Diagram-MMU: A Multi-Modal Benchmark for Scientific Diagrams

Weihao Bo[1,2⋆], Shan Zhang[3†], Yanpeng Sun[4⋆], Jie Liu[2,5], Yongke Yao[2,6], Jinhao Du[7], Wei He[2], Kai Zou[8], Zechao Li[1†], and Jingdong Wang[2‡]

[1]Nanjing University of Science and Technology, China [2]Baidu Inc., China
[3]AIML, Adelaide University, Australia [4]SUTD, Singapore
[5]Southeast University, China [6]East China Normal University, China
[7]University of Oxford, United Kingdom [8]NetMind.ai, London, United Kingdom
[⋆]Equal contribution [†]Corresponding author [‡]Project lead

Abstract. Multimodal Large Language Models (MLLMs) have been growing the capability for scientific writing and collaboration. For example, OpenAI Prism is a free workspace for scientific writing and collaboration. One important feature in Prism is turning scientific diagrams directly into LaTeX TikZ code. In this paper, we build a benchmark, Diagram-MMU, a multi-modal benchmark designed to assess MLLMs' ability for scientific diagram parsing and understanding. Diagram-MMU features 3.7k curated diagrams and 18.3k human-validated questions across six domains. It evaluates MLLMs on three tasks common in vibe writing workspaces: diagram-to-code parsing, diagram-to-code editing, and diagram question answering, alongside agentic settings per task. The evaluation of 12 MLLMs reveals that diagram-to-code tasks are more challenging than diagram question answering: models can reason well over diagrams but struggle to parse and edit them, underscoring the need for methods to enhance MLLMs' capability in diagram-to-code generation. Under agentic settings, most models improve parsing and editing performance but degrade on question answering, while Claude-4.6 Opus consistently improves across all three tasks. Our benchmark is publicly available at `https://huggingface.co/datasets/AIGrounding/Diagram-MMU`.

Keywords: Scientific diagrams · Multimodal large language models · Diagram-to-code · Diagram understanding · Benchmark

1 Introduction

In the wake of significant advancements in large language models (LLMs) [24, 36, 47], multimodal large language models (MLLMs) [9, 15, 25, 37] have rapidly evolved to extend their versatility across vision–language tasks [45, 46], spanning visual perception [34], description [31], generation [20], and agentic modeling [7]. To bring these capabilities into everyday research workflows, OpenAI Prism [26] provides a free vibe writing workspace where any researcher who writes in LaTeX can draft papers, generate figures, and convert scientific diagrams into TikZ code.

Table 1: Comparison with diagram-to-code and question-answering benchmarks. #Diag.: unique diagrams across diagram types (#Diag. Type). Tasks include three fundamental tasks (evaluating how to think): Diagram-to-Code Parsing (D2C-P), Editing (D2C-E), and Question Answering (DQA), alongside agentic settings (evaluating how to act). Evaluation spans multi-levels: object-level F1 for fine-grained element grounding, CrystalBLEU for TikZ syntax correctness, image-level for visual appearance similarity, and answer accuracy (Acc.) for DQA. Benchmarks without TikZ Code (✗) target Python charts, HTML WebUIs, SVG graphics, or no code (ChartE^3). ✓* is partially supported (limited editing or text-only input). ⊘: synthetic diagrams.

Benchmark	Data			Tasks				Evaluation Metrics			
	TikZ Code	#Diag. Type	#Diag.	D2C-P	D2C-E	DQA	Agentic	Object	CBLEU	Image	Answer Acc.
Diagram-to-Code Parsing											
ChartMimic ⊘ [42]	✗	1	2.4k	✓	✓*	✗	✗	✓	✓	✓	✗
InfiBench-V [18]	✗	2	0.3k	✓	✗	✗	✗	✗	✓	✓	✗
Plot2Code [41]	✗	1	0.1k	✓	✗	✗	✗	✗	✓	✓	✗
SVG-Diagrams [30]	✗	4	0.5k	✓	✗	✗	✗	✗	✗	✓	✗
DiagramGenBenchmark [40]	✓	2	0.4k	✓	✓*	✗	✗	✗	✓	✓	✗
AutomaTikZ [5]	✓	4	1.0k	✓*	✗	✗	✗	✗	✓	✓	✗
DeTikZify [6]	✓	4	1.5k	✓	✗	✗	✗	✗	✓	✓	✗
Image2Struct [29]	✓	4	0.3k	✓	✗	✗	✗	✗	✗	✓	✗
Diagram-to-Code Editing											
ChartE^3 [19]	✗	1	0.8k	✗	✓	✗	✗	✓	✗	✓	✗
ChartM^3⊘ [43]	✗	1	1.0k	✗	✓	✗	✗	✗	✓	✓	✗
Diagram Question Answering											
CharXiv [39]	✗	1	2.3k	✗	✗	✓	✗	✗	✗	✗	✓
ChartQA [23]	✗	1	1.0k	✗	✗	✓	✗	✗	✗	✗	✓
SGP-Bench ⊘ [27]	✗	2	3.5k	✗	✗	✓*	✗	✗	✗	✗	✓
MATHEMETRIC ⊘ [32]	✗	3	1.2k	✗	✗	✓	✗	✓	✗	✗	✓
Diagram-MMU (Ours)	✓	**6**	**3.7k**	✓	✓	✓	✓	✓	✓	✓	✓

Scientific diagrams are the primary visual medium through which researchers convey experimental results, mathematical relationships, molecular structures, and circuit topologies [32, 39, 42]. A researcher working with diagrams needs to: (1) parse a diagram into editable TikZ code so it can be included in a manuscript; (2) edit a diagram–changing colors, labels, data values, or layout–to fit specific requirements; and (3) describe or reason about diagram to interpret domain-specific semantics of symbols, or even perform hypothetical reasoning (what-if) about how answers change conditioned on modified diagrams.

To assist with the above tasks, MLLMs need strong foundational abilities in perception, coding, domain knowledge, and reasoning (*e.g.*, *how to think* [9]). Moreover, in the vibe writing workspace, models also need to know *how to act* (agentic ability [9]): searching TikZ for unfamiliar syntax, coding based on the provided visual objects, building on TikZ codes to apply edits, or deciding which of these steps to invoke for task solving. Most existing benchmarks (Table 1) cover narrow diagram types (predominantly charts) with Python or SVG as the code representation, include a single task (coding, editing, or question answering), and no current benchmark designs agentic settings during evaluation.

We thus introduce Diagram-MMU, a multi-modal benchmark designed to assess MLLMs' ability across diagram-to-code parsing (D2C-P), diagram-to-code editing (D2C-E), and diagram question answering (DQA), alongside agentic evaluation settings. This benchmark comprises a carefully curated dataset sourced

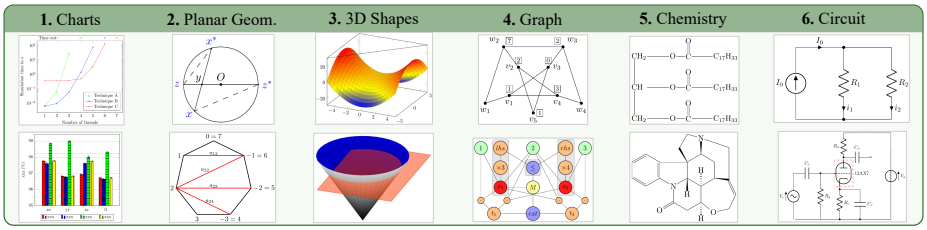


Fig. 1: Demo illustration of the six types of diagrams in Diagram-MMU.

from TikZ/LaTeX documentations, with a selection and filtering process to control quality. The final 3,744 unique diagrams paired with 18,305 evaluation samples are cross-validated by 13 graduate students. Specifically, (1) we adopt TikZ code, which is commonly used in paper writing in LaTeX and integrates directly into Overleaf and Prism, and cover six scientific diagram types including charts, planar geometry, 3D shapes, graphs, chemistry, and circuit diagrams (Fig. 1)–for the first time covering chemistry and circuits for diagram-to-code tasks; (2) for evaluation metrics on diagram-to-code tasks, we use three levels: image-level measuring visual appearance similarity, code-level measuring syntactic correctness, and our object-level F1 scores measuring the basic objects that the code draws; (3) we design 16 controllable evaluation settings (Table 4): 3 for foundational ability and 13 for agentic ability–context utilization, tool use, state management, and planning. Importantly, for tool use we build our TikZ search tool as an MCP server [1], enabling MLLMs to selectively access relevant references, reducing the noise of web search and the context rot caused by loading full PDF manuals.

