (Complexity & Robustness):** How does structural difficulty, at both element and page levels, affect model reconstruction fidelity?
- **RQ3 (Impact of Agent Techniques):** How do agent-related strategies—including self-reflection, environmental feedback, and multi-agent collaboration—affect model performance?
- **RQ4 (Format-Sensitive Comprehension):** How does input format (screenshots, Markdown, or LaTeX) influence a model's ability to answer context-dependent questions requiring information from one or more SAEs?

4.3 Main Results (RQ1 & RQ2)

4.3.1 E2E Performance (RQ1). From Table 4, we summarize two key findings from the dual-code evaluation: 1) From a capability perspective, while open-weight models are rapidly closing the gap with proprietary systems, complex structural reasoning remains a pervasive bottleneck. Gemini-3-Pro stands out as the top performer, yet its sub-optimal VRF layout and Pixel Sim. scores show that flawless high-fidelity generation remains challenging. Notably, the open-weight Qwen3.5-397B-A17B achieves overall visual fidelity rivaling commercial counterparts like GPT-5.4 and Claude Opus 4.6. However, structurally dense elements like pseudocode expose severe vulnerabilities across all capability tiers. While Gemini-3-Pro maintains a lead (56.52 VRF-S), other strong frontier models (e.g., GPT-5.4 at 13.04, Claude Opus 4.6 at 17.39) suffer dramatic performance collapses, underscoring that robust algorithmic formatting is a universally unsolved challenge for current MLLMs. 2) From a metric perspective, code executability and visual fidelity reveal a systematic decoupling in chart reconstruction. Frontier models achieve exceptionally high execution rates, yet their VRF-A scores lag significantly behind, exposing a fundamental gap: generating

Models	Formula	Table		Chart		Pseudocode		Layout				Overall
	Visual	Code	Visual	Code	Visual	Code	Visual	Code	Visual			Visual
	CDM↑	TEDS↑	VRF-S↑	Exec. Rate↑	VRF-A↑	PRS↑	VRF-S↑	Exec. Rate↑	Reading Order↓	PageIoU↑	VRF-A↑	Pixel Sim.↑
Claude Haiku 4.5	27.71	44.10	9.46	49.57	4.75	10.82	0.00	42.33	75.50	23.17	25.75	20.32
Claude Sonnet 4.6	52.10	72.80	38.28	93.59	27.75	47.74	21.74	86.33	46.03	46.34	50.50	41.48
Claude Opus 4.6	60.61	71.32	35.43	78.63	35.00	49.50	17.39	84.00	49.97	49.45	48.50	39.76
Qwen3.5-35B-A3B	63.69	72.35	16.50	72.65	11.50	59.77	8.70	69.70	57.52	35.60	42.75	33.01
Qwen3.5-27B	68.38	84.03	48.25	71.64	18.75	65.88	26.09	80.13	46.33	44.76	52.00	39.40
Qwen3.5-122B-A10B	69.38	88.31	54.65	85.39	43.75	73.10	13.04	85.19	40.15	50.42	53.50	40.57
Qwen3.5-Flash	64.88	75.48	38.07	73.35	13.25	67.36	4.35	69.36	57.13	36.90	41.75	33.01
Qwen3.5-Plus	69.96	91.92	42.20	93.01	40.50	73.67	21.74	89.23	34.89	55.08	61.25	44.18
Qwen3.5-397B-A17B	70.45	92.17	61.39	92.23	45.50	73.78	34.78	93.27	33.86	55.80	64.25	45.74
GPT-5.3-Codex	68.55	90.68	36.50	96.15	54.25	39.83	8.70	91.33	34.82	55.94	58.25	44.29
GPT-5.4	66.64	87.39	43.26	**97.86**	59.25	41.26	13.04	90.00	36.60	53.83	58.50	43.54
Kimi-K2.5	69.51	92.68	57.14	97.60	43.25	78.55	30.43	93.60	32.21	56.39	62.75	45.93
Gemini-3-Flash	70.38	89.63	62.48	97.01	52.75	72.88	34.78	92.33	31.06	57.78	65.50	45.39
Gemini-3-Pro	**70.83**	**94.49**	**63.09**	96.97	**60.00**	**80.51**	**56.52**	**94.95**	**25.19**	**61.52**	**73.00**	**47.28**

Table 4: Overall model performance across layout, formula, table, chart, and Pseudocode parsing tasks under code- and visual-based metrics (higher is better except Reading Order). Light green: Visual; light blue: Code.

runnable plotting code is far easier for current models than faithfully reproducing the nuanced aesthetics and data distributions of the original figures. Similarly, conventional metrics like TEDS paint an overly optimistic picture of table parsing, while our stricter VRF-S criterion reveals a steep performance plunge. This pattern indicates that partial structural recovery is insufficient—only perfect reconstruction counts.

4.3.2 Complexity & Robustness (RQ2). Fig. 4 reveals a clear difficulty-dependent degradation across all four element types, but with markedly different failure profiles by model tier. For formulas, all three models drop as difficulty increases, with Gemini-3-Pro and Qwen3.5-Plus showing relatively smooth declines, while Qwen3.5-35B-A3B falls more sharply on hard cases.

For tables and charts, the tier gap widens substantially at higher difficulty: Gemini-3-Pro degrades moderately, whereas Qwen3.5-Plus exhibits a large hard-case drop and Qwen3.5-35B-A3B collapses to near-floor performance on tables and remains consistently low on charts. The largest instability appears in pseudocode, where both Qwen models approach near-zero on hard samples, while Gemini-3-Pro, although still strongest, also shows a pronounced decline from easy to hard.

Overall, the curves indicate that current MLLMs retain partial robustness on formula and table reconstruction only at higher capability tiers, while chart and pseudocode reconstruction remain the dominant bottlenecks under increasing structural complexity.

