

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

Weiha0 Bo^{1,2*} Shan Zhang^{3†} Yanpeng Sun^{4*} Jie Liu^{2,5} Yongke Yao^{2,6}
Jinhao Du⁷ Wei He² Kai Zou⁸ Zechao Li^{1†} Jingdong Wang^{2‡}

¹Nanjing University of Science and Technology ²Baidu Inc ³AIML, Adelaide University ⁴SUTD
⁵Southeast University ⁶East China Normal University ⁷University of Oxford ⁸NetMind.ai

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 L^AT_EX 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.

Date: August 13, 2026

Connection Email: shan.zhang@adelaide.edu.au, yanpeng_sun@sutd.edu.sg

Project Page: <https://vi-ocean.github.io/projects/diagram-mm>



1 Introduction

In the wake of significant advancements in large language models (LLMs) [1–3], multimodal large language models (MLLMs) [4–7] have rapidly evolved to extend their versatility across vision–language tasks [8–11]. To bring these capabilities into everyday research workflows, OpenAI Prism [12] provides a free vibe writing workspace where any researcher who writes in L^AT_EX can draft papers, generate figures, and convert scientific diagrams into TikZ code.

Scientific diagrams are the primary visual medium through which researchers convey experimental results, mathematical relationships, molecular structures, and circuit topologies [13–15]. 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

*Equal contribution †Corresponding Author ‡Project Lead

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), Diagram-to-Code Editing (D2C-E), and Diagram 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 (ChartE³). ✓ is partially supported (limited editing or text-only input). ∅ denotes the synthetic diagrams.

Benchmark	Data			Tasks				Evaluation Metrics			
	TikZ Code	#Diag. Type	#Diag.	D2C-P	D2C-E	DQA	Agentic	Object	CBLEU	Image	Answer Acc.
<i>Diagram-to-Code Parsing</i>											
ChartMimic ∅ [14]	✗	1	2.4k	✓	✓	✗	✗	✓	✓	✓	✗
InfiBench-V [16]	✗	2	0.3k	✓	✗	✗	✗	✗	✓	✓	✗
Plot2Code [17]	✗	1	0.1k	✓	✗	✗	✗	✗	✓	✓	✗
SVG-Diagrams [18]	✗	4	0.5k	✓	✗	✗	✗	✗	✗	✓	✗
DiagramGenBenchmark [19]	✓	2	0.4k	✓	✓	✗	✗	✗	✓	✓	✗
AutomaTikZ [20]	✓	4	1.0k	✓	✗	✗	✗	✗	✓	✓	✗
DeTikZify [21]	✓	4	1.5k	✓	✗	✗	✗	✗	✓	✓	✗
Image2Struct [22]	✓	4	0.3k	✓	✗	✗	✗	✗	✗	✓	✗
<i>Diagram-to-Code Editing</i>											
ChartE ³ [23]	✗	1	0.8k	✗	✓	✗	✗	✓	✗	✓	✗
ChartM ³ ∅ [24]	✗	1	1.0k	✗	✓	✗	✗	✗	✓	✓	✗
<i>Diagram Question Answering</i>											
CharXiv [13]	✗	1	2.3k	✗	✗	✓	✗	✗	✗	✗	✓
ChartQA [25]	✗	1	1.0k	✗	✗	✓	✗	✗	✗	✗	✓
SGP-Bench ∅ [26]	✗	2	3.5k	✗	✗	✓	✗	✗	✗	✗	✓
MATHEMETRIC ∅ [15]	✗	3	1.2k	✗	✗	✓	✗	✓	✗	✗	✓
Diagram-MMU (Ours)	✓	6	3.7k	✓	✓	✓	✓	✓	✓	✓	✓

(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 [7]). Moreover, in the vibe writing workspace, models also need to know how to act (agentic ability [7]): 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 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 from TikZ/L^AT_EX 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 L^AT_EX 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 [27], enabling MLLMs to selectively access

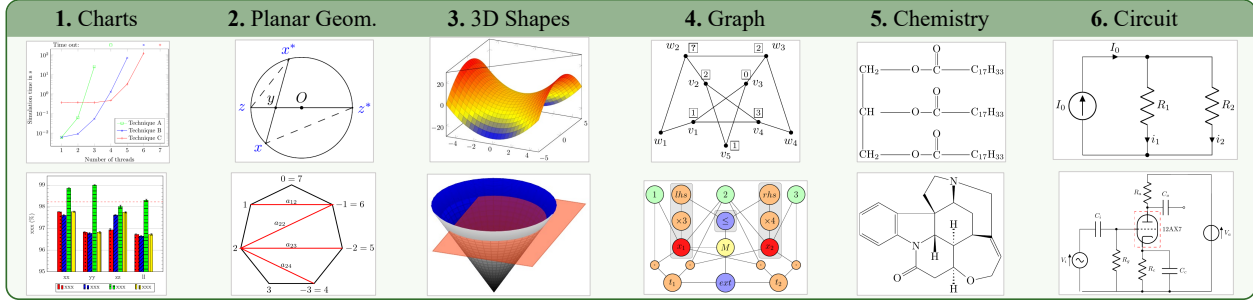


Figure 1 Demo illustration of the six types of diagrams in Diagram-MMU.

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 [13, 17, 23–25] or narrowly cover scientific diagram types [15, 18, 20, 21], with several using synthetic diagrams [15, 24, 26]. Moreover, benchmarks that adopt Python or SVG as the code representation produce standalone scripts or markup outside the $\text{\LaTeX}$ 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.

Code representation. Most coding benchmarks adopt Python, primarily for charts [14, 17, 23, 24], or SVG/HTML, for icons, fonts, emojis and web UI [16, 18]. Although these languages are common, they are not native to $\text{\LaTeX}$: they are rendered by matplotlib/CairoSVG into standalone image files rather than compiled inline within $\text{\LaTeX}$ 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

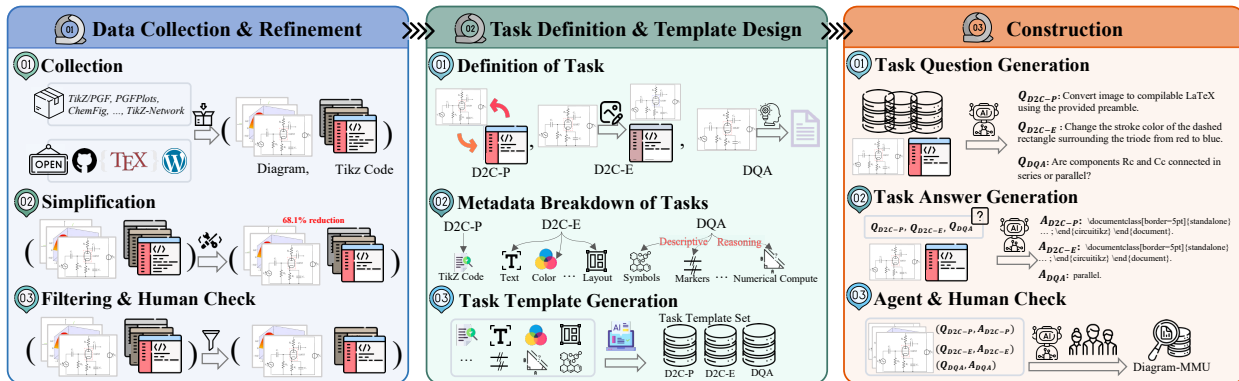


Figure 2 Task construction pipeline for Diagram-MMU.