Overall, Diagram-MMU is the first benchmark to test both foundational and agentic abilities for solving scientific diagram-related tasks. Evaluation on 12 MLLMs (6 closed-source, 6 open-source) reveals the key findings on *foundational ability*: (1) Models can reason well over diagrams (DQA accuracy up to 86%) but struggle to code them (D2C-P object-level F1 ranges 31–57%), revealing perception and coding limitations; (2) Models that fail on diagram-to-code parsing also struggle with editing (D2C-E) and answering tasks (DQA); (3) Gemini-3.0 Pro achieves the most balanced profile across three tasks. On the *agentic ability*, (4) agency benefits editing more than parsing, likely because textual editing instructions help guide tool and context use; (5) most models degrade from excessive retrieval or poorly targeted queries in the TikZ search process, while Claude-4.6 Opus has the strongest tool use ability; (6) planning is the weakest agentic capability, especially on DQA (-0.3 to -8.8 accuracy degradation). Diagram-MMU serves as a pilot evaluation testbed for scientific diagram parsing, editing, and understanding in vibe writing workspaces, and we hope it inspires further works.

2 Related Works

Most existing benchmarks focus on charts [19, 23, 39, 41, 43] or narrowly cover scientific diagram types [5, 6, 30, 32], with several using synthetic diagrams [27,

32, 43]. Moreover, benchmarks that adopt Python or SVG as the code representation produce standalone scripts or markup outside the LaTeX ecosystem, limiting direct integration into scientific authoring environments. We compare Diagram-MMU with existing diagram benchmarks in Table 1; extended discussion is provided in Appendix §A.

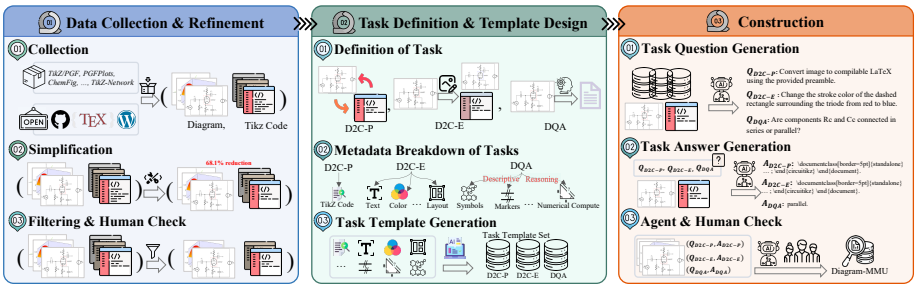


Fig. 2: Task construction pipeline for Diagram-MMU.

Code representation. Most coding benchmarks adopt Python, primarily for charts [19, 41–43], or SVG/HTML, for icons, fonts, emojis and web UI [18, 30]. Although these languages are common, they are not native to LaTeX: they are rendered by matplotlib/CairoSVG into standalone image files rather than compiled inline within LaTeX authoring environments (Overleaf). Moreover, those are vector graphics, generating low-level path elements and fail to maintain accurate geometric relations or only generate outputs of limited complexity [5]. Due to the expressiveness and diverse packages supporting science in TikZ, [5, 6, 29] build TikZ-based benchmarks, but evaluate only diagram-to-code parsing on limited diagram types. Vibe writing workspaces, *e.g.*, Prism [26], further advance the utilization of TikZ code: even researchers are not familiar with TikZ syntax, vibe writing frameworks can assist them in diagram-to-code parsing, editing, and integrating the output automatically into manuscripts. We thus use TikZ as code representation. Beyond covering broader diagram types and evaluation metrics, we also build a TikZ search tool as an MCP server to provide TikZ syntax references to advance TikZ integration into vibe writing frameworks.

Task coverage and question types. ChartE^3 [19] designs local and global editing templates for charts, and CharXiv [39] introduces descriptive and reasoning question types for chart understanding; Diagram-MMU extends such designs from charts to all six scientific domains. In Diagram-MMU, each diagram is paired with coding, editing, and understanding questions for joint evaluation, and DQA answers are open-ended and graded by an LLM judge, avoiding the selection bias of multiple-choice [12, 21] or yes/no formats.

3 Diagram-MMU

3.1 Data Collection

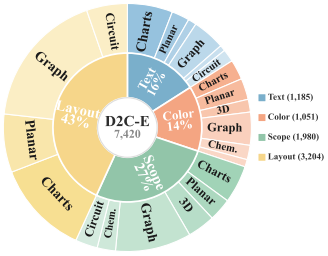
Source. As shown in the first column of Fig. 2. We collect the TikZ source code from two primary sources. One source is official package handbooks, *i.e.*, the

Table 2: Data statistics of Diagram-MMU. In Table 2a, TikZ.p indicates TikZ package; basic graphical objects are shared across diagrams, while domain-specific objects are summarized in the footnote; #Q denotes total questions; Knowledge is domain-specific for the DQA task. Figure 2b is the metadata breakdown of D2C-E, editing dimensions acorss text, color, scope, and layout: scope operates on local transformations and layout operates on global transformations (*e.g.*, chart type conversion).

	D2C-P		DQA	
Domain	#Q	TikZ.p	#Q	Knowledge
Charts	959	`pgfplots`	1,834	statistics, trend/extremum analysis
Planar Geom.	598	`tikz/tkz-euclide`	1,118	Euclidean notation/formulas
3D Shapes	236	`pgfplots`	455	solid geometry, cross-sections
Graph	1,356	`tikz`	2,679	degree, connectivity, paths, tree
Chemistry	187	`chemfig`	366	bond, atom, functional groups
Circuit	403	`circuitikz`	694	topology, electrical/power laws

Shared objects: `path:open`, `path:closed`, `node:rectangle`, `node:circle`, `text`, ⋯;
Domain-specific: `data_point/axis` (Charts, 3D), `node:vsource/node:resistor` (Circuit).

(a)



(b)

TikZ/PGF manuals: PGFPlots, CircuiTikZ, TKZ-Euclide, ChemFig, and TikZ-Network documentation. The other source is community resources, including `texample.net`, TeX Stack Exchange, and GitHub `tikz_favorites`. There are totally 6,849 pieces of TikZ code, 3,540 and 3,309 per source.

Simplification, filtering, and categorization. We simplify the TikZ code by removing redundant and useless codes, such as preamble packages unrelated to diagrams (*e.g.*, theorem environments). We check the correctness of the simplied code by two ways: compiling the simplified code using `pdflatex`, `lualatex`, or `xelatex` and comparing the rendered images of the original and simplified code. We compare the renderd images by an automatic MLLM model comparison and a further mannual comparison. If the compile is not successful or the rendered images are not the same, the corresponding code sample is removed.

We then manually remove (near-)duplicate and trivial diagrams (*e.g.*, single arrows, borders, or logos). We use the MLLM, Qwen3-VL-235B-A22B, to classify the diagram and the code into 6 domains: charts, planar geometry, 3D shapes, graph structures, chemical expressions, and circuit. Then, we mannually check the classification correctness and make adjustment if necessary. Finally, we have **3,744** diagrams. Fig. 1 presents the examples of the diagrams of the 6 domains. The statistics is given in Table 2.