4.3.3 Case Study. Fig. 5 shows an academic screenshot containing a challenging table and a complex chart (left), together with the parsing result produced by Gemini-3-Pro. As can be seen, the reconstructed table exhibits substantial deviations, particularly in its color rendering and the text in the upper-left corner. The chart parsing is even more problematic, with severe errors throughout. In addition, inconsistencies in the sizes of the reconstructed table and chart relative to the original screenshot lead to noticeable discrepancies in the overall page layout. This trend is also reflected by the VRF-A scores in this case study: Layout 75.00, Table 58.33, Chart 43.75 and Pseudocode 100.

4.4 Impact of Agentic Techniques (RQ3)

To answer RQ3—which explores how common agent strategies affect the model's DP capabilities—we systematically evaluate our exploratory multi-agent baseline across different configurations. As shown in Table 5, we degrade the full baseline system by incrementally removing core agentic modules (e.g., w/o Visual Feedback, w/o Tools, and w/o Multi-Agent). For a more comprehensive comparison, we also include a setting where the base model is augmented solely with simple self-reflection (Base w/ Self-reflection). By comparing these configurations, we can empirically isolate and quantify how different levels of agentic scaffolding contribute to the final full-page reconstruction accuracy, and assess whether current MLLMs can fundamentally overcome their limitations when provided with external tools and iterative reasoning.

Table 5 highlights several key insights about agentic scaffolding. First, the Full MA System yields a solid and consistent performance gain of roughly 3 to 7 points across all dimensions compared to the static Base model. This confirms the overall efficacy of combining tooling, feedback, and decomposition. However, while most modules contribute positively to complex elements like charts and layouts, we observe intriguing counter-productive effects in

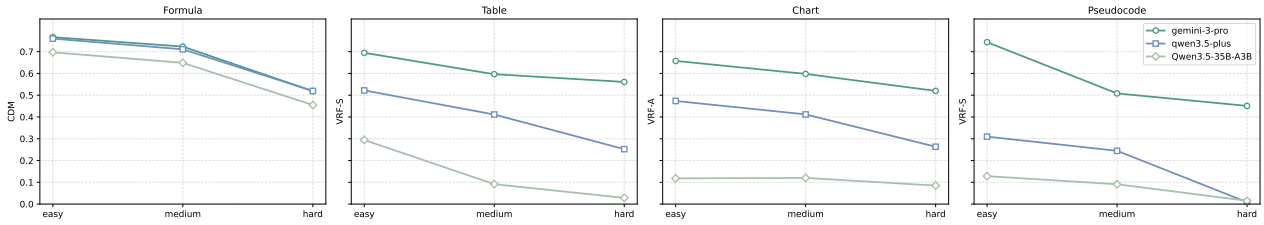

Figure 4: Performance across element-level difficulty (easy, medium, hard) for three representative models at different performance tiers: Gemini-3-Pro (strong), Qwen3.5-Plus (medium), and Qwen3.5-35B-A3B (weak). Table reconstruction is the most robust, while formula, chart, and pseudocode performance degrades with increasing complexity.

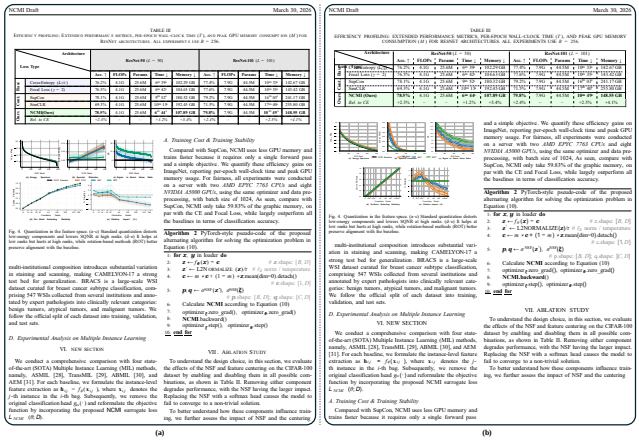

Figure 5: Case study of CADP on a challenging sample: the source screenshot (left panel) v.s. the parsed result by Gemini-3-Pro (right panel). VRF-A scores: Layout 75.00, Table 58.33, Chart 43.75 and Pseudocode 100.

Setting	Avg. Calls	Layout (VRF-A)	Formula (CDM)	Table (VRF-S)	Chart (VRF-A)	Pseudocode (VRF-S)
Base (Gemini-3-Pro)	**1.0**	73.00	70.83	66.09	60.00	56.52
w/ Self-reflection	2.0	71.00	68.80	63.99	58.50	52.17
Full MA System	6.0	**77.20**	74.20	**71.40**	**66.50**	61.80
w/o Visual Feedback	4.5	74.10	**75.30**	66.80	62.00	58.50
w/o Tools	5.0	76.50	73.40	68.20	61.80	61.10
w/o Multi-Agent	3.0	75.80	72.60	67.90	63.50	**62.50**

Table 5: Ablation study for RQ3, designed to quantify how agentic components contribute to reconstruction quality.

structurally rigid domains. For instance, removing Visual Feedback actually increases the score for Formula (from 74.20 to 75.30). This suggests that pixel-based visual critics sometimes misinterpret dense, tiny math symbols, overriding correct LaTeX source with hallucinated "fixes." Similarly, removing Multi-Agent collaboration marginally improves Pseudocode reproduction (62.50 vs. 61.80). This occurs because passing raw textual code through multiple redundant generation cycles between agents inadvertently degrades precise whitespace indents and specific syntax structures. Finally, augmenting the base model with self-reflection causes a uniform performance regression, reinforcing the fact that reflection without external environmental grounding only introduces instability.