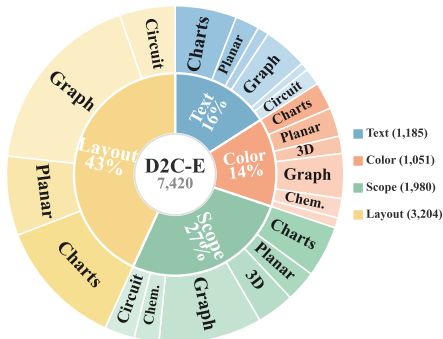
Table 2 Data statistics of Diagram-MMU. In Table 2a, TikZ.p indicates TikZ package; basic graphical objects are shared across diagrams, while domain-specific objects are summarized in the footnote; #Q denotes total questions; Knowledge is domain-specific for the DQA task. Figure 2b is the metadata breakdown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

Domain	D2C-P		DQA	
	#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) (Updated)

complexity [20]. Due to the expressiveness and diverse packages supporting science in TikZ, [20–22] build TikZ-based benchmarks, but evaluate only diagram-to-code parsing on limited diagram types. Vibe writing workspaces, e.g., Prism [12], 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. ChartE³ [23] designs local and global editing templates for charts, and CharXiv [13] 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 [28, 29] 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 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 simplified 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 rendered images by an automatic MLLM model comparison and a further manual 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 manually check the classification correctness and make adjustment if necessary. Finally, we have **3,744** diagrams. Fig. 1 presents

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 legend 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.

agent select two tasks from the templates of the corresponding domain; (3) Generate the answer from models; (4) Agent-check 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 [30, 31] (see Appendix §B.8 for per-domain examples).

3.4 Evaluation Metrics

Diagram-to-Code Parsing and Editing. We evaluate 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 [22, 23]:

- (1) SSIM [32] compares brightness, contrast, structure patterns (higher is better);
- (2) CLIP Score [33] measures semantic similarity between two images in a shared vision-language space (higher is better);
- (3) LPIPS [34] uses deep features from a VGG network to measure perceptual distance in a way that matches human judgement (lower is better);
- (4) FID [35] 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 [36] to measure the similarity between generated and ground-truth TikZ code, as in [20]. 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) $F1_{text}$ evaluates exact string matching of text labels and numeric values;
- (3) $F1_{color}$ compares element colors via permutation-based assignment using CIEDE2000 perceptual distance [14];
- (4) $F1_{bbox}$ assesses spatial positioning accuracy via Intersection-over-Union ($\text{IoU} \geq 0.3$) 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: $\frac{1}{4} (F1_{type} + F1_{text} + F1_{color} + F1_{bbox})$.

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.

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 objects in diagram, 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

Diagram Question Answering. Following [37], 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.

3.5 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 [38–41].

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 Mintlify¹ 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).

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 §4 for detailed results and analysis.)

¹<https://mintlify.com>

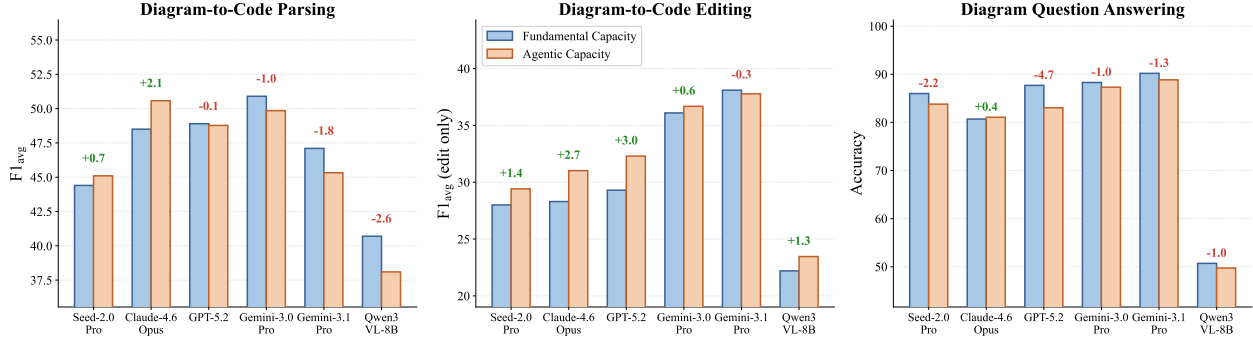


Figure 4 Foundational *vs.* agentic performance across three tasks. Green/red annotations indicate gains/drops from foundational to agentic. Current models reason well over diagrams but struggle with visual perception and coding under foundational evaluation. Agency improves editing more than parsing,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

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 perception weakness aligned with the findings of BabyVision [42]. 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 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.

4 Experiments

Evaluated Models. We evaluate 12 closed-source and open-source MLLMs on Diagram-MMU. Closed-source (6): Gemini-3.1 Pro [4], Gemini-3.0 Pro [43], Gemini-3.0 Flash [44], GPT-5.2 [6], Claude-4.6 Opus [45], and Seed-2.0 Pro [7]; Open-source (6): five general-purpose models, e.g., Qwen3.5-397B-A17B [5], Qwen3-VL-235B-A22B [46], Kimi-K2.5 [47], Qwen3-VL-8B [46], and InternVL3-38B [48], and one specialist TikZero+ 10B [49] 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.

4.1 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 $F1_{avg}$ for D2C-P and D2C-E (per object F1 score in Fig. 6 and more analysis 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:

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, %); $\downarrow$ 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
	<i>All</i>	$F1_{avg}$	CBLEU	(SC / FL $\downarrow$)		<i>All</i>	$F1_{avg}$ (p / e)	CBLEU (p / e)	(SC / FL $\downarrow$)		Acc.
Close-Source Multimodal Large Language Models											
Gemini-3.1 Pro	48.38	49.94	32.88	(62.33 4.05)	<u>51.22</u>	(60.60 40.84)	(42.02 2.98)	(66.47 2.52)			<u>86.29</u>
Gemini-3.0 Pro	53.28	54.64	<u>32.76</u>	(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 <u>2.98</u>)			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	<u>52.41</u>	<u>52.23</u>	31.27	<u>(73.73 7.11)</u>	51.18	<u>(62.56 31.95)</u>	<u>(43.59 1.42)</u>	<u>(71.78 4.61)</u>			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	<u>55.98</u>	<u>55.11</u>	37.92	<u>(74.90 6.69)</u>	52.25	(63.79 33.38)	<u>(43.05 1.85)</u>	<u>(70.07 4.57)</u>			63.60
Kimi-K2.5	56.97	57.48	<u>36.13</u>	(77.31 5.20)	<u>52.13</u>	<u>(63.12 36.52)</u>	(42.11 2.02)	(69.88 3.17)			<u>79.74</u>
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)							–

28.81–32.88; open-source: 30.91–37.92). However, a larger gap emerges at object perception: closed-source $F1_{avg}$ 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.