3.2 Task Definition

Diagram-to-Code Pasing (D2C-P). The task is to prompt the MLLM to parse the input diagram into the TikZ code. An example of the prompt is "Recreate this diagram using LaTeX. Provide the full, executable code. Start with `\documentclass[tikz,border=5pt]{standalone}\usepackage{pgfplots}...` ". It contains instruction conditioned on a provided preamble (document class, required packages, and libraries). The top row of Fig. 3 illustrates this task.

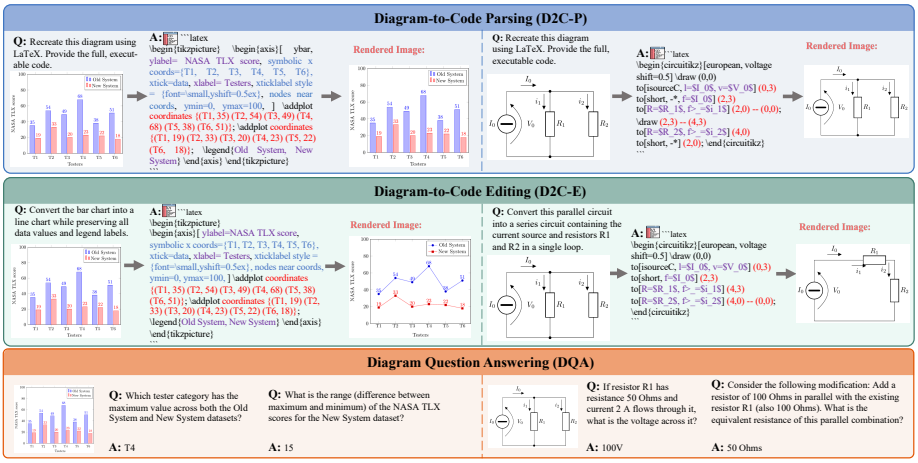


Fig. 3: Task overview of Diagram-MMU. Diagram-to-Code Parsing is to prompt the MLLMs to parse the input diagram into the TikZ code; Diagram-to-Code Editing is to prompt the MLLMs to enerate the TiKZ code that can produce an rendered diagram meeting with the modification requirement; and Diagram Question Answering is to prompt the MLLMs to answer a question about the input diagram.

Diagram-to-Code Editing (D2C-E). This task is to prompt the MLLM to generate the TiKZ code that can produce the rendered diagram meeting with the modification requirement. An example of the prompt is "Convert the diagram in the image into a complete, executable LaTeX code block, applying the following modification: convert the line chart to a scatter plot by removing the lines connecting the data markers. Start with `\documentclass[tikz,bor` `der=5pt]{standalone}\usepackage{pgfplots}...`" The second row of Fig. 3 gives an example.

Diagram question answering (DQA). This task is to prompt the MLLM to answer a question about the input diagram. An example of the prompt is "Which tester category has the maximum value across both the OldSystem and New System datasets?" and the expected answer is "T4". The third row of Fig. 3 illustrates this task.

3.3 Task Construction

Overview. For D2C-P, we use fixed task templates with placeholders instantiated directly from the original diagrams (the first row of Fig. 3). For D2C-E and DQA task generation, we design an agentic pipeline, as shown in the third column of Fig. 2. Given diagrams, TikZ source codes, and task-specific templates as inputs, an agent generates corresponding tasks.

The generation pipeline for D2C-E includes: (1) Template predefinition for each domain; (2) Given diagram, agent select two tasks from the templates of the corresponding domain; (3) Generate the answer from models; (4) Agent-check

Table 3: Definition and examples of D2C-Editing Task in Diagram-MMU.

Domain	Editing Dim.	Definition	Example
Charts	Text	Modify labels, tick marks, or legend text.	Rename label speed to velocity.
	Color	Change fill, stroke, or text color of elements.	Change bar color to blue/legend text to red.
	Scope	Add or remove data series or elements.	Move the legent to the top-left.
	Layout	Modify chart type or filter data.	Convert bar chart to line chart/remove values below 10.
Planar Geom.	Text	Modify point labels or annotations.	Change vertices from ABC to DEF.
	Color	Change line or region color.	Change the line AB to red.
	Scope	Add, delete, or rotate geometric elements.	Add midpoint M on segment AB.
	Layout	Add construction lines or auxiliary geometry.	Construct the circumcircle of triangle ABC.
3D Shapes	Text	Modify labels of vertices or faces.	Rename edge AB to MN.
	Color	Change the color of edges or regions.	Highlight the cross-section polygon in green.
	Scope	Add/delete solid elements.	Add the altitude from A to plane BCD.
	Layout	–	–
Graph	Text	Modify node or edge labels.	Rename node A to Start.
	Color	Change node or edge color.	Highlight shortest path edges in red.
	Scope	Add or delete nodes or edges.	Add an edge between node B and C.
	Layout	Rearrange layout or extract subgraph.	Re-layout graph using circular layout.
Chemistry	Text	Modify atom labels, charges, or subscripts.	Change CH_3 to CH_2.
	Color	Change atom or bond color.	Highlight oxygen atoms in red.
	Scope	Add or delete bonds or atoms.	Add a double bond between C and O.
	Layout	–	–
Circuit	Text	Modify component labels or values.	Rename the label from V_in to V_Source.
	Color	Change wire or component color.	Highlight power supply line in red.
	Scope	Swap or reposition components.	Move capacitor C1 to the right of R2.
	Layout	Modify circuit topology or branch structure.	Convert a series RLC circuit to a parallel RLC circuit.

and modification; (5) Human-check and modification. The DQA task generation pipeline is similar to that of D2C-E; only task templates differ.

We provide the detailed agentic pipeline in Appendix §B.2, and all templates in Appendix §B.4 & §B.6. To guarantee quality, we conduct a rigorous manual review: all samples are manually reviewed by 13 graduate students, where each annotator's assigned samples are cross-validated by another annotator to ensure that the language is clear and unambiguous, and that the question is answerable with logically consistent options and a correct designated answer.

Question types. D2C-P has one standard question type following its task definition: parsing the diagram into the corresponding TikZ code. D2C-E also follows a standard editing format: prompting models to generate the corresponding TikZ code conditioned on the changes, but we cover four editing dimensions, *e.g.*, text, color, scope, and layout. Example illustrations are shown in Table 3.

We construct two types of questions in DQA: descriptive and reasoning, with a total of 60 templates across the 6 domains (see Appendix §B.6 for details). Descriptive questions assess the model's capability in extracting and aggregating basic information from diagrams, where the correct answer must strictly adhere to domain-specific knowledge, *e.g.*, two nodes connected by a line represent a geometric segment in planar geometry but a single bond between atoms in chemistry. Reasoning questions evaluate the model's ability to perform numerical computation, where the correct answer requires applying domain-specific formulas and laws, *e.g.*, Ohm's law for circuit analysis, Euclidean formulas for geometric area computation, or solid geometry formulas for polyhedron volume.

Descriptive questions, constructed from 23 templates, require: (1) identifying symbols and markers per domain (*e.g.*, zigzag symbols as resistors in circuits, bond types and functional groups in chemistry, right-angle squares in geometry); (2) aggregating diagram information to count elements (*e.g.*, vertices in geometry, ticks/legends in charts, atoms/functional groups in chemistry) and to recognize typology and data patterns (*e.g.*, node degree in graphs, increase/decrease trends in charts). See Appendix §B.7 for data illustrations.

Reasoning questions include standard questions (formed by 18 templates) and what-if questions (formed by 19 templates). The what-if questions require the model to predict answer conditioned on a specific element modification, *e.g.*, "if the data value changes to X, what is the mean of the data series?" All questions require: (1) numerical computation by applying domain-specific formulas (*e.g.*, min, max, and median in charts; area and perimeter in geometry); (2) visual reasoning over diagram (*e.g.*, path length and height of tree in graphs; molecular properties from bond structure in chemistry). Our diagram QAs require only domain semantics/laws and numerical computation, rather than advanced expert-level knowledge [22, 48] (see Appendix §B.8 for per-domain examples).