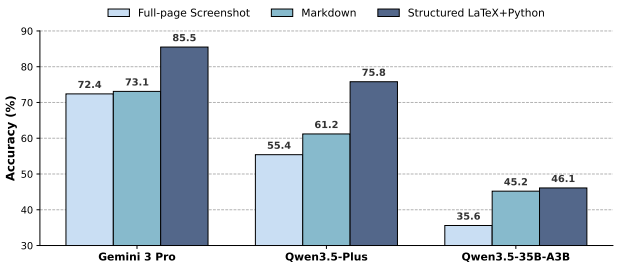

Figure 6: Format-sensitive QA performance across three input representations.

4.5 Format-Sensitive Comprehension (RQ4)

To investigate how input representation affects structured scientific comprehension, we construct a QA benchmark of 90 manually annotated questions requiring evidence from one or two SAEs and their surrounding context. We evaluate three MLLMs at different performance tiers (Gemini-3-Pro, Qwen3.5-Plus, and Qwen3.5-35B-A3B) under three input formats: full-page screenshots, Markdown, and our structured LaTeX+Python representation. For each query, the model receives the question and the complete paper in one of these formats.

Fig. 6 reveals a capability-dependent effect of representation. Qwen3.5-35B-A3B gains little from LaTeX+Python over Markdown, suggesting that smaller models struggle to exploit complex programmatic structure. Gemini-3-Pro performs similarly on screenshots and Markdown, indicating that strong visual grounding can partially mitigate the lossiness of raw page images. In contrast, Qwen3.5-Plus benefits substantially from LaTeX+Python and even surpasses Gemini-3-Pro with Markdown. This performance inversion suggests that structure-preserving representations provide useful inductive biases that can partially compensate for model-scale limitations and improve scientific comprehension.

5 Discussion

The proposed CADP paradigm extends beyond benchmark performance and offers practical value for downstream scientific document understanding. In particular, the Dual-Code representation preserves both page-level structure and executable semantics, providing a stronger foundation than plain-text formats for full-page reconstruction and reasoning.

Potential for Reinforcement Learning in Compilable Document Parsing. Compared with flat Markdown, Dual-Code is substantially more expressive for reconstructing complex academic

pages. This makes CADP naturally compatible with reinforcement learning: generated code can be compiled/executed, rendered outputs can be directly compared with reference pages, and reconstruction metrics can serve as reward signals. Given the large-scale LaTeX ecosystem (e.g., arXiv), this setting is promising for pretraining and post-training toward robust scientific document parsing.

Enabling Graph-Based Retrieval-Augmented Reasoning. LaTeX also provides explicit structural signals that are useful for Graph-RAG. Cross-references (\ref), citations (\cite), table hierarchies, and section organization encode rich relations across text and SAEs (e.g., tables and charts). Preserving these relations in DualCode can alleviate the structure-breaking issue of chunk-based RAG and improve both intra-document and cross-document QA.

6 Conclusion

We present CADP, a paradigm that reconstructs academic pages as contextual LaTeX and executable Python, together with CADP-Bench, an expert-verified benchmark evaluated via re-injection compilation. Experiments on SOTA MLLMs and an exploratory multi-agent baseline show that high-fidelity executable reconstruction remains challenging. Meanwhile, structured LaTeX+Python improves downstream comprehension, particularly for medium-tier models, highlighting the potential of compilable, structure-preserving representations for scientific document understanding.

Acknowledgments

This work is partially supported by National Nature Science Foundation of China under No. 62476058. We thank the Big Data Computing Center of Southeast University for providing the facility support on the numerical calculations in this paper.

References

[1] Hadi Amini, Md Jueal Mia, Yasaman Saadati, Ahmed Imteaj, Seyedsina Nabavirazavi, Urmish Thakker, Md Zarif Hossain, Awal Ahmed Fime, and SS Iyengar. 2025. Distributed llms and multimodal large language models: A survey on advances, challenges, and future directions. *arXiv preprint arXiv:2503.16585* (2025).

[2] Muhammad Arslan, Hussam Ghanem, Saba Munawar, and Christophe Cruz. 2024. A Survey on RAG with LLMs. *Procedia computer science* 246 (2024), 3781–3790.

[3] Jonas Belouadi, Eddy Ilg, Margret Keuper, Hideki Tanaka, Masao Utiyama, Raj Dabre, Steffen Eger, and Simone Ponzetto. 2025. Tikzero: Zero-shot text-guided graphics program synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 17793–17806.

[4] Jinyue Chen, Lingyu Kong, Haoran Wei, Chenglong Liu, Zheng Ge, Liang Zhao, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. 2024. OneChart: Purify the Chart Structural Extraction via One Auxiliary Token. In *Proceedings of the 32nd ACM International Conference on Multimedia*.

[5] Qi Chen, Jingxuan Wei, Zhuoya Yao, Haiguang Wang, Gaowei Wu, Bihui Yu, Siyuan Li, and Cheng Tan. 2025. ResearchPulse: Building Method-Experiment Chains through Multi-Document Scientific Inference. In *Proceedings of the 33rd ACM International Conference on Multimedia*. 9110–9119.

[6] Qiguang Chen, Mingda Yang, Libo Qin, Jinhao Liu, Zheng Yan, Jiannan Guan, Dengyun Peng, Yiyan Ji, Hanjing Li, Mengkang Hu, et al. 2025. Ai4research: A survey of artificial intelligence for scientific research. *arXiv preprint arXiv:2507.01903* (2025).