(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. 7). On diagrams where D2C-P succeeds, models consistently achieve higher D2C-E scores than on D2C-P failures: the object $F1_{avg}$ (average of preserve-

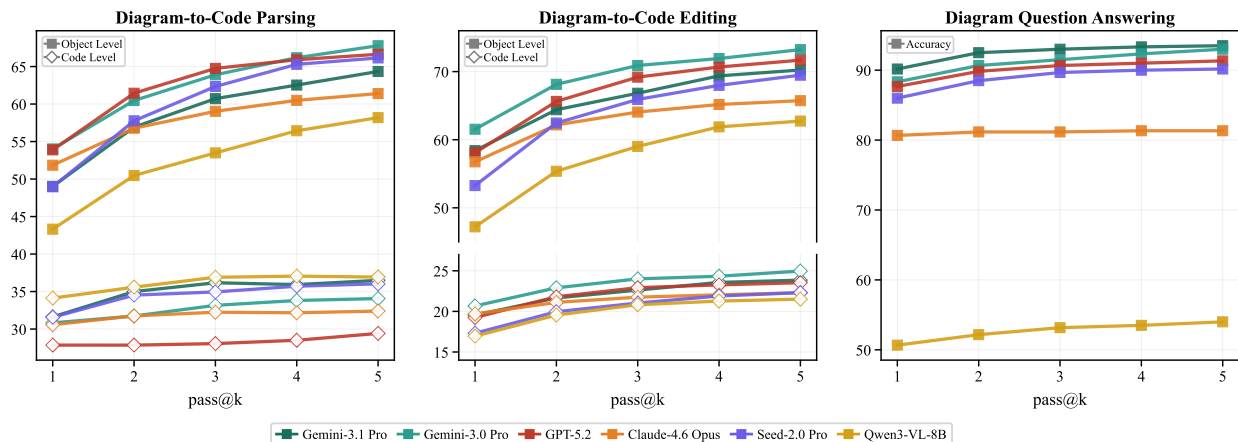


Figure 5 Pass@k evaluation across three tasks: performance improves with k , with gains peaking from pass@1 to pass@2 and saturating beyond $k = 4$.

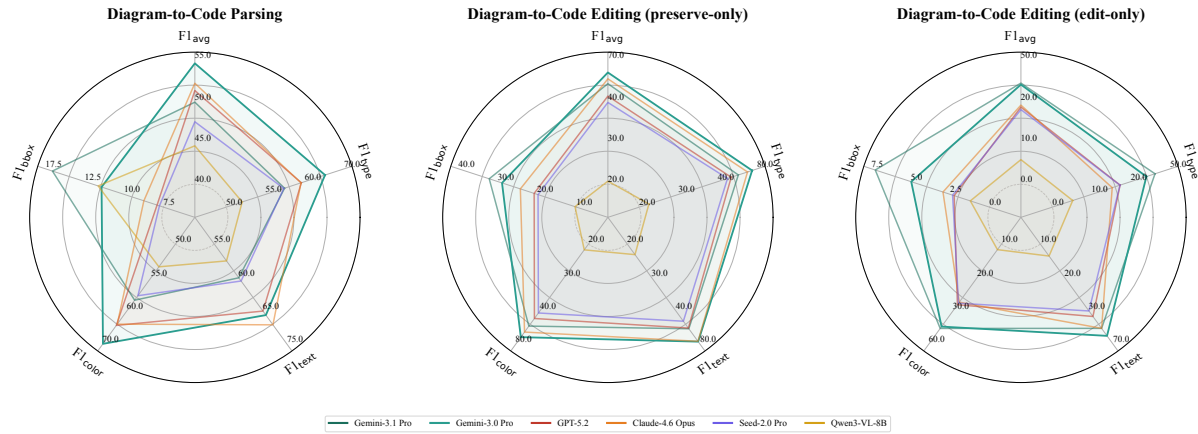


Figure 6 (Updated) Breakdown of F1 scores. Each panel shows five axes (avg, type, bbox, color, text), each with its own scale. $F1_{bbox}$ drops sharply in Diagram-to-Code Parsing, especially for GPT-5.2, Claude-4.6 Opus, and Seed-2.0 Pro.

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 [2, 3].

(4) Models struggle with fine-grained spatial grounding. All six models achieve relatively high performance on text extraction and color recognition (Fig. 6). However, spatial grounding (bbox) is consistently the weakest perception dimension. While models can recognize what elements exist, they struggle to encode where they are located. This deficit is sharpest for GPT-5.2 and Claude-4.6 Opus, where both score 62–71 on type, text, and color yet only 8.0 and 9.2 on bbox. Editing instructions in D2C-E compensate for this weakness only on preserved elements, not on newly edited ones: Claude-4.6 Opus’s $F1_{bbox}$ rises from 9.2 to 24.3 on preserve, and GPT-5.2 from 8.0 to 20.8; however, edit-only $F1_{bbox}$ drops to 3.0 and 2.3, respectively, since placing newly introduced elements is harder than retaining existing ones. Qwen3-VL-8B, however, falls behind all models across every perception dimension.

(5) 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).

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 domain knowledge reasoning.

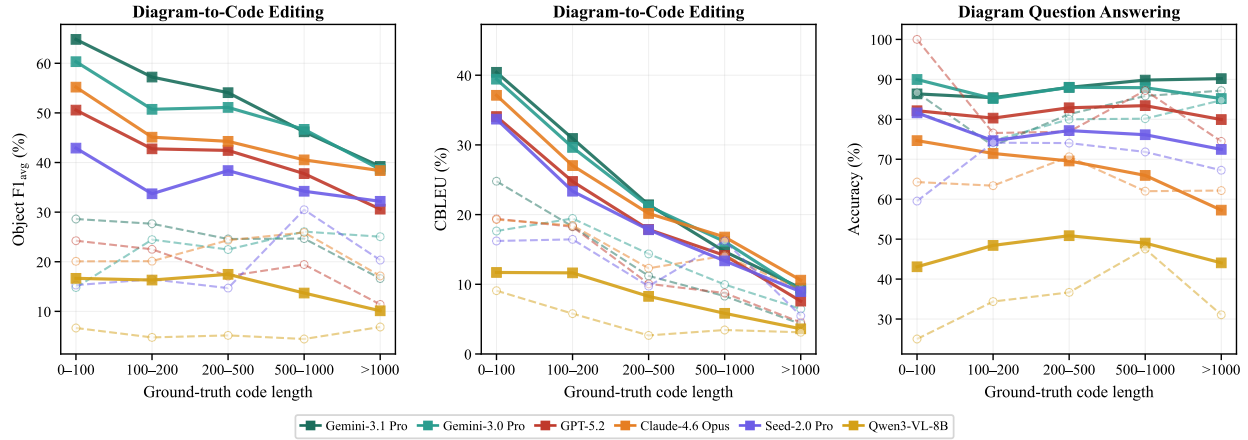


Figure 7 (Updated) Model performance on D2C-E/DQA across two diagram sets split by D2C-P output: successful rendering (solid lines) *vs.* failed rendering (dashed lines). D2C-E is evaluated by object $F1_{avg}$ (left) and CrystalBLEU (middle), and DQA by accuracy (right). The x -axis denotes ground-truth code length. Models that fail D2C-P also score lower on D2C-E and DQA.