3.4 Evaluation Metrics

Diagram-to-Code Parsing and Editing. We evaluation the diagram-to-code parsing and editing tasks from the three aspects: image, code, and object.
Image-based. We compile both generated and ground-truth TikZ code into images and compare them with four classical metrics, following papers [19, 29]:
(1) SSIM [38] compares brightness, contrast, structure patterns (higher is better);
(2) CLIP Score [28] measures semantic similarity between two images in a shared vision-language space (higher is better);
(3) LPIPS [49] uses deep features from a VGG network to measure perceptual distance in a way that matches human judgement (lower is better);
(4) FID [17] compares the distribution of generated and original images using inception network features (lower is better).

Together, these metrics assess whether the rendered output looks visually similar to the input diagram.
Code-based. We use CrystalBLEU [11] to measure the similarity between generated and ground-truth TikZ code, as in [5]. This focuses on how well the model reproduces the correct coding syntax and structure.
Object-based. We compile both generated and ground-truth TikZ code into DVI and convert them to SVG, from which we extract a Semantic Object Model (SOM)—a structured list of typed, attributed elements (*e.g.*, nodes, paths, text labels, data series). We then organize them into type, text, color, and BBox (see Appendix §C.1 for detailed extraction process). Finally, we compute four F1 scores between the generated and ground-truth SOMs, as follows:
(1) $F1_{type}$ measures whether the generated diagram contains the correct types of elements (*e.g.*, node:circle and node:rectangle);
(2) $\mathrm{F1}_{\mathrm{text}}$ evaluates exact string matching of text labels and numeric values;

Table 4: Evaluation settings on Diagram-MMU. TikZ search tool is built as an MCP server and prompts are abbreviated (full versions in Appendix §E).

ID	Settings	Abbreviated Prompts	Capability
S1	Diagram-to-code parsing	Direct coding	Foundational
S2	+ Objects	Use the provided perception data for coding	Agentic
S3	+ TikZ search tool	Use search tool for TikZ syntax and examples when necessary	Agentic
S4	+ Objects (model-gen.)	First perceive basic obejcts in diargam, then generate code	Agentic
S5	+ S2&S3	Plan where to call tool in synergy with perception data	Agentic
S6	Diagram-to-code editing	Direct editing	Foundational
S7	+ Objects	Use the provided perception data for editing	Agentic
S8	+ TikZ search tool	Use search tool for TikZ syntax and examples when necessary	Agentic
S9	+ TikZ codes (required)	First generate TikZ code of diagram, then edit code	Agentic
S10	+ TikZ codes (optional)	Plan which diagrams require TikZ code generation for editing	Agentic
S11	+ S7&S9	Plan which diagrams require TikZ code and where to call tool for editing	Agentic
S12	Diagram question answering	Direct answering	Foundational
S13	+ Objects	Use the provided perception data for answering	Agentic
S14	+ TikZ codes (required)	First generate TikZ code of diagram, then answer question	Agentic
S15	+ TikZ codes (optional)	Plan which diagrams require TikZ code generation for answering	Agentic
S16	+ TikZ search tool&S14	Plan which diagrams require TikZ code and where to call tool for answering	Agentic

(3) $F1_{color}$ compares element colors via permutation-based assignment using CIEDE2000 perceptual distance [42];
(4) $F1_{bbox}$ assesses spatial positioning accuracy via Intersection-over-Union (IoU ≥ 0.3) between element bounding boxes.

Together, these four dimensions evaluate whether the model faithfully reconstructs the semantic structure, content and layout of diagram objects. The formal formulation of F1 score per dimension is provided in Appendix §C.2. $F1_{avg}$ is the average across the four dimensions: $\frac{1}{4}\left(\mathrm{F1}_{type} + \mathrm{F1}_{text} + \mathrm{F1}_{color} + \mathrm{F1}_{bbox}\right)$.

For the editing task, we further split the code- and object-based metrics into two parts: preserve-only, computed over elements that should stay the same, and edit-only, computed over elements that the instruction asks to change.

Diagram Question Answering. Following [50], we evaluate diagram question answering using accuracy. We use Qwen3-Next-80B-A3B-Instruct to extract the answer and assign binary scores (0 for incorrect, 1 for correct). The grading prompt is provided in Appendix §C.3.

4 Evaluation Scheme

In real scientific writing workflows, MLLMs working with diagrams need foundational abilities, *e.g.*, perception, coding, domain knowledge, and reasoning, but also the ability to know when and how to act (*a.k.a* agentic capability): they parsing a diagram may first consult TikZ documentation for unfamiliar syntax (tool use), leverage perceptual objects in the diagram to write TikZ code (context utilization), build on generated diagram code to edit (state management), or decide which of these steps to invoke or in what order (planning). Diagram-MMU is, to our knowledge, the first benchmark to provide 16 flexible control settings evaluating both foundational and agentic capability of current MLLMs (Table 4). Specifically, the four levels of agentic ability are defined as: (1) Context utilization: whether models can leverage task-relevant information from context;

(2) Tool use: knowing when to invoke tools, what to query, and how to incorporate the results; (3) State management: whether models can build on prior outputs incrementally toward the final goal; (4) Planning: deciding which ability is needed and how to combine them for task solving [8, 16, 33, 44].

For tool use evaluation, we build our TikZ search tool as an MCP server. Naïve approaches such as web search or loading full PDF manuals introduce noise: browsers may return forum threads and advertisements, while complete manuals (*e.g.*, the PGFPlots reference spans ∼560 pages) cause context rot. We use Mintlify[1] to generate an MCP server from curated TikZ documentation, enabling selective access to relevant references instead of ingesting entire documents. MCP serves as a unified tool interface, allowing any MLLM to query syntax references without model-specific implementations (details in Appendix §F).

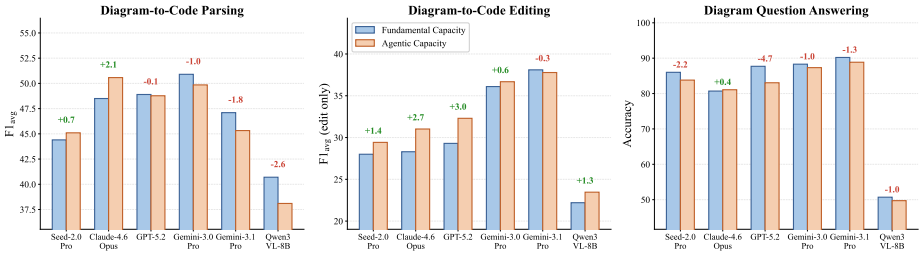


Fig. 4: Foundational *vs.* agentic performance across three tasks. Green/red annotations indicate gains/drops from foundational to agentic. Current models reason well over diagrams but struggle with visual perception and coding under foundational evaluation. Agency improves editing more than parsing, likely because textual instructions guide tool and context use, but nearly all models degrade on DQA.

We do not claim Diagram-MMU evaluates the full spectrum of agentic ability; as a pilot benchmark for evaluating agentic ability in vibe writing w.r.t. diagram-to-code and understanding tasks, our coverage targets only certain aspects. We discuss limitations in Appendix §H and hope to inspire future efforts to advance more thorough agentic evaluation dimensions in the vibe writing workspaces.

Results. Fig. 4 reports $F1_{avg}$ for D2C-P, $F1_{avg}$ (edit-only) for D2C-E, and accuracy for DQA (see §5 for detailed results and analysis.)

Foundational capacity: current models show stronger capability on reasoning but weaker ability on perception and coding, revealing an interesting asymmetry: they excel in knowledge-intensive evaluations but still struggle to link visual objects to coding skills. In short, models can reason deeply over diagrams but fail to code over them– a visual understanding weakness aligned with the findings of BabyVision [10]. Overall, Gemini-3.0 Pro achieves the most balanced profile.