[7] Cheng Cui, Ting Sun, Suyin Liang, Tingquan Gao, Zelun Zhang, Jiaxuan Liu, Xueqing Wang, Changda Zhou, Hongen Liu, Manhui Lin, Yue Zhang, Yubo Zhang, Handong Zheng, Jing Zhang, Jun Zhang, Yi Liu, Dianhai Yu, and Yanjun Ma. 2025. PaddleOCR-VL: Boosting Multilingual Document Parsing via a 0.9B Ultra-Compact Vision-Language Model. *ArXiv* abs/2510.14528 (2025). https://api.semanticscholar.org/CorpusID:282139252

[8] Harsh Desai, Pratik Kayal, and Mayank Singh. 2021. TabLeX: A Benchmark Dataset for Structure and Content Information Extraction from Scientific Tables. In *16th International Conference on Document Analysis and Recognition, ICDAR 2021, Lausanne, Switzerland, September 5-10, 2021, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12822)*, Josep Lladós, Daniel Lopresti, and Seiichi Uchida (Eds.). Springer, 554–569. doi:10.1007/978-3-030-86331-9_36

[9] Yihao Ding, Siwen Luo, Yue Dai, Yanbei Jiang, Zechuan Li, Geoffrey Martin, and Yifan Peng. 2025. A Survey on MLLM-based Visually Rich Document Understanding: Methods, Challenges, and Emerging Trends. *CoRR* abs/2507.09861 (2025). arXiv:2507.09861 doi:10.48550/ARXIV.2507.09861

[10] Kuicai Dong, Shurui Huang, Fangda Ye, Wei Han, Zhi Zhang, Dexun Li, Wenjun Li, Qu Yang, Gang Wang, Yichao Wang, et al. 2025. Doc-researcher: A unified system for multimodal document parsing and deep research. *arXiv preprint arXiv:2510.21603* (2025).

[11] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. 2024. A survey on llm-as-a-judge. *The Innovation* (2024).

[12] Tingting Han, Qian Zhang, Fangling Chen, Ziyao Wang, Bin Pan, Qianyu Chen, Xuhui Liu, Xiangyuan Lan, and Dongyu Zhang. 2024. ChartLlama: A Multimodal Large Language Model for Chart Understanding and Generation. In *Proceedings of the 32nd ACM International Conference on Multimedia*.

[13] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking. *Proceedings of the 30th ACM International Conference on Multimedia* (2022). https://api.semanticscholar.org/CorpusID:248228056

[14] IDP Authors. 2025. Intelligent Document Parsing: Towards End-to-end Document Parsing via Decoupled Content Parsing and Layout Grounding. In *Findings of the Association for Computational Linguistics: EMNLP*.

[15] Nan Jiang, Shanchao Liang, Chengxiao Wang, Jiannan Wang, and Lin Tan. 2025. LATTE: Improving Latex Recognition for Tables and Formulae with Iterative Refinement. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, Toby Walsh, Julie Shah, and Zico Kolter (Eds.). AAAI Press, 4030–4038. doi:10.1609/AAAI.V39I4.32422

[16] Rihui Jin, Yu Li, Guilin Qi, Nan Hu, Yuan-Fang Li, Jiaoyan Chen, Jianan Wang, Yongrui Chen, Dehai Min, and Sheng Bi. 2025. Hegta: Leveraging heterogeneous graph-enhanced large language models for few-shot complex table understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 24294–24302.

[17] Pratik Kayal, Mrinal Anand, Harsh Desai, and Mayank Singh. 2023. Tables to LaTeX: structure and content extraction from scientific tables. *Int. J. Document Anal. Recognit.* 26, 2 (2023), 121–130. doi:10.1007/S10032-022-00420-9

[18] Jinke Li, Jiarui Yu, Chenxing Wei, Hande Dong, Qiang Lin, Liangjing Yang, Zhicai Wang, and Yanbin Hao. 2025. Unisvg: A unified dataset for vector graphic understanding and generation with multimodal large language models. In *Proceedings of the 33rd ACM International Conference on Multimedia*. 13156–13163.

[19] Xin Li, Mingming Gong, Yunfei Wu, Jianxin Dai, Antai Guo, Xinghua Jiang, Haoyu Cao, Yinsong Liu, Deqiang Jiang, and Xing Sun. 2025. DREAM: Document Reconstruction via End-to-end Autoregressive Model. In *Proceedings of the 33rd ACM International Conference on Multimedia*. 2949–2957.

[20] Zichao Li, Aizier Abulaiti, Yaojie Lu, Xuanang Chen, Jia Zheng, Hongyu Lin, Xianpei Han, Shanshan Jiang, Bin Dong, and Le Sun. 2025. READoc: A Unified Benchmark for Realistic Document Structured Extraction. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025 (Findings of ACL, Vol. ACL 2025)*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, 21889–21905. https://aclanthology.org/2025.findings-acl.1128/

[21] Zhang Li, Yuliang Liu, Qiang Liu, Zhiyin Ma, Ziyang Zhang, Shuo Zhang, Zidun Guo, Jiarui Zhang, Xinyu Wang, and Xiang Bai. 2025. MonkeyOCR: Document Parsing with a Structure-Recognition-Relation Triplet Paradigm. *ArXiv* abs/2506.05218 (2025). https://api.semanticscholar.org/CorpusID:279244468

[22] Zhecheng Li, Guoxian Song, Yiwei Wang, Zhen Xiong, Junsong Yuan, and Yujun Cai. 2025. A^2R^2: Advancing Img2LaTeX Conversion via Visual Reasoning with Attention-Guided Refinement. *CoRR* abs/2507.20890 (2025). arXiv:2507.20890 doi:10.48550/ARXIV.2507.20890

[23] Jun Ling, Yao Qi, Tao Huang, Shibo Zhou, Yanqin Huang, Jiang Yang, Ziqi Song, Ying Zhou, Yang Yang, Heng Tao Shen, and Peng Wang. 2025. Table2LaTeX-RL: High-Fidelity LaTeX Code Generation from Table Images via Reinforced Multimodal Language Models. *CoRR* abs/2509.17589 (2025). arXiv:2509.17589 doi:10.48550/ARXIV.2509.17589

[24] Junbo Niu, Zheng Liu, Zhuangcheng Gu, Bin Wang, Linke Ouyang, Zhiyuan Zhao, Tao Chu, Tianyao He, Fan Wu, Qintong Zhang, et al. 2025. Mineru2. 5: A decoupled vision-language model for efficient high-resolution document parsing. *arXiv preprint arXiv:2509.22186* (2025).