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 ($\uparrow$ gain, $\downarrow$ 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	$\uparrow 0.2$	$\downarrow 0.2$	$\uparrow 1.2$	$\uparrow 1.4$	$\downarrow 1.2$	$\downarrow 1.2$	$\uparrow 2.6$	$\uparrow 1.5$	86.0	$\uparrow 1.0$	$\uparrow 0.8$	$\downarrow 1.8$	$\downarrow 8.8$
Claude-4.6 Opus	48.5	30.6	$\uparrow 1.2$	$\uparrow 0.3$	$\uparrow 2.2$	$\uparrow 1.9$	$\downarrow 0.2$	$\downarrow 0.9$	$\uparrow 5.1$	$\uparrow 2.8$	80.7	0.0	$\uparrow 3.8$	$\downarrow 2.0$	$\downarrow 0.3$
GPT-5.2	48.9	27.9	$\uparrow 2.9$	$\downarrow 2.9$	$\downarrow 0.3$	$\downarrow 0.5$	$\downarrow 1.5$	$\downarrow 0.8$	$\downarrow 1.6$	$\downarrow 4.1$	87.7	$\downarrow 7.3$	$\downarrow 1.3$	$\downarrow 4.3$	$\downarrow 5.7$
Gemini-3.0 Pro	50.9	30.8	$\downarrow 2.0$	$\downarrow 3.2$	$\downarrow 1.0$	$\downarrow 0.6$	$\downarrow 1.3$	$\downarrow 1.2$	$\uparrow 0.1$	$\downarrow 0.5$	88.3	$\uparrow 1.2$	$\downarrow 2.5$	$\downarrow 2.3$	$\downarrow 0.3$
Gemini-3.1 Pro	47.1	31.6	$\uparrow 1.9$	$\uparrow 1.4$	$\downarrow 4.3$	$\downarrow 3.0$	$\uparrow 0.3$	$\downarrow 0.8$	$\downarrow 5.0$	$\downarrow 4.5$	90.2	$\uparrow 0.5$	$\downarrow 3.2$	$\downarrow 2.7$	0.0
Qwen3-VL-8B	40.7	34.1	$\downarrow 6.1$	$\downarrow 5.6$	$\downarrow 0.1$	$\downarrow 1.2$	$\uparrow 1.4$	$\downarrow 0.2$	$\downarrow 5.6$	$\downarrow 6.4$	50.7	$\uparrow 3.0$	$\downarrow 1.3$	$\downarrow 2.3$	$\downarrow 3.3$

(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

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 ($\uparrow$ gain, $\downarrow$ 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	$\uparrow$ 9.3/ $\uparrow$ 7.1	$\uparrow$ 6.6/ $\uparrow$ 0.5	$\uparrow$ 0.7/ $\uparrow$ 0.2	$\uparrow$ 0.7/ $\uparrow$ 0.1	$\downarrow$ 2.3/ $\downarrow$ 0.4	$\downarrow$ 1.3/ $\uparrow$ 0.1	0.0/ $\downarrow$ 0.4	$\uparrow$ 0.1/ $\downarrow$ 0.3	$\uparrow$ 0.6/ $\uparrow$ 0.6	$\uparrow$ 0.1/0.0
Claude-4.6 Opus	55.9/28.3	39.3/1.3	$\uparrow$ 8.9/ $\uparrow$ 6.0	$\uparrow$ 6.0/0.0	$\uparrow$ 4.4/ $\uparrow$ 2.2	$\uparrow$ 4.7/ $\uparrow$ 0.5	$\uparrow$ 2.5/ $\uparrow$ 2.6	$\uparrow$ 2.5/0.0	$\downarrow$ 1.5/ $\downarrow$ 0.2	$\downarrow$ 0.4/ $\uparrow$ 0.1	$\uparrow$ 4.1/ $\uparrow$ 3.0	$\uparrow$ 5.6/ $\uparrow$ 0.7
GPT-5.2	54.9/29.3	38.6/1.2	$\uparrow$ 10.6/ $\uparrow$ 5.9	$\uparrow$ 7.2/ $\uparrow$ 0.1	$\uparrow$ 1.3/ $\uparrow$ 2.6	$\uparrow$ 2.4/0.0	$\uparrow$ 2.5/ $\uparrow$ 3.5	$\uparrow$ 2.7/ $\uparrow$ 0.1	$\uparrow$ 0.9/ $\uparrow$ 1.1	$\uparrow$ 1.6/ $\uparrow$ 0.3	$\uparrow$ 2.6/ $\uparrow$ 1.9	$\uparrow$ 2.9/ $\uparrow$ 0.4
Gemini-3.0 Pro	58.8/36.1	40.9/1.9	$\uparrow$ 4.1/ $\uparrow$ 3.3	$\uparrow$ 2.4/ $\uparrow$ 0.4	$\uparrow$ 2.2/ $\downarrow$ 0.1	$\downarrow$ 0.1/ $\uparrow$ 0.5	$\downarrow$ 1.1/ $\downarrow$ 0.6	$\downarrow$ 2.0/ $\uparrow$ 0.2	$\downarrow$ 0.4/ $\downarrow$ 0.5	$\downarrow$ 1.7/ $\uparrow$ 0.2	$\uparrow$ 2.8/ $\uparrow$ 0.8	$\uparrow$ 0.7/ $\uparrow$ 0.5
Gemini-3.1 Pro	57.6/38.1	38.0/2.5	$\uparrow$ 6.6/ $\uparrow$ 4.8	$\uparrow$ 4.6/ $\uparrow$ 0.6	$\downarrow$ 8.5/ $\downarrow$ 3.5	$\downarrow$ 5.3/ $\uparrow$ 0.8	$\downarrow$ 5.2/ $\downarrow$ 3.7	$\downarrow$ 2.9/ $\uparrow$ 0.2	$\uparrow$ 0.6/ $\uparrow$ 1.7	$\uparrow$ 0.5/ $\uparrow$ 0.7	$\downarrow$ 4.4/ $\downarrow$ 0.9	$\downarrow$ 2.9/ $\uparrow$ 1.4
Qwen3-VL-8B	47.0/22.2	34.2/0.8	$\uparrow$ 8.4/ $\uparrow$ 3.3	$\uparrow$ 4.3/ $\uparrow$ 0.2	$\downarrow$ 2.0/ $\downarrow$ 0.2	$\downarrow$ 2.2/ $\uparrow$ 0.2	$\downarrow$ 1.0/ $\uparrow$ 2.5	$\downarrow$ 1.4/ $\uparrow$ 0.4	$\downarrow$ 3.3/ $\downarrow$ 1.0	$\downarrow$ 3.0/0.0	0.0/ $\uparrow$ 1.7	$\downarrow$ 0.9/ $\uparrow$ 0.3

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.

5 Conclusion

Scientific diagrams are an important visual medium for expressing abstract ideas, and their TikZ code representation can be natively compiled inline within $\text{\LaTeX}$ 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

We thank all 13 annotators for their contributions to data quality assurance. Collectively, they cross-validated over 2,000 scientific diagrams across six domains. Three annotators are co-authors. The ten non-author contributors include one research scientist from NetMind.ai, eight PhD students from Nanjing University of Science and Technology (NJUST), and one PhD student from Westlake University. Kindly refer to § I for details of their contributions.