Agentic capacity: Across the coding tasks, closed-source models show stronger agency on D2C-E than D2C-P. For example, GPT-5.2 drops -0.1 on D2C-P but

[1] `https://mintlify.com`

Table 5: Main results on foundational capability. Object-level metric is $F1_{avg}$ (%), with preserve-only (p) and edit-only (e) split for diagram-to-code editing. Code-level metric is CrystalBLEU (%). Image-level is measured by the average of SSIM and CLIP (SC, %) and the average of FID and LPIPS (FL, %); ↓ means lower is better. *All* averages $F1_{avg}$, CBLEU, and SC (higher is better). Diagram question answering (DQA) uses accuracy (Acc., %). **Bold** is best in class; underline is second best.

Model	Diagram-to-Code Parsing (D2C-P)					Diagram-to-Code Editing (D2C-E)								DQA
	All	$F1_{avg}$	CBLEU	(SC / FL↓)		*All*	$F1_{avg}$ (p / e)		CBLEU (p / e)		(SC / FL↓)			Acc.
Close-Source Multimodal Large Language Models														
Gemini-3.1 Pro	48.38	49.94	**32.88**	(62.33	**4.05**)	51.22	(60.60	**40.84**)	(42.02	**2.98**)	(66.47	**2.52**)		86.29
Gemini-3.0 Pro	**53.28**	**54.64**	32.76	(72.44	4.61)	**53.60**	(**65.12**	40.24)	(**44.77**	2.34)	(**72.38**	3.12)		**86.46**
Gemini-3.0 Flash	50.97	51.19	30.67	(71.04	4.79)	49.01	(59.29	37.24)	(42.21	2.17)	(67.44	2.98)		84.07
GPT-5.2	51.88	51.35	28.81	(**75.48**	6.56)	46.15	(55.59	31.15)	(38.75	1.15)	(65.33	4.55)		81.67
Claude-4.6 Opus	52.41	52.23	31.27	(73.73	7.11)	51.18	(62.56	31.95)	(43.59	1.42)	(71.78	4.61)		68.75
Seed-2.0 Pro	50.89	47.57	32.56	(72.54	7.55)	43.73	(53.03	30.14)	(37.49	1.41)	(61.86	5.34)		75.90
Open-Source Multimodal Large Language Models														
Qwen3.5-397B-A17B	52.72	51.38	32.92	(73.84	5.73)	50.82	(62.75	36.35)	(**43.66**	1.91)	(**70.43**	3.76)		**83.42**
Qwen3-VL-235B-A22B	55.98	55.11	**37.92**	(74.90	6.69)	**52.25**	(**63.79**	33.38)	(43.05	1.85)	(70.07	4.57)		63.60
Kimi-K2.5	**56.97**	**57.48**	36.13	(**77.31**	**5.20**)	52.13	(63.12	**36.52**)	(42.11	**2.02**)	(69.88	**3.17**)		79.74
Qwen3-VL-8B	48.26	44.66	33.31	(66.81	8.23)	19.00	(21.26	9.95)	(16.29	0.51)	(26.94	9.50)		47.12
InternVL3-38B	41.13	31.47	30.91	(61.02	10.14)	38.85	(44.72	22.69)	(33.13	1.29)	(57.95	7.02)		56.39
TikZero+ 10B	25.30	15.43	17.19	(43.29	12.67)				–					–

gains +3.0 on D2C-E; we hypothesize that the textual editing instructions help guide models to know when to use tools and what context is useful for target editing. For DQA, unlike direct answering (input diagram → answer), in the agentic setting the model plans to invoke diagram-to-code parsing when needed (input diagram → TikZ code → answer) and decides what to query in the TikZ search tool. Except for Claude-4.6 Opus (+0.4), all models degrade on DQA, exposing weak planning ability for complex tasks, but more critically, revealing that models cannot effectively map codes to high-level reasoning.

5 Experiments

Evaluated Models. We evaluate 12 closed-source and open-source MLLMs on Diagram-MMU. Closed-source (6): Gemini-3.1 Pro [15], Gemini-3.0 Pro [14], Gemini-3.0 Flash [13], GPT-5.2 [25], Claude-4.6 Opus [2], and Seed-2.0 Pro [9]; Open-source (6): five general-purpose models, *e.g.*, Qwen3.5-397B-A17B [37], Qwen3-VL-235B-A22B [3], Kimi-K2.5 [35], Qwen3-VL-8B [3], and InternVL3-38B [51], and one specialist TikZero+ 10B [4] which is fine-tuned on 456K diagram–TikZ code pairs. TikZero+ 10B is not trained on editing or question answering samples, so we evaluate it only on the diagram-to-code parsing task. Generation configurations of models are provided in Table D.1.

5.1 Main Results

Fundamental Capacity. Table 5 presents the main results of fundamental capabilities under pass@1 evaluation (average over 6 types of diagrams; per-type results in Appendix §G.1. Pass@k evaluation across three tasks is shown in

Fig. 5, where models improve with increasing k, with the largest relative gain from pass@1 to pass@2, and saturate when k exceeds 4. Table 5 reports $\mathrm{F1}_{\mathrm{avg}}$ for D2C-P and D2C-E (per object F1 score in Appendix § G.2).

(1) A larger performance gap among open-source models in object perception. D2C-P requires models to perceive basic objects and integrate them into code. Image-based metrics measure visual appearance similarity; all general-purpose models score highest here, and the gap between closed- and open-source models is small (*e.g.*, closed-source models achieve 62.33–75.48 and open-source models 61.02–77.31). At coding syntax and structure evaluation (CrystalBLEU), generic models score lowest, with a similarly small gap (closed-source: 28.81–32.88; open-source: 30.91–37.92). However, a larger gap emerges at object perception: closed-source $\mathrm{F1}_{\mathrm{avg}}$ clusters within 47.57–54.64, while open-source spans 31.47–57.48, and the specialist TikZero+ trails far behind at only 15.43. This means a generated diagram may appear visually plausible at image level yet contain incorrect identification at object level.

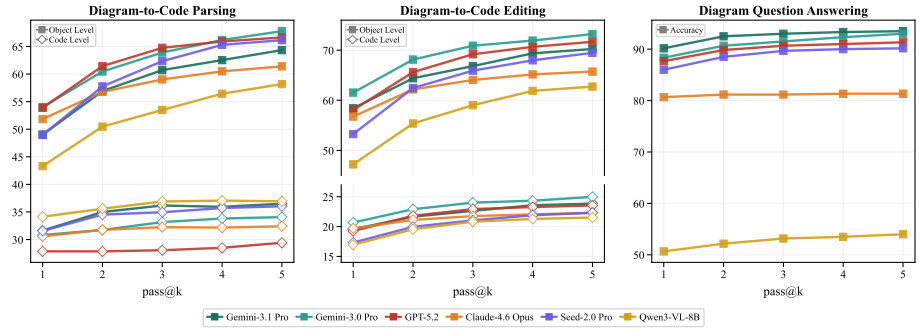


Fig. 5: Pass@k evaluation across three tasks: performance improves with k, with gains peaking from pass@1 to pass@2 and saturating beyond $k = 4$.

(2) Models follow editing instructions but struggle to produce matching code. In the editing task, models reach only 0.51–2.98 CrystalBLEU on the edit partitions, yet achieve relatively higher object F1 scores, indicating they may generate the correct objects in line with the editing instructions but fail to place them at the correct positions or formats to match the ground truth code.

(3) Models failing on D2C-P also struggle with D2C-E and DU. We split diagrams into two sets conditioned on whether D2C-P produces executable TikZ code (Fig. 6). On diagrams where D2C-P succeeds, models consistently achieve higher D2C-E scores than on D2C-P failures: the object $\mathrm{F1}_{\mathrm{avg}}$ (average of preserve-only&edit-only) gap between the two sets reaches 20–40 points across most models and code lengths. The similar pattern holds at code-level metrics. Even models can produce executable TikZ code on D2C-P failed-rendering diagrams during editing; this does not yield accurate editing. We hypothesize that

such executable TikZ results from textual editing instructions, which guide code generation, *e.g.*, leveraging the model's stronger text-to-code ability [36, 47].