[25] Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. 2025. OmniDocBench: Benchmarking Diverse PDF Document Parsing with Comprehensive Annotations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*. Computer Vision Foundation / IEEE, 24838–24848. doi:10.1109/CVPR52734.2025.02313

[26] Jake Poznanski, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Aman Rangapur, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. 2025. olmOCR: Unlocking Trillions of Tokens in PDFs with Vision Language Models. *CoRR* abs/2502.18443 (2025). arXiv:2502.18443 doi:10.48550/ARXIV.2502.18443

[27] Libo Qin, Qiguang Chen, Yuhang Zhou, Zhi Chen, Yinghui Li, Lizi Liao, Min Li, Wanxiang Che, and Philip S Yu. 2025. A survey of multilingual large language models. *Patterns* 6, 1 (2025).

[28] SciMDR Authors. 2026. SciMDR: A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers. *arXiv preprint* (2026).

[29] Visual Programmability Authors. 2025. Visual Programmability: A Guide for Code-as-Thought in Chart Understanding. *arXiv preprint* (2025).

[30] Bin Wang, Zhuangcheng Gu, Guang Liang, Chao Xu, Bo Zhang, Botian Shi, and Conghui He. 2024. UniMERNet: A Universal Network for Real-World Mathematical Expression Recognition. arXiv:2404.15254 [cs.CV] https://arxiv.org/abs/2404.15254

[31] Shuaijie Wang, Hao Li, Yuting Chen, Yao Peng, Wei Pang, Xiangpeng Chen, Zhaowen Wang, Jianbing Shen, and Xiangyuan Lan. 2022. mmLayout: Multi-grained MultiModal Transformer for Document Understanding. In *Proceedings of the 30th ACM International Conference on Multimedia*. 4877–4886.

[32] Renqiu Xia, Hancheng Ye, Xiangchao Yan, Qi Liu, Hongbin Zhou, Zijun Chen, Botian Shi, Junchi Yan, and Bo Zhang. 2025. ChartX and ChartVLM: A Versatile Benchmark and Foundation Model for Complicated Chart Reasoning. *IEEE Trans. Image Process.* 34 (2025), 7436–7447. doi:10.1109/TIP.2025.3607618

[33] Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran Xu, Xinyu Zhu, Siheng Li, Yuxiang Zhang, Gongye Liu, Xiaomei Nie, Deng Cai, and Yujiu Yang. 2025. ChartMimic: Evaluating LMM's Cross-Modal Reasoning Capability via Chart-to-Code Generation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net. https://openreview.net/forum?id=sGpCzsfd1K

[34] Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Yuhao Dan, Chenlin Zhao, Guohai Xu, Chenliang Li, Junfeng Peng, Shuai Yuan, Ji Zhang, Jian Sun, Qi Tian, Liang He, Xingjian He, Xin Tao, and Peng Gao. 2024. mPLUG-DocOwl 1.5: Unified Structure Learning for OCR-free Document Understanding. In *Proceedings of the 32nd ACM International Conference on Multimedia*.

[35] Qintong Zhang, Victor Shea-Jay Huang, Bin Wang, Junyuan Zhang, Zhengren Wang, Hao Liang, Shawn Wang, Matthieu Lin, Conghui He, and Wentao Zhang. 2024. Document Parsing Unveiled: Techniques, Challenges, and Prospects for Structured Information Extraction. *CoRR* abs/2410.21169 (2024). arXiv:2410.21169 doi:10.48550/ARXIV.2410.21169

[36] et al. Zhao. 2024. OmniDocBench: Benchmarking Diverse PDF Document Parsing with Comprehensive Annotations. *arXiv preprint arXiv:2412.07626* (2024).

[37] Xuanle Zhao, Xuexin Liu, Haoyue Yang, Xianzhen Luo, Fanhu Zeng, Jianling Li, Qi Shi, and Chi Chen. 2025. ChartEdit: How Far Are MLLMs From Automating Chart Analysis? Evaluating MLLMs' Capability via Chart Editing. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025 (Findings of ACL, Vol. ACL 2025)*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, 3616–3630. https://aclanthology.org/2025.findings-acl.185/

[38] et al. Zhou. 2026. Benchmarking MLLMs on Scan-Oriented Academic Paper Reasoning. In *International Conference on Learning Representations (ICLR)*.