References

- [1] Minimax. Minimax m2.5: Built for real-world productivity. <https://www.minimax.io/news/minimax-m25>, 2026.
- [2] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. [arXiv preprint arXiv:2507.20534](https://arxiv.org/abs/2507.20534), 2025.
- [3] Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chengxing Xie, Cunxiang Wang, et al. Glm-5: from vibe coding to agentic engineering. [arXiv preprint arXiv:2602.15763](https://arxiv.org/abs/2602.15763), 2026.
- [4] Google DeepMind. Gemini 3.1 pro model card. <https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-1-Pro-Model-Card.pdf>, 2026.
- [5] Qwen Team. Qwen3.5: Towards native multimodal agents. <https://qwen.ai/blog?id=qwen3.5>, 2026.
- [6] OpenAI. Gpt-5.2. <https://developers.openai.com/api/docs/models/gpt-5.2>, 2025.
- [7] 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.
- [8] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, 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*, pages 9556–9567, 2024.
- [9] Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, 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)*, pages 15134–15186, 2025.
- [10] Yanpeng Sun, Jing Hao, Ke Zhu, Jiang-Jiang Liu, Yuxiang Zhao, Xiaofan Li, Gang Zhang, Zechao Li, and Jingdong Wang. Descriptive caption enhancement with visual specialists for multimodal perception. [arXiv preprint arXiv:2412.14233](https://arxiv.org/abs/2412.14233), 2024.
- [11] Weihao Bo, Shan Zhang, Yanpeng Sun, Jingjing Wu, Qunyi Xie, Xiao Tan, Kunbin Chen, Wei He, Xiaofan Li, Na Zhao, et al. Agentic learner with grow-and-refine multimodal semantic memory. [arXiv preprint arXiv:2511.21678](https://arxiv.org/abs/2511.21678), 2025.
- [12] OpenAI. Prism. <https://openai.com/prism/>, 2026.
- [13] Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, et al. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *Advances in Neural Information Processing Systems*, 37:113569–113697, 2024.
- [14] Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran XU, Xinyu Zhu, Siheng Li, Yuxiang Zhang, et al. Chartmimic: Evaluating lmm’s cross-modal reasoning capability via chart-to-code generation. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [15] Yanpeng Sun, Shan Zhang, Wei Tang, Aotian Chen, Piotr Koniusz, Kai Zou, Yuan Xue, and Anton van den Hengel. Math blind: Failures in diagram understanding undermine reasoning in mllms. [arXiv preprint arXiv:2503.20745](https://arxiv.org/abs/2503.20745), 2025.
- [16] Lingjie Jiang, Shaohan Huang, Xun Wu, Yixia Li, Dongdong Zhang, and Furu Wei. Viscodex: Unified multimodal code generation via merging vision and coding models. [arXiv preprint arXiv:2508.09945](https://arxiv.org/abs/2508.09945), 2025.
- [17] Chengyue Wu, Zhixuan Liang, Yixiao Ge, Qiushan Guo, Zeyu Lu, Jiahao Wang, Ying Shan, and Ping Luo. 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*, pages 3006–3028, 2025.
- [18] Juan A Rodriguez, Abhay Puri, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images and text. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16175–16186, 2025.

- [19] Jingxuan Wei, Cheng Tan, Qi Chen, Gaowei Wu, Siyuan Li, Zhangyang Gao, Linzhuang Sun, Bihui Yu, and Ruifeng Guo. From words to structured visuals: A benchmark and framework for text-to-diagram generation and editing. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 13315–13325, 2025.
- [20] Jonas Belouadi, Anne Lauscher, and Steffen Eger. Automatizk: Text-guided synthesis of scientific vector graphics with tikz. In The Twelfth International Conference on Learning Representations.
- [21] Jonas Belouadi, Simone Ponzetto, and Steffen Eger. Detikzify: Synthesizing graphics programs for scientific figures and sketches with tikz. Advances in Neural Information Processing Systems, 37:85074–85108, 2024.
- [22] Josselin S Roberts, Tony Lee, Chi H Wong, Michihiro Yasunaga, Yifan Mai, and Percy Liang. Image2struct: Benchmarking structure extraction for vision-language models. Advances in Neural Information Processing Systems, 37:115058–115097, 2024.
- [23] Shuo Li, Jiajun Sun, Zhekai Wang, Xiaoran Fan, Hui Li, Dingwen Yang, Zhiheng Xi, Yijun Wang, Zifei Shan, Tao Gui, et al. ChartE³: A comprehensive benchmark for end-to-end chart editing. arXiv preprint arXiv:2601.21694, 2026.
- [24] Donglu Yang, Liang Zhang, Zihao Yue, Liangyu Chen, Yichen Xu, Wenxuan Wang, and Qin Jin. Chartm3: Benchmarking chart editing with multimodal instructions. In Proceedings of the 33rd ACM International Conference on Multimedia, pages 5001–5009, 2025.
- [25] Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In Findings of the association for computational linguistics: ACL 2022, pages 2263–2279, 2022.
- [26] Zeju Qiu, Weiyang Liu, Haiwen Feng, Zhen Liu, Tim Z Xiao, Katherine M Collins, Joshua B Tenenbaum, Adrian Weller, Michael J Black, and Bernhard Schölkopf. Can large language models understand symbolic graphics programs? In The Thirteenth International Conference on Learning Representations, 2025.
- [27] Anthropic. Model context protocol. <https://modelcontextprotocol.io>, 2024.
- [28] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In European conference on computer vision, pages 216–233. Springer, 2024.
- [29] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, 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, pages 24108–24118, 2025.
- [30] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In The Twelfth International Conference on Learning Representations.
- [31] Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In European Conference on Computer Vision, pages 169–186. Springer, 2024.
- [32] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4):600–612, 2004.
- [33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In International conference on machine learning, pages 8748–8763. PmLR, 2021.
- [34] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 586–595, 2018.
- [35] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems, 30, 2017.

- [36] Aryaz Eghbali and Michael Pradel. Crystalbleu: precisely and efficiently measuring the similarity of code. In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, pages 1–12, 2022.
- [37] Runjie Zhou, Youbo Shao, Haoyu Lu, Bowei Xing, Tongtong Bai, Yujie Chen, Jie Zhao, Lin Sui, Haotian Yao, Zijia Zhao, et al. Worldvqa: Measuring atomic world knowledge in multimodal large language models. arXiv preprint arXiv:2602.02537, 2026.
- [38] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In The eleventh international conference on learning representations, 2022.
- [39] Rajkumar Buyya et al. Agentic artificial intelligence (ai): Architectures, taxonomies, and evaluation of large language model agents. arXiv preprint arXiv:2601.12560, 2026.
- [40] Thomas Hartung. Ai, agentic models and lab automation for scientific discovery—the beginning of science. Frontiers in Artificial Intelligence, 8:1649155, 2025.
- [41] Shixiang Tang, Yizhou Wang, Lu Chen, Yuan Wang, Sida Peng, Dan Xu, and Wanli Ouyang. Human-centric foundation models: Perception, generation and agentic modeling. arXiv preprint arXiv:2502.08556, 2025.
- [42] Liang Chen, Weichu Xie, Yiyang Liang, Hongfeng He, Hans Zhao, Zhibo Yang, Zhiqi Huang, Haoning Wu, Haoyu Lu, Yiping Bao, et al. Babyvision: Visual reasoning beyond language. arXiv preprint arXiv:2601.06521, 2026.
- [43] Google DeepMind. Gemini 3 pro model card. <https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf>, 2025.
- [44] Google DeepMind. Gemini 3 flash model card. <https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf>, 2025.
- [45] Anthropic. Claude opus 4.6. <https://www.anthropic.com/claude/opus>, 2026.
- [46] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. arXiv preprint arXiv:2511.21631, 2025.
- [47] Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2.5: Visual agentic intelligence. arXiv preprint arXiv:2602.02276, 2026.
- [48] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. arXiv preprint arXiv:2504.10479, 2025.
- [49] Jonas Belouadi, Eddy Ilg, Margret Keuper, Hideki Tanaka, Masao Utiyama, Raj Dabre, Steffen Eger, and Simone Ponzetto. Tikzero: Zero-shot text-guided graphics program synthesis. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 17793–17806, 2025.
- [50] Haruto Yoshida, Keito Kudo, Yoichi Aoki, Ryota Tanaka, Itsumi Saito, Keisuke Sakaguchi, and Kentaro Inui. Nodes are early, edges are late: Probing diagram representations in large vision-language models. arXiv preprint arXiv:2603.02865, 2026.
- [51] Xue Li, Yiyao Sun, Wei Cheng, Yinglun Zhu, and Haifeng Chen. Chain-of-region: Visual language models need details for diagram analysis. In The Thirteenth International Conference on Learning Representations, 2025.
- [52] Mojdeh Rahmadian, Ashkan Sami, and Yanchao Yu. Challenges and feasibility of multimodal llms in er diagram evaluation. Cogent Education, 12(1):2590901, 2025.
- [53] Sakthivel Thangaraj, Neelesh Kumar Shukla, and Viji Krishnamurthy. Ontology-driven multimodal framework for automated interpretation and description of architecture diagrams. In 2025 IEEE International Conference on Big Data (BigData), pages 2493–2502. IEEE, 2025.
- [54] Shue Shiinoki, Ryo Koshihara, Hayato Motegi, and Masumi Morishige. Overcoming vision language model challenges in diagram understanding: A proof-of-concept with xml-driven large language models solutions. arXiv preprint arXiv:2502.04389, 2025.
- [55] Bowen Yu and Cláudio T. Silva. Flowsense: A natural language interface for visual data exploration within a dataflow system. IEEE Transactions on Visualization and Computer Graphics, 26(1):1–11, 2020.