(4) Models struggle most with 3D shapes across all three tasks. 3D shapes pose the greatest challenge for both closed- and open-source models across all three tasks, likely due to the scarcity of TikZ-3D training samples in pretraining corpora. For D2C-P and D2C-E, charts are the second most challenging domain after 3D, as they contain a higher density of small-scale elements (*e.g.*, tick labels, data markers). Model-specific DQA weaknesses diverge beyond 3D: Seed-2.0 Pro drops notably on charts; Claude-4.6 Opus underperforms on graph structures; and Gemini-3.0 Pro shows relatively lower accuracy on circuits (see Table G.3 in Appendix §G.3).

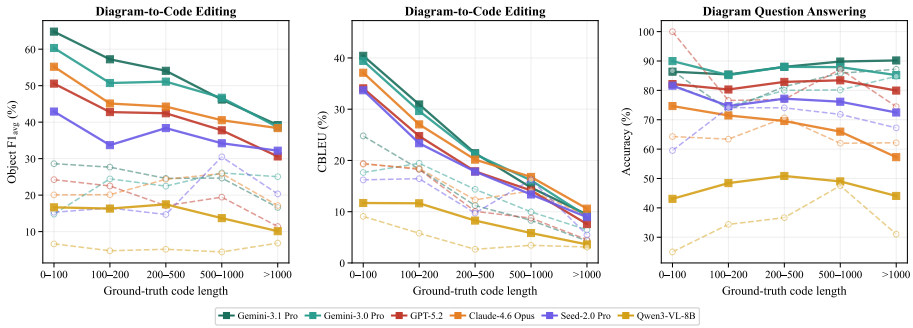


Fig. 6: Model performance on D2C-E/DQA across two diagram sets split by D2C-P output: successful rendering (solid lines) *vs.* failed rendering (dashed lines). D2C-E is evaluated by object $F1_{avg}$ (left) and CrystalBLEU (middle), and DQA by accuracy (right). The x-axis denotes ground-truth code length. Models that fail D2C-P also score lower on D2C-E and DQA.

Agentic Capacity. We select a **mini** split of 300 diagrams (50 per domain) with 1,500 QA instances (300/600/600 for D2C-P/D2C-E/DQA), evaluated on 6 representative models: Seed-2.0 Pro, Claude-4.6 Opus, GPT-5.2, Gemini-3.0 Pro, Gemini-3.1 Pro, and Qwen3-VL-8B.

(1) Context utilization. We provide diagram objects as context across all three tasks (Tables 6 and 7). On D2C-E, objects improve all models consistently: $F1_{avg}$ gains 4.1–10.6 on preserve and 3.3–7.1 on edit; thus objects may help models locate which elements to keep and which to modify. On D2C-P, results are mixed: four models gain, while Gemini-3.0 Pro (-2.0) and Qwen3-VL-8B (-6.1) degrade. At code level, only Claude-4.6 Opus (+0.3) and Gemini-3.1 Pro (+1.4) gain, showing objects aid element perception more than code synthesis. On DQA, most models benefit, while GPT-5.2 drops sharply (-7.3), indicating it cannot map low-level objects to semantic reasoning.

Takeaway: Most models can use objects to ground local edits but struggle to integrate them into syntax or domian knowledege reasoning.

Table 6: Main results on agentic evaluation settings across D2C-P (S1–S5) and DQA (S12–S16). S1/S12 uses direct coding/answering; **Context utilization**: S2/S13 provide diagram objects as context; **Tool use**: S3 provides TikZ search tool; **State management**: S4 perceives objects first, then builds upon them to code; S14 requires answering building upon the coding states; **Planning**: S5 coordinates objects and tool; S15 optionally generates code before answering; S16 adds tool access to S15 (Table 4). Δ (↑gain, ↓drop) over direct coding/answering S1/S12.

Model	S1		S2		S3		S4		S5		S12	S13	S14	S15	S16
	$F1_{avg}$	CBLEU	$F1_{avg}$	CBLEU	$F1_{avg}$	CBLEU	$F1_{avg}$	CBLEU	$F1_{avg}$	CBLEU	Acc.	Acc.	Acc.	Acc.	Acc.
Seed-2.0 Pro	44.4	31.6	↑0.2	↓0.2	↑1.2	↑1.4	↓1.2	↓1.2	↑2.6	↑1.5	86.0	↑1.0	↑0.8	↓1.8	↓8.8
Claude-4.6 Opus	48.5	30.6	↑1.2	↑0.3	↑2.2	↑1.9	↓0.2	↓0.9	↑5.1	↑2.8	80.7	0.0	↑3.8	↓2.0	↓0.3
GPT-5.2	48.9	27.9	↑2.9	↓2.9	↓0.3	↓0.5	↓1.5	↓0.8	↓1.6	↓4.1	87.7	↓7.3	↓1.3	↓4.3	↓5.7
Gemini-3.0 Pro	50.9	30.8	↓2.0	↓3.2	↓1.0	↓0.6	↓1.3	↓1.2	↑0.1	↓0.5	88.3	↑1.2	↓2.5	↓2.3	↓0.3
Gemini-3.1 Pro	47.1	31.6	↑1.9	↑1.4	↓4.3	↓3.0	↑0.3	↓0.8	↓5.0	↓4.5	90.2	↑0.5	↓3.2	↓2.7	0.0
Qwen3-VL-8B	40.7	34.1	↓6.1	↓5.6	↓0.1	↓1.2	↑1.4	↓0.2	↓5.6	↓6.4	50.7	↑3.0	↓1.3	↓2.3	↓3.3

(2) Tool use. Most models exhibit weak tool-calling ability. The sharpest failure is Gemini-3.1 Pro (-4.3 $F1_{avg}$ on D2C-P). Manual inspection of trajectory reveals that model iteratively queries without a clear stopping criterion, causing context rot until terminated by the token-length limit (tool-call counts and failure rates in Appendix §G.4); Qwen3-VL-8B trajectory issues poorly targeted queries, lacking ability of what to query in the search tool. Claude-4.6 Opus performs best, with consistent gains on both DC (+2.2 $F1_{avg}$) and DE (+4.4/+2.2).

Takeaway: Only Claude-4.6 Opus consistently benefits from tool access; Gemini-3.1 Pro suffers from excessive retrieval; Qwen3-VL-8B fails at querying.

(3) State management. We require models to first generate the diagram's TikZ code, then build upon it for editing (D2C-E) or answering (DQA). On D2C-E, most models degrade on preserve $F1_{avg}$, indicating the limitation of managing intermediate code states for precise editing. Claude-4.6 Opus (+2.5/+2.6) and GPT-5.2 (+2.5/+3.5) are exceptions, successfully building on their generated code. On DQA, Claude-4.6 Opus gains substantially (+3.8) and Seed-2.0 Pro gains marginally (+0.8), while all other models degrade (-1.3 to -3.2).

Takeaway: Most models fail to manage intermediate code states across steps; only Claude-4.6 Opus maintains coherence from coding to editing&reasoning.

(4) Planning. We test whether models can coordinate multiple agentic abilities: combining objects with tool use for D2C-P, optionally generating code before editing or answering, and coordinating tool use with optional code generation (D2C-E and DU). On D2C-P, Claude-4.6 Opus (+5.1/+2.8) and Seed-2.0 Pro (+2.6/+1.5) coordinate both resources successfully, while (-1.6/-4.1), Gemini-3.1 Pro (-5.0/-4.5), and Qwen3-VL-8B (-5.6/-6.4) degrade. On D2C-E, most models show marginal or negative changes under optional code generation; adding tool access recovers Claude-4.6 Opus (+4.1/+3.0) and GPT-5.2 (+2.6/+1.9), but degrades Gemini-3.1 Pro (-4.4/-0.9). On DQA, all six models degrade under optional code generation, and adding tool access amplifies degradation for Seed-2.0 Pro (-8.8) and GPT-5.2 (-5.7).