A Prompts

Prompt for Compilable Academic Document Parsing

```
I will provide a screenshot of a single-page academic paper. Please convert it according to the following requirements.

[General Rules]
- You may output a brief thinking process, but you must output three code blocks in this fixed order: Python first, then a package block, then LaTeX.
- If there are no charts in the screenshot, the Python code block should contain only a single comment line: # No charts detected.
- The overall LaTeX format template has been provided externally. I will compile it using pdflatex. Determine whether any additional packages are needed to reliably reproduce the page content. If needed, output them in a package block. Try to use existing packages whenever possible.
- The LaTeX code block must contain only the body content between \begin{document} and \end{document}. It is strictly forbidden to output the LaTeX preamble in the LaTeX code block (such as \documentclass or \usepackage).
- Any additional packages, if needed, must be placed only in the package block, not in the LaTeX code block.
- Do not output \begin{document} or \end{document} themselves. Ignore headers, footers, page numbers, and similar information. Don't add any extra content. Don't use multi-column formatting in the main text code.

Figure:
* First determine whether there are any charts in the screenshot. If so, convert all figures/subfigures appearing in the screenshot into Python code for reproduction.
* If the data or parameters in a figure are not fully specified, reasonable approximations may be made.
* The code should use savefig to save the images to the current directory.
* File names should be fig1.pdf, fig2.pdf, ... (in the order they appear in the screenshot: top to bottom, left to right).
* If the original figure contains multiple subfigures within one Figure, you may choose:
    * Option A: save each subfigure separately as one fig{i}.pdf
    * Option B: reproduce them in Python as one combined figure with multiple subplots, and save it as one fig{i}.pdf
    * The chosen option must be explained in the Thinking Process, and the LaTeX figure inclusion must correspond to it.

All content other than figures (layout, text, tables, pseudo-code, formulas, etc.):
* Convert it into LaTeX code, preserving the original structure, hierarchy, and layout as much as possible.
* Use standard LaTeX math environments for mathematical formulas, and reproduce algorithms (pseudo-code).
* Use LaTeX table environments for tables, preserving alignment, borders, and captions as much as possible.
* The LaTeX must include code for inserting the images, and the paths and file names must exactly match the Python output.

The following packages are included in the template:
<<<USEPACKAGE_BLOCK>>>

Output format:
(Thinking Process)

```python
# Complete runnable Python code
```

(Thinking Process)

```package
% Leave this block empty if no additional packages are needed.
\usepackage{package_name}
```

```latex
% LaTeX body code
```
```

Prompt for Table Restoration Evaluation

```
You are an expert in document parsing and table structure evaluation. Your task is to rigorously assess the restoration fidelity between a table in the original page image and the corresponding table in one or more rendered page images / parsing outputs.

Input:
1. One GT image containing the original table.
2. One or more predicted images or parsing-result renderings for the same test sample.

Evaluation Objective:
Evaluate how faithfully the predicted result restores the table in the GT image. The evaluation must consider:
- table topology and structure
- cell content accuracy
- formatting and alignment
- advanced visual styling
```

```
If a large table is split across multiple predicted pages, assess whether the page split is logically handled and whether all content is
preserved without omission.

Scoring Instructions:
For each dimension below, assign an integer score from 0 to 4, strictly following the rubric. Also provide a brief, objective justification.

1. table_structure
Evaluate row/column organization, merged/split cells, and header hierarchy.
   - 4: Perfect structural restoration. Row and column layout exactly matches the GT. All merged cells (spanned cells) are restored with
   correct positions and spans. Header hierarchy is clear and perfectly aligned.
   - 3: Structure is mostly correct. There are only minor non-critical merge/split errors, but they do not affect the ownership of data
   across full rows or columns.
   - 2: Noticeable structural errors exist, such as row/column misalignment, row shifting, or incorrect header merges that partially
   confuse data association.
   - 1: The table topology is severely damaged. Many merged cells fail or are incorrectly split, making it difficult to locate original
   data by row and column.
   - 0: The table structure collapses completely, appearing as plain text or with fully broken row/column logic.

2. table_content
Evaluate the accuracy of text, numbers, symbols, and preservation of empty cells inside each cell.
   - 4: All cell content is restored with 100% accuracy. No missing, extra, or incorrect characters. Punctuation is exact. Empty cells are
   correctly preserved.
   - 3: Content is overwhelmingly accurate, with only a few minor OCR-like issues (for example, full-width/half-width symbol confusion, or
   0 vs O), but no change to core meaning or numeric value.
   - 2: Multiple obvious content errors exist, such as missing words, incorrect characters, altered numbers, missing decimal points, wrong
   units, or hallucinated text.
   - 1: Large portions of the content are missing or incorrect. Important data (such as amounts, key metrics, or identifiers) is badly
   distorted, or many empty cells are incorrectly filled with garbage text.
   - 0: Content is almost entirely unrecognizable, empty, or completely wrong.

3. format_and_alignment
Evaluate alignment, line breaks, font hierarchy, emphasis styles (bold, italic, underline), and overall visual proportion.
   - 4: Alignment is fully correct (for example, centered titles, right-aligned numbers, left-aligned body text). Line wrapping is natural
   and closely matches the GT. Font hierarchy is clear. Bold, italic, underline, and other emphasis styles are accurately restored. Column
   widths, row heights, and overall proportions closely match the GT.
   - 3: Most formatting and alignment are correct, with only minor local deviations. Line breaks or font proportions may differ slightly. A
   few emphasis styles may be imperfect, but overall readability and interpretation are unaffected.
   - 2: Clear formatting problems exist, such as inconsistent alignment across columns, awkward line wrapping, unbalanced font hierarchy,
   or partially missing bold/italic/underline styles. The table remains readable, but usability is reduced.
   - 1: Formatting is obviously chaotic. Text is cramped, overflowed, miswrapped, misaligned, or overlapping. Alignment logic fails in many
   places. Emphasis styles are mostly missing or incorrect, seriously harming readability.
   - 0: The table loses nearly all formatting logic and visual organization. Content is piled together and cannot be understood through
   layout or styling.

4. advanced_style
Evaluate higher-level visual styling, including background colors, fills/shading, text colors, border thickness and line styles, separator
hierarchy, bullets, arrows, asterisks, brackets, superscripts/subscripts, and other special symbols or decorative elements.
   - 4: All advanced styles are accurately restored. Background fills, shading, text colors, border weights, solid/dashed lines, separator
   hierarchy, and special symbols are complete and correctly positioned.
   - 3: Most advanced styles are restored correctly. There may be slight deviations in color intensity, border thickness, or a few symbol
   style details, but the visual hierarchy and emphasis remain clear.
   - 2: Noticeable advanced-style loss or errors exist, such as missing header background color, missing key borders, inconsistent line
   styles, or missing/replaced special symbols, causing visible mismatch from the GT.
   - 1: Advanced styling is severely distorted. Many colors, borders, and symbols are missing or wrong, and important regions can no longer
   be distinguished visually.
   - 0: Almost no advanced styling is preserved. Colors, borders, and special symbols are entirely missing or fundamentally incorrect.

Output Requirements:
Output in JSON format, maintaining the following structure. Do not add any extra text:
```json
{
  "metrics": {
    "table_structure": {"reasoning": "", "score": x},
    "table_content": {"reasoning": "", "score": x},
    "format_and_alignment": {"reasoning": "", "score": x},
    "advanced_style": {"reasoning": "", "score": x}
  }
}
```
Notes:
Compare them on a cell-by-cell basis
```