- [56] Qian Wang, Aleksandar Cvejić, Abdelrahman Eldesokey, and Peter Wonka. Editclip: Representation learning for image editing. pages 15960–15970, 2025.
- [57] Cheng Tan, Qi Chen, Jingxuan Wei, Gaowei Wu, Zhangyang Gao, Siyuan Li, Bihui Yu, Ruifeng Guo, and Stan Z Li. Sketchagent: Generating structured diagrams from hand-drawn sketches. 2025.
- [58] ZENG Xingchen, Zhewei Su, Hengming Zhang, Juyong Jiang, Jiazhi Xia, and Wei Zeng. Davinci: Reinforcing visual-structural syntax in mllms for generalized scientific diagram parsing. In The Fourteenth International Conference on Learning Representations, 2026.
- [59] Qingyang Mao, Qi Cai, Yehao Li, Yingwei Pan, Mingyue Cheng, Ting Yao, Qi Liu, and Tao Mei. Visual autoregressive modeling for instruction-guided image editing. arXiv preprint arXiv:2508.15772, 2025.
- [60] Christian Meske, Tobias Hermanns, Esther Von der Weiden, Kai-Uwe Loser, and Thorsten Berger. Vibe coding as a reconfiguration of intent mediation in software development: Definition, implications, and research agenda. IEEE Access, 13:213242–213259, 2025.
- [61] Christian Greisinger and Steffen Eger. Tikzilla: Scaling text-to-tikz with high-quality data and reinforcement learning. arXiv preprint arXiv:2603.03072, 2026.
- [62] Ke Wang, Junting Pan, Linda Wei, Aojun Zhou, Weikang Shi, Zimu Lu, Han Xiao, Yunqiao Yang, Houxing Ren, Mingjie Zhan, et al. Mathcoder-vl: Bridging vision and code for enhanced multimodal mathematical reasoning. In Findings of the Association for Computational Linguistics: ACL 2025, pages 2505–2534, 2025.
- [63] Sher Badshah, Moamen Moustafa, and Hassan Sajjad. Clev: Llm-based evaluation through lightweight efficient voting for free-form question-answering. In Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 1513–1531, 2025.

Diagram-MMU: A Multi-Modal Benchmark for Scientific Diagrams (Appendix)

Contents

A	Related Works	20
A.1	Diagram Parsing	20
A.2	Diagram Editing	20
A.3	Vibe Writing	20
B	Benchmark Details	20
B.1	Data Statistics	20
B.2	Task Generation Pipeline	22
B.3	D2C-P Examples	23
B.4	D2C-E Templates	26
B.5	D2C-E Examples	27
B.6	DQA Templates	31
B.7	DQA Descriptive Question Examples	36
B.8	DQA Reasoning Question Examples	38
C	Evaluation Methodology	40
C.1	Semantic Object Model (SOM) Pipeline	40
C.2	Evaluation Metrics	41
C.3	DQA LLM-as-a-Judge Details	43
C.4	LLM Judge Human Agreement Validation	44
D	Implementation Details	45
D.1	Model Configurations	45
D.2	Inference and Compilation	45
E	Evaluation Prompts	46
E.1	Foundational Evaluation Prompts	46
E.2	Agentic Evaluation Prompts	46
F	MCP-Based TikZ Documentation Server	49
F.1	Server Architecture and Implementation	49
F.2	Indexed Documentation List	50

G	Additional Experimental Results	51
G.1	Foundational Results Per Diagram Type	51
G.2	F1 Scores Per Object	51
G.3	DQA Analysis Per Diagram Type	52
G.4	Agentic Tool-Call Statistics	52
H	Limitations and Future Work	54
I	Acknowledgement	55

A Related Works

A.1 Diagram Parsing

Diagram parsing extracts the topology of a diagram, e.g., the basic elements and their relationships. This requires symbolic perception instead of pixel-level understanding. A geometric primitive carries meaning as a symbol: its semantics come from its type, path and spatial relationships, not from its pixel values. For example, a circle remains a circle regardless of its color, size or line thickness. This contrasts with natural images, where each pixel contributes semantic content through color, texture, and intensity. This distinction explains why MLLMs that perform well on natural image tasks struggle with diagrams [15]. A common surrogate is Optical Character Recognition (OCR), where models extract text annotations to interpret diagrams. However, OCR captures only textual content while ignoring geometric topology, thus failing at structural relationship extraction [50, 51]. Consequently, error rates remain high for domain-specific diagrams like scientific charts and Entity-Relationship (ER) schemas [52]. To address this, recent work integrates visual parsing with domain ontologies [53] or uses Region Decomposition [54]. These methods capture cross-node dependencies, transforming pixel-level features into high-level semantics. An alternative line of work represents diagrams as code, called diagram-to-code parsing, which is the focus of this paper. We review these methods in §2&Tabel 1.

A.2 Diagram Editing

Diagram editing is shifting from manual graphical manipulation to intent-driven editing using natural language [19, 55]. Unlike general image editing via diffusion models, editing technical diagrams requires strict global logical coherence. Models must modify local attributes without causing structural damage [56]. Recent frameworks solve this by parsing instructions and modifying vector graphics or code directly [57]. For example, DiagramAgent uses a multi-agent system. It decomposes user instructions into explicit plans, allowing specialized agents to collaboratively edit and verify $\text{\LaTeX}$ /TikZ or DOT representations [58]. Other methods combine visual representation learning with autoregressive frameworks, such as VAREdit [59]. These approaches bridge the gap between language intents and vector graphics, safely handling complex topological changes.