Takeaway: Planning is the weakest agentic capability; no model reliably composes multiple abilities as task complexity grows from D2C-P to DQA.

Table 7: Agentic evaluation results across D2C-E (S6–S11) with preserve (p) / edit (e) split. S6 uses direct editing; **Context utilization**: S7 provides diagram objects as context; **Tool use**: S8 provides TikZ search tool; **State management**: S9 requires editing building upon the coding states; **Planning**: S10 optionally generates code for editing; S11 adds tool access to S10 (Table 4). Δ (↑gain, ↓drop) over direct editing.

Model	S6		S7		S8		S9		S10		S11	
	$F1_{avg}$(p/e)	CBLEU(p/e)	$F1_{avg}$(p/e)	CBLEU(p/e)	$F1_{avg}$(p/e)	CBLEU(p/e)	$F1_{avg}$(p/e)	CBLEU(p/e)	$F1_{avg}$(p/e)	CBLEU(p/e)	$F1_{avg}$(p/e)	CBLEU(p/e)
Seed-2.0 Pro	50.6/28.0	34.3/1.5	↑9.3/↑7.1	↑6.6/↑0.5	↑0.7/↑0.2	↑0.7/↑0.1	↓2.3/↓0.4	↓1.3/↑0.1	0.0/↓0.4	↑0.1/↓0.3	↑0.6/↑0.6	↑0.1/0.0
Claude-4.6 Opus	55.9/28.3	39.3/1.3	↑8.9/↑6.0	↑6.0/0.0	↑4.4/↑2.2	↑4.7/↑0.5	↑2.5/↑2.6	↑2.5/0.0	↓1.5/↓0.2	↓0.4/↑0.1	↑4.1/↑3.0	↑5.6/↑0.7
GPT-5.2	54.9/29.3	38.6/1.2	↑10.6/↑5.9	↑7.2/↑0.1	↑1.3/↑2.6	↑2.4/0.0	↑2.5/↑3.5	↑2.7/↑0.1	↑0.9/↑1.1	↑1.6/↑0.3	↑2.6/↑1.9	↑2.9/↑0.4
Gemini-3.0 Pro	58.8/36.1	40.9/1.9	↑4.1/↑3.3	↑2.4/↑0.4	↑2.2/↓0.1	↓0.1/↑0.5	↓1.1/↓0.6	↓2.0/↑0.2	↓0.4/↓0.5	↓1.7/↑0.2	↑2.8/↑0.8	↑0.7/↑0.5
Gemini-3.1 Pro	57.6/38.1	38.0/2.5	↑6.6/↑4.8	↑4.6/↑0.6	↓8.5/↓3.5	↓5.3/↑0.8	↓5.2/↓3.7	↓2.9/↑0.2	↑0.6/↑1.7	↑0.5/↑0.7	↓4.4/↓0.9	↓2.9/↑1.4
Qwen3-VL-8B	47.0/22.2	34.2/0.8	↑8.4/↑3.3	↑4.3/↑0.2	↓2.0/↓0.2	↓2.2/↑0.2	↓1.0/↑2.5	↓1.4/↑0.4	↓3.3/↓1.0	↓3.0/0.0	0.0/↑1.7	↓0.9/↑0.3

6 Conclusion

Scientific diagrams are an important visual medium for expressing abstract ideas, and their TikZ code representation can be natively compiled inline within LaTeX during scientific paper writing. Vibe writing platforms provide MLLM-assisted paper writing, and one important feature is converting scientific diagrams directly into TikZ code. We present Diagram-MMU, a TikZ-based benchmark with 16 evaluation settings across diagram-to-code parsing, editing, and understanding alongside agentic settings per task. We also build a TikZ search tool as an MCP server that any model can selectively query TikZ syntactic references without any model-specific implementations. Comprehensive evaluation of 12 MLLMs reveals their weaknesses in coding diagrams into TikZ, and stronger agentic abilities, *e.g.*, context utilization and tool use, could help. We hope our analysis can inspire further research to build more thorough evaluation settings and develop methods to improve MLLM-assisted vibe writing workspaces.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No. 62425603) and Basic Research Program of Jiangsu Province (Grant No. BK20240011).

We thank all 13 annotators for data quality assurance; together, they cross-validated over 2,000 scientific diagrams across six domains. Three annotators are co-authors. The ten non-author contributors are one NetMind.ai research scientist, eight PhD students at Nanjing University of Science and Technology (NJUST), and one PhD student at Westlake University. Domain abbreviations: 3D = 3D Shape, Chem. = Chemical Expressions, Graph = Graph Structures, Planar = Planar Geometry, Circuit = Circuit Diagrams. Contributions (cases; domains): Xinyu Miao (NJUST; 419; 3D, Charts, Chem., Graph, Planar), Yiyou Gao (NJUST; 331; 3D, Charts, Graph, Planar), Ruolin Wang (NJUST; 271; 3D, Charts, Graph, Planar), Xinyao Hu (NetMind.ai; 208; 3D, Charts, Graph, Planar), Jinhong Yang (NJUST; 200; Graph), Zongxin Zhu (NJUST; 194; Charts, Graph, Planar), Yu Wang (NJUST; 181; Graph, Planar), Xinyu Zhang (NJUST; 170; Charts, Circuit, Graph), Xueji Fang (Westlake University; 107; 3D, Charts, Graph, Planar), and Wei Tang (NJUST; 11; 3D, Charts).

References

1. Anthropic: Model context protocol. https://modelcontextprotocol.io (2024), accessed: 2026-06-27
2. Anthropic: Claude opus 4.6. https://www.anthropic.com/claude/opus (2026), accessed: 2026-06-27
3. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)
4. Belouadi, J., Ilg, E., Keuper, M., Tanaka, H., Utiyama, M., Dabre, R., Eger, S., Ponzetto, S.: Tikzero: Zero-shot text-guided graphics program synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17793–17806 (2025)
5. Belouadi, J., Lauscher, A., Eger, S.: Automatikz: Text-guided synthesis of scientific vector graphics with tikz. In: International Conference on Learning Representations (2024)
6. Belouadi, J., Ponzetto, S., Eger, S.: Detikzify: Synthesizing graphics programs for scientific figures and sketches with tikz. Advances in Neural Information Processing Systems **37**, 85074–85108 (2024)
7. Bo, W., Zhang, S., Sun, Y., Wu, J., Xie, Q., Tan, X., Chen, K., He, W., Li, X., Zhao, N., Wang, J., Li, Z.: Vilomem: Agentic learner with grow-and-refine multimodal semantic memory. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5476–5486 (June 2026)
8. Buyya, R., et al.: Agentic artificial intelligence (ai): Architectures, taxonomies, and evaluation of large language model agents. arXiv preprint arXiv:2601.12560 (2026)
9. Bytedance Seed: Seed2.0 model card: Towards intelligence frontier for real-world complexity. https://lf3-static.bytednsdoc.com/obj/eden-cn/lapzild-tss/ljhwZthlaukjlkulzlp/seed2/0214/Seed2.0%20Model%20Card.pdf (2026), accessed: 2026-06-27
10. Chen, L., Xie, W., Liang, Y., He, H., Zhao, H., Yang, Z., Huang, Z., Wu, H., Lu, H., Bao, Y., et al.: Babyvision: Visual reasoning beyond language. arXiv preprint arXiv:2601.06521 (2026)
11. Eghbali, A., Pradel, M.: Crystalbleu: precisely and efficiently measuring the similarity of code. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. pp. 28:1–28:12 (2022)
12. Fu, C., Dai, Y., Luo, Y., Li, L., Ren, S., Zhang, R., Wang, Z., Zhou, C., Shen, Y., Zhang, M., et al.: Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 24108–24118 (2025)
13. Google DeepMind: Gemini 3 flash model card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf (2025), accessed: 2026-06-27
14. Google DeepMind: Gemini 3 pro model card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf (2025), accessed: 2026-06-27
15. Google DeepMind: Gemini 3.1 pro model card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-1-Pro-Model-Card.pdf (2026), accessed: 2026-06-27
16. Hartung, T.: Ai, agentic models and lab automation for scientific discovery—the beginning of scaince. Frontiers in Artificial Intelligence **8**, 1649155 (2025)

17. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in Neural Information Processing Systems **30**, 6626–6637 (2017)
18. Jiang, L., Huang, S., Wu, X., Li, Y., Zhang, D., Wei, F.: Viscodex: Unified multimodal code generation via merging vision and coding models. arXiv preprint arXiv:2508.09945 (2025)
19. Li, S., Sun, J., Wang, Z., Fan, X., Li, H., Yang, D., Xi, Z., Wang, Y., Shan, Z., Gui, T., et al.: ChartE^3: A comprehensive benchmark for end-to-end chart editing. arXiv preprint arXiv:2601.21694 (2026)
20. Li, X., Wu, C., Sun, Y., Zhou, J., Qu, D., Qu, Y., Bo, W., Yu, H., Liang, D.: Fvar: Next-focus prediction for visual autoregressive modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 30391–30401 (2026)
21. Liu, Y., Duan, H., Zhang, Y., Li, B., Zhang, S., Zhao, W., Yuan, Y., Wang, J., He, C., Liu, Z., et al.: Mmbench: Is your multi-modal model an all-around player? In: European Conference on Computer Vision. pp. 216–233. Springer (2024)
22. Lu, P., Bansal, H., Xia, T., Liu, J., Li, C., Hajishirzi, H., Cheng, H., Chang, K.W., Galley, M., Gao, J.: Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In: International Conference on Learning Representations (2024)
23. Masry, A., Do, X.L., Tan, J.Q., Joty, S., Hoque, E.: Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In: Findings of the Association for Computational Linguistics: ACL 2022. pp. 2263–2279 (2022)
24. Minimax: Minimax m2.5: Built for real-world productivity. `https://www.minimax.io/news/minimax-m25` (2026), accessed: 2026-06-27
25. OpenAI: Gpt-5.2. `https://developers.openai.com/api/docs/models/gpt-5.2` (2025), accessed: 2026-06-27
26. OpenAI: Prism. `https://openai.com/prism/` (2026), accessed: 2026-06-27
27. Qiu, Z., Liu, W., Feng, H., Liu, Z., Xiao, T.Z., Collins, K.M., Tenenbaum, J.B., Weller, A., Black, M.J., Schölkopf, B.: Can large language models understand symbolic graphics programs? In: International Conference on Learning Representations (2025)
28. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning. pp. 8748–8763. PMLR (2021)
29. Roberts, J.S., Lee, T., Wong, C.H., Yasunaga, M., Mai, Y., Liang, P.: Image2struct: Benchmarking structure extraction for vision-language models. Advances in Neural Information Processing Systems **37**, 115058–115097 (2024)
30. Rodriguez, J.A., Puri, A., Agarwal, S., Laradji, I.H., Rodriguez, P., Rajeswar, S., Vazquez, D., Pal, C., Pedersoli, M.: Starvector: Generating scalable vector graphics code from images and text. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16175–16186 (2025)
31. Sun, Y., Hao, J., Zhu, K., Liu, J.J., Li, X., Zhao, N., Li, Z., Wang, J.: Enhancing descriptive captions with visual attributes for multimodal perception. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1683–1694 (2026)
32. Sun, Y., Zhang, S., Tang, W., Chen, A., Koniusz, P., Zou, K., Xue, Y., Hengel, A.v.d.: Math blind: Failures in diagram understanding undermine reasoning in mllms. arXiv preprint arXiv:2503.20745 (2025)

33. Tang, S., Wang, Y., Chen, L., Wang, Y., Peng, S., Xu, D., Ouyang, W.: Human-centric foundation models: Perception, generation and agentic modeling. arXiv preprint arXiv:2502.08556 (2025)
34. Tang, W., Sun, Y., Zhang, S., Bo, W., Li, X., Koniusz, P., Li, W., Zhao, N., Li, Z.: Artemis: Structured visual reasoning for perception policy learning. arXiv preprint arXiv:2512.01988 (2025)
35. Team, K., Bai, T., Bai, Y., Bao, Y., Cai, S., Cao, Y., Charles, Y., Che, H., Chen, C., Chen, G., et al.: Kimi k2.5: Visual agentic intelligence. arXiv preprint arXiv:2602.02276 (2026)
36. Team, K., Bai, Y., Bao, Y., Chen, G., Chen, J., Chen, N., Chen, R., Chen, Y., Chen, Y., Chen, Y., et al.: Kimi k2: Open agentic intelligence. arXiv preprint arXiv:2507.20534 (2025)
37. Team, Q.: Qwen3.5: Towards native multimodal agents. `https://qwen.ai/blog?id=qwen3.5` (2026), accessed: 2026-06-27
38. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004)
39. Wang, Z., Xia, M., He, L., Chen, H., Liu, Y., Zhu, R., Liang, K., Wu, X., Liu, H., Malladi, S., et al.: Charxiv: Charting gaps in realistic chart understanding in multimodal llms. Advances in Neural Information Processing Systems **37**, 113569–113697 (2024)
40. Wei, J., Tan, C., Chen, Q., Wu, G., Li, S., Gao, Z., Sun, L., Yu, B., Guo, R.: From words to structured visuals: A benchmark and framework for text-to-diagram generation and editing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13315–13325 (2025)
41. Wu, C., Liang, Z., Ge, Y., Guo, Q., Lu, Z., Wang, J., Shan, Y., Luo, P.: Plot2code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. In: Findings of the Association for Computational Linguistics: NAACL 2025. pp. 3006–3028 (2025)
42. Yang, C., Shi, C., Liu, Y., Shui, B., Wang, J., Jing, M., Xu, L., Zhu, X., Li, S., Zhang, Y., et al.: Chartmimic: Evaluating lmm's cross-modal reasoning capability via chart-to-code generation. In: International Conference on Learning Representations (2025)
43. Yang, D., Zhang, L., Yue, Z., Chen, L., Xu, Y., Wang, W., Jin, Q.: Chartm3: Benchmarking chart editing with multimodal instructions. In: Proceedings of the 33rd ACM International Conference on Multimedia. pp. 5001–5009 (2025)
44. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K.R., Cao, Y.: React: Synergizing reasoning and acting in language models. In: International Conference on Learning Representations (2023)
45. Yue, X., Ni, Y., Zhang, K., Zheng, T., Liu, R., Zhang, G., Stevens, S., Jiang, D., Ren, W., Sun, Y., et al.: Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9556–9567 (2024)
46. Yue, X., Zheng, T., Ni, Y., Wang, Y., Zhang, K., Tong, S., Sun, Y., Yu, B., Zhang, G., Sun, H., et al.: Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 15134–15186 (2025)
47. Zeng, A., Lv, X., Hou, Z., Du, Z., Zheng, Q., Chen, B., Yin, D., Ge, C., Xie, C., Wang, C., et al.: Glm-5: from vibe coding to agentic engineering. arXiv preprint arXiv:2602.15763 (2026)

48. Zhang, R., Jiang, D., Zhang, Y., Lin, H., Guo, Z., Qiu, P., Zhou, A., Lu, P., Chang, K.W., Qiao, Y., et al.: Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In: European Conference on Computer Vision. pp. 169–186. Springer (2024)
49. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 586–595 (2018)
50. Zhou, R., Shao, Y., Lu, H., Xing, B., Bai, T., Chen, Y., Zhao, J., Sui, L., Yao, H., Zhao, Z., et al.: Worldvqa: Measuring atomic world knowledge in multimodal large language models. arXiv preprint arXiv:2602.02537 (2026)
51. Zhu, J., Wang, W., Chen, Z., Liu, Z., Ye, S., Gu, L., Tian, H., Duan, Y., Su, W., Shao, J., et al.: Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. arXiv preprint arXiv:2504.10479 (2025)