Prompt for Chart Restoration Evaluation

```
You are a professional data visualization evaluation expert. Your task is to strictly evaluate the consistency between a "Ground Truth" chart and a "Predicted" chart, focusing on the data expression, visual elements, and layout design of the charts themselves.

[Scoring Criteria]
Please score the following four dimensions on a scale of 0 to 4. The scoring must strictly follow the specific criteria below, and you must provide concise, objective reasons:

1. Data Expression & Morphology (trend_shape) - The core metric, evaluating the accuracy of data transmission.
   - 4: The main trends, data point distribution, peak/valley positions, clustering structures, or overall shape are highly consistent with the original chart; visual deviations are difficult to discern with the naked eye.
   - 3: The overall shape is consistent, but there are extremely slight deviations in a very small number of local data points, line curvature, or bar heights, which do not affect the data's conclusions.
   - 2: Possesses the basic trend and rough structure of the original chart, but there are deviations clearly visible to the naked eye (e.g., some data points are misaligned, trend lines do not fit well).
   - 1: Retains only extremely vague morphological similarities; the relative sizes, trends, or distributions of key data have changed, easily misleading readers.
   - 0: Incorrect chart type, or the data morphology has absolutely no relation to the original chart.

2. Text & Labels (text_labels) - Evaluating the completeness of information transmission.
   - 4: The spelling, casing, and position of all key text (main/subtitles, legends, axis labels, tick values, data labels) are completely accurate.
   - 3: The text is basically complete, with only minor typographic shifts in non-critical positions, or occasional typos/omissions that do not affect understanding.
   - 2: There are obvious missing texts (e.g., missing an axis label) or relatively many typos, but this does not hinder guessing the basic content of the chart.
   - 1: Key information such as legends or axis ticks is lost, or overlapping and cluttered text makes it difficult to read.
   - 0: No meaningful text labels exist at all, or the text content completely contradicts the original chart.

3. Color & Visual Style (color_style) - Evaluating the consistency of visual encoding.
   - 4: Visual encodings such as color values (RGB/Hex), opacity, line types (solid/dashed), point types (circle/square/triangle), shadows, or fill patterns are highly consistent with the original chart.
   - 3: The color scheme is similar (e.g., same color palette but slight differences in saturation), line and point types basically correspond, and semantic differentiation functions remain intact.
   - 2: Completely different colors are used, but correct categorical differentiation is maintained (e.g., original uses red/blue, reproduction uses green/yellow); or there is a major style difference but the chart remains readable.
   - 1: Changes in color or visual style destroy the readability of the data (e.g., the original uses two colors to distinguish two categories, but the reproduction uses only one color).
   - 0: Extremely poor visual style, or severe contrast issues make the chart unviewable.

4. Local Layout & Proportions (local_layout) - Evaluating spatial composition.
   - 4: The aspect ratio of the plot area, legend position, axis margins, grid line density, and relative spatial relationships of internal elements are highly consistent.
   - 3: The overall layout is harmonious; only the legend position is slightly shifted, or there are slight differences in margin proportions.
   - 2: Obvious proportion imbalances appear (e.g., the original is a wide chart, but the reproduction is a square chart), or the legend/title encroaches on too much of the plot area's space.
   - 1: Severe mutual overlapping between elements occurs (e.g., labels blocking lines, legends covering data); spatial utilization is extremely unreasonable.
   - 0: The internal layout is fragmented, losing the basic structure of a chart.

[Output Requirements]
Output in JSON format, maintaining the following structure. Do not add any extra text:
```json
{
  "metrics": {
    "trend_shape": {"reasoning": "", "score": x},
    "text_labels": {"reasoning": "", "score": x},
    "color_style": {"reasoning": "", "score": x},
    "local_layout": {"reasoning": "", "score": x}
  }
}
```
```