A.3 Vibe Writing

With vibe coding [60], researchers provide a diagram and describe the desired changes in natural language, and an MLLM generates the compilable code; a workflow that fundamentally depends on strong diagram-to-code generation. In the vibe writing environment, human creators no longer need to write tedious macros or calculate coordinates. By providing simple logical intents or vibes, users guide multimodal agents to autonomously handle structural inference, code synthesis, and visual rendering. This radically reshapes how researchers create academic literature and diagrams.

Generating highly precise TikZ/ $\text{\LaTeX}$ code is a major research focus [20]. However, high-quality diagram-code pairs are scarce. To address this, models like TikZero use zero-shot architectures to decouple visual and textual features [49]. Similarly, DeTikZify optimizes generated TikZ code using Monte Carlo Tree Search (MCTS) and compiler feedback [21]. Furthermore, TikZilla improves visual fidelity using Reinforcement Learning (RL) and inverse graphics rewards [61], while MathCoder-VL synthesizes massive aligned datasets using a “model-in-the-loop” strategy [62].

B Benchmark Details

B.1 Data Statistics

Diagram-MMU covers six scientific diagram types, each associated with one or more dedicated TikZ packages. The full dataset contains 3,744 diagrams paired with 18,305 evaluation instances (1 D2C-P + 2 D2C-E + 2 DQA per diagram), with a balanced **mini split** of 300 diagrams (50 per domain). Table B.1 summarizes the per-domain source distribution and task instance counts. We describe each type of diagram, as follows:

(1) Diagram types:

Table B.1 Per-domain statistics of the Diagram-MMU. “Official” denotes diagrams sourced from TikZ package documentation; “Community” denotes diagrams from community resources. Each diagram is paired with up to 5 evaluation questions (1 D2C-P + 2 D2C-E + 2 DQA); a small number of instances were removed during human cross-validation, so per-task totals are slightly below the nominal $1 \times / 2 \times / 2 \times$ diagram count.

Domain	#Diag.	Primary TikZ Pkg.	Data Source		Task Instances			Total	Code Stats	
			Official	Community	D2C-P	D2C-E	DQA		Avg. Lines	Avg. Chars
Charts	960	pgfplots	579	381	959	1,912	1,834	4,705	34.6	1,098
Planar Geom.	601	tikz/tkz-euclide	253	348	598	1,162	1,118	2,878	26.9	1,074
3D Shapes	237	pgfplots	146	91	236	471	455	1,162	29.6	988
Graph Struct.	1,356	tikz	299	1,057	1,356	2,699	2,679	6,734	34.9	1,452
Chemistry	187	chemfig	185	2	187	373	366	926	18.5	412
Circuit	403	circuitikz	387	16	403	803	694	1,900	21.2	660
Total	3,744	–	1,849	1,895	3,739	7,420	7,146	18,305	30.9	1,134

Table B.2 D2C-E editing dimension breakdown for the full dataset and mini split. 3D Shapes and Chemistry have no layout-dimension edits (marked “–”).

Domain	Full Dataset					Mini Split				
	Text	Color	Scope	Layout	Total	Text	Color	Scope	Layout	Total
Charts	438	194	365	915	1,912	21	10	19	50	100
Planar Geom.	172	205	221	564	1,162	20	17	14	49	100
3D Shapes	81	123	267	–	471	14	23	63	–	100
Graph Struct.	315	319	735	1,330	2,699	10	12	28	50	100
Chemistry	57	139	177	–	373	9	41	50	–	100
Circuit	122	71	215	395	803	18	7	26	49	100
Total	1,185	1,051	1,980	3,204	7,420	92	110	200	198	600

- **Charts** (960 diagrams). Statistical and data visualizations rendered primarily with `pgfplots`, including line plots, bar charts, scatter plots, pie charts, and histograms. Domain knowledge includes statistics, trend analysis, and extremum identification.
- **Planar Geometry** (601 diagrams). Two-dimensional geometric constructions drawn with `tikz` and `tkz-euclide`, including triangles, circles, polygons, and angle/distance annotations. Domain knowledge covers Euclidean notation, area/perimeter formulas, and geometric theorems.
- **3D Shapes** (237 diagrams). Three-dimensional solid geometry visualizations rendered with `pgfplots` and `tikz`, including prisms, pyramids, spheres, and cross-section illustrations. Domain knowledge includes solid geometry, surface area, and volume computation.
- **Graph Structures** (1,356 diagrams). Combinatorial and network-theoretic diagrams drawn with `tikz` and `tikz-network`, including directed/undirected graphs, trees, automata, and flow networks. Domain knowledge includes degree, connectivity, shortest paths, and tree properties.
- **Chemistry** (187 diagrams). Molecular and chemical structure diagrams rendered with `chemfig`, including organic molecules, functional groups, and reaction schemes. Domain knowledge covers bond types, atom labels, and functional group identification.
- **Circuit Diagrams** (403 diagrams). Electrical and electronic circuit schematics drawn with `circuitikz`, including resistive networks, RC/RLC circuits, and logic gates. Domain knowledge includes circuit topology, Ohm’s law, and series/parallel analysis.

(2) D2C-E editing dimensions. Each D2C-E sample is annotated with one of four editing dimensions: `text` (label/annotation modification), `color` (fill/stroke color change), `scope` (local element addition, deletion, or transformation), and `layout` (global structural change such as chart type conversion or circuit topology modification). We exclude layout editing from 3D shapes and chemistry. Table B.2 shows the per-domain breakdown.

(3) DQA question types. DQA comprises two question types: descriptive and reasoning. Reasoning questions are further split into standard and `what-if` types (definitions in §3.3). Table B.3 shows the per-domain breakdown.

(4) DQA answer formats. Each DQA question specifies one of three output instruction types for answering:

Table B.3 DQA question type breakdown for the full dataset and mini split. Standard questions cover both descriptive and reasoning types; what-if questions are reasoning-only and require hypothetical inference conditioned on element modifications.

Domain	Full Dataset				Mini Split			
	Standard	What-if	Total	% What-if	Standard	What-if	Total	% What-if
Charts	1,244	590	1,834	32.2	78	22	100	22.0
Planar Geom.	780	338	1,118	30.2	65	35	100	35.0
3D Shapes	344	111	455	24.4	75	25	100	25.0
Graph Struct.	1,787	892	2,679	33.3	72	28	100	28.0
Chemistry	250	116	366	31.7	74	26	100	26.0
Circuit	480	214	694	30.8	70	30	100	30.0
Total	4,885	2,261	7,146	31.6	434	166	600	27.7

OI-NUM (numerical answer, 5,013 questions), OI-TERM (domain-specific term or label, 1,779 questions), and OI-LIST (list of elements, 354 questions). This distribution reflects that the majority of questions require numerical computation (e.g., calculating area, degree, or resistance values), followed by identification of domain-specific entities.

(5) Mini split. The mini split is a class-balanced subset of the full dataset, containing exactly 50 diagrams per domain (300 total) with 1,500 evaluation instances (300 D2C-P + 600 D2C-E + 600 DQA). The per-domain task breakdowns of the mini split are shown in Tables B.2 and B.3.

B.2 Task Generation Pipeline

Figure 2 illustrates the overall benchmark construction pipeline. Herein, we detail the task-specific generation procedure:

(1) Diagram-to-Code Parsing (D2C-P). D2C-P uses fixed task templates instantiated directly from the original diagrams. The task question asks the model to convert the diagram image into compilable TikZ code, conditioned on a provided preamble (document class, required packages, and libraries). The original TikZ source code serves as the ground-truth answer. No agentic generation is required for this task.