Prompt for Document Layout Restoration Evaluation

```
You are a professional document typesetting and layout evaluation expert. Your task is to strictly evaluate the layout restoration consistency between a "Ground Truth raw page image" and one or more "Predicted rendered page images", focusing on the macro distribution, alignment logic, and spacing proportions of the page.
```

```
Input contains:
1. A Ground Truth raw page image.
2. One or more rendered output images (predicted results) corresponding to the same test sample.

[Evaluation Principles and Constraints]
- Only evaluate layout reconstruction fidelity. Ignore OCR text recognition errors, minor deviations in formula content, or correctness of data within charts.
- Due to font scaling or rendering engine differences, the prediction may render multiple pages; consider which page each element appears on.

[Scoring Criteria]
Please assign scores from 0 to 4 for the following three dimensions. The scoring must strictly follow the criteria below, and provide concise, objective justifications:

1. macro_distribution:
Evaluate the overall layout structure, relative positions of modules, and continuity of reading flow (including cross-page logic).
   - 4: The macro layout structure (single/double column) is perfectly reproduced. All modules (title/charts/body text/formulas) match the GT positions exactly. Perfect single-page layout.
   - 3: The structure is largely consistent, with slight vertical drift of modules; in multi-page cases, reading flow is smooth with only a few extra lines of text.
   - 2: Noticeable structural changes (e.g., partial shift from double-column to single-column), or charts are clearly displaced or moved to another page.
   - 1: The layout structure is severely damaged; charts/formulas intrude into the text flow.
   - 0: Content is completely disordered, structure collapses, and reading flow is broken.

2. alignment_logic:
Evaluate alignment details such as edges, baselines, indentation, and centering.
   - 4: All alignment types are perfectly reproduced, and all content remains within a reasonable layout without any noticeable anomalies.
   - 3: Most alignments are correct, with only a few minor indentation or edge alignment issues in some paragraphs, formulas, or charts.
   - 2: Multiple noticeable alignment failures (e.g., centered titles shifted to one side, loss of list indentation hierarchy, or content exceeding normal boundaries).
   - 1: Large-scale misalignment; uneven column edges; visually chaotic layout.
   - 0: No coherent alignment logic at all.

3. spacing_proportion:
Evaluate whitespace (line spacing, paragraph spacing, margins) and relative visual scale (font hierarchy, chart scaling).
   - 4: Whitespace closely matches GT; element proportions are perfectly reproduced (charts properly scaled; clear distinction between heading levels and body text).
   - 3: Spacing and proportions are generally reasonable; occasional minor inconsistencies in paragraph spacing or slight chart scaling issues; font hierarchy slightly weakened but still clear.
   - 2: Noticeable imbalance (e.g., charts tightly packed with no spacing, large unreasonable blank areas; small icons overly enlarged, or important charts severely reduced).
   - 1: Severe distortion (e.g., overlapping content; body text larger than headings, breaking hierarchy).
   - 0: Elements are compressed, overlapping, or fragmented; proportions are completely out of control, causing visual distortion.

[Output Requirements]
Output in JSON format, maintaining the following structure. Do not add any extra text:
```json
{
  "metrics": {
    "macro_distribution": {"reasoning": "", "score": x},
    "alignment_logic": {"reasoning": "", "score": x},
    "spacing_proportion": {"reasoning": "", "score": x}
  }
}
```
```

Prompt for Pseudocode Restoration Evaluation

```
You are a professional expert in pseudocode layout parsing and visual restoration evaluation. Your task is to strictly evaluate the visual and structural consistency between a GT algorithm render and a predicted algorithm render.

[Evaluation Criteria]
Please score the following three dimensions on a scale of 0 to 4. The scoring must strictly follow the specific criteria below, and provide concise, objective reasons:

1. Structure and Indentation Fidelity (structure_and_indentation): Examine the number of lines of code, line numbers (if any), hierarchical indentation, and the border lines/guidelines of code blocks.
   - 4: Hierarchical indentation is perfectly consistent with the GT, accurately restoring the nested levels of control flows such as if/for/while; line numbers correspond completely; the overall algorithm borders and separator lines have no omissions.
   - 3: The vast majority of the structure is correct, occasionally there are slight deviations in single-line indentation, but it does not affect the visual hierarchy of the overall control flow.
```

```
   - 2: There are obvious structural flaws, such as missing original line numbers in the GT, or the indentation levels of some nested code
   blocks are lost/confused, but the rough structure remains.
   - 1: The indentation logic is severely damaged, multiple nested levels are flattened or disordered, resulting in an extremely chaotic
   visual control flow of the code.
   - 0: No structure to speak of, all code is crowded together or completely flattened into regular paragraphs.

2. Math Symbols and Content Transcription (math_and_content_transcription): Examine the accuracy of variable names, mathematical formulas,
Greek letters, and logical/assignment operators (such as \leftarrow, \forall, \sum).
   - 4: All regular text, complex mathematical symbols, superscripts/subscripts, and special operators are perfectly and precisely
   recognized and restored, without a single error or omission.
   - 3: The main content is precise, with only a very few (1-2) minor OCR or LaTeX compilation flaws.
   - 2: There are multiple obvious symbol transcription errors, but the original meaning can still be barely guessed.
   - 1: Large-scale failure in recognizing mathematical symbols, garbled text, or loss; core formulas are completely damaged.
   - 0: Completely failed to correctly recognize the math/symbol content in the pseudocode, or presents meaningless garbled text.

3. Style and Formatting (style_and_formatting): Examine the style of keywords (bold/italic), font environment (distinction between math
mode and text mode), and the alignment logic of comments.
   - 4: The font weight (bold/regular) of keywords (e.g., Input, while, return) is consistent with the GT; variables (italic math mode) and
   regular explanatory text (upright) are strictly distinguished; end-of-line comment symbols and indentation alignment are perfectly
   restored.
   - 3: The style is mostly correct; occasionally keywords might be forgotten to be bolded, or comment alignment has extremely slight
   misalignment.
   - 2: Obvious neglect of style details, for example, no bold/italic distinction throughout the entire text, or comment positions are
   messy and unaligned.
   - 1: Formatting style is completely wrong, for example, forcing all plain text into math mode, making it visually extremely
   unprofessional.
   - 0: No formatting style to speak of at all.


[Output Requirements]
Output in JSON format, maintaining the following structure. Do not add any extra text:
```json
{
  "metrics": {
    "structure_and_indentation": {"reasoning": "", "score": x},
    "math_and_content_transcription": {"reasoning": "", "score": x},
    "style_and_formatting": {"reasoning": "", "score": x}
  }
}
```
```