(2) Diagram-to-Code Editing (D2C-E). D2C-E employs an agentic pipeline with two main components: a generation agent (Gemini-3 Flash) and a verifier agent comprising two judge models (GPT-5.2 and Gemini-3 Pro) for cross-validation. The pipeline proceeds as follows:

- **Template predefinition.** We manually design a pool of editing task templates across four dimensions (text, color, scope, layout) for each of the six domains (Table 3). Each template contains placeholders for diagram-specific elements (e.g., labels, colors, vertices, and edges).
- **Task selection.** Given a diagram and its TikZ source code, the generation agent selects two editing tasks from the template pool of the corresponding domain, constrained to cover distinct editing dimensions to ensure diversity.
- **Answer generation.** The generation agent instantiates the selected templates with specific diagram elements and generates the corresponding modified TikZ code. Multi-engine compilation (`pdflatex`→`lualatex`→`xelatex`) is performed to verify executability.
- **Agent verification.** The verifier agents evaluate generated results against predefined criteria: all edits must be visually observable and conform to the selected editing dimension, and unaffected elements must remain unchanged. A case is accepted only when both verifier agents agree.
- **Iterative refinement.** Failed cases, together with verifier feedback, are returned to the generation agent for regeneration, with a maximum of three attempts.
- **Human verification.** All accepted instances are manually reviewed by 13 graduate students. Each annotator’s assigned instances are cross-validated by another annotator to ensure language clarity, logical consistency, and answer correctness.

(3) Diagram Question Answering (DQA). The DQA generation pipeline follows the same agentic workflow as D2C-E; only the task templates differ. The generation agent produces descriptive and reasoning questions

based on the diagram, its TikZ source code, and domain-specific templates encoding symbol semantics and domain knowledge (§B.6). Template placeholders are instantiated using diagram elements extracted from the source code. For example, in organic chemistry, functional groups (e.g., COOH or OH) are extracted to generate questions such as “which functional group is present in this molecule?”. The verifier agents then assess each (diagram, question, answer) triple for logical consistency with both visual evidence and domain knowledge; regeneration with verifier feedback is allowed for up to three attempts, and only validated QA pairs are retained. Human verification follows the same cross-validation protocol as D2C-E.

Each diagram yields five evaluation instances (1 D2C-P + 2 D2C-E + 2 DQA). The template-guided selection enforces dimensional diversity: D2C-E instances cover distinct editing dimensions per diagram, and DQA questions span both descriptive and reasoning types. All templates and rubric designs are detailed in §B.4 and §B.6.

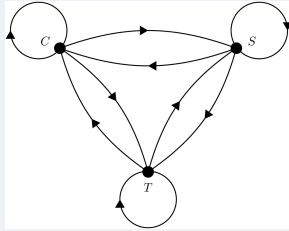
B.3 D2C-P Examples

Figure B.1 presents representative D2C-P case studies across diagram domains. Each case shows the input diagram, the task prompt, the ground-truth TikZ code, and the rendered output.



Graph Structures Case 1

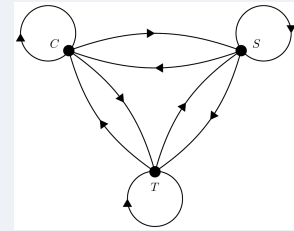
Q: Recreate this diagram using $\LaTeX$. Provide the full, executable code.



A:

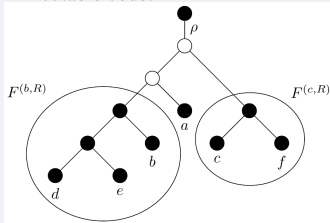
```
\begin{tikzpicture}[>=triangle 60]
\draw[thick,fill=black] (0,0) circle (1.5mm);
\draw[thick,fill=black] (5,0) circle (1.5mm);
\draw[thick,fill=black] (2.5,-3.5) circle (1.5mm);
\node [left] at (-0.15, 0.2) {\$C\$};
\node [right] at (5.2,0.2) {\$S\$};
\node [below] at (2.5,-3.7) {\$T\$};
...
\draw[>=,5,thick](2.5,-3.5) to [bend left=20](0,0);
\draw[>=,5,thick](0,0) to [bend left=20](2.5,-3.5);
\draw[yscale=-1,yshift=8.55cm,>=,5,thick](2.5,-4.27) circle (0.8cm);
\draw[>=,5,thick](2.5,-3.5) to [bend left=20](5,0);
\draw[>=,5,thick](5,0) to [bend left=20](2.5,-3.5);
\end{tikzpicture}
```

Rendered Image:



Graph Structures Case 2

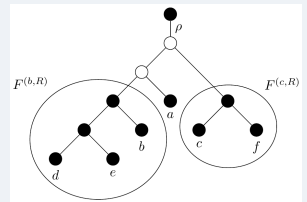
Q: Recreate this diagram using $\LaTeX$. Provide the full, executable code.



A:

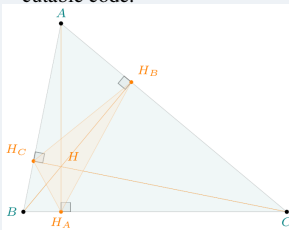
```
% [5 packages: fontenc, inputenc, lmodern...]
\begin{tikzpicture}
[vertex/.style={minimum size=2pt,fill,draw,circle},
open/.style={fill=none},
sibling distance=1.5cm,level distance=75cm,
every fit/.style={ellipse,draw,inner sep=-2pt},
leaf/.style={label={\name=#1\$}} below: {\$#1\$}} ,auto]
\node [vertex] (root) {}
...
child { node [vertex,leaf=c] {} }
child { node [vertex,leaf=f] {} } } }
edge from parent node {\$vho\$} ;
\node [fit=(d) (e) (b) (b's parent),label=above left:\$F^{(b,R)}\$] {};
\node [fit=(c) (f) (f's parent),label=above right:\$F^{(c,R)}\$] {};
\end{tikzpicture}
```

Rendered Image:



Planar Geometry Case 1

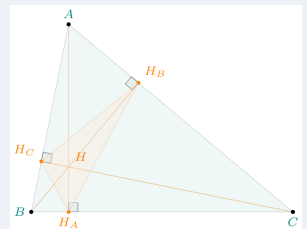
Q: Recreate this diagram using $\LaTeX$. Provide the full, executable code.



A:

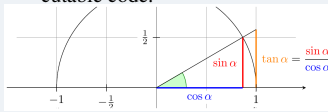
```
% [3 packages: amsmath,amssymb, xcolor, tkz-euclide]
\begin{tikzpicture}[scale=.75]
\tkzDefPoints{ 1/5/A,0/0/B,7/0/C}
\tkzDefSpecTriangle[orthic](A,B,C){H_A,H_B,H_C}
\tkzDefTriangleCenter[ortho](B,C,A)
\tkzGetPoint{H}
\tkzDrawSegments[new](A,H_A B,H_B C,H_C)
\tkzMarkRightAngles[fill=gray!20,opacity=.5](A,H_A C B,H_B A C,H_C,A)
...
\tkzLabelPoints[left](B)
\tkzLabelPoints[above](A)
\tkzLabelPoints[new](H_A)
\tkzLabelPoints[new,above left](H_C)
\tkzLabelPoints[new,above right](H_B,H)
\end{tikzpicture}
```

Rendered Image:



Planar Geometry Case 2

Q: Recreate this diagram using $\LaTeX$. Provide the full, executable code.



A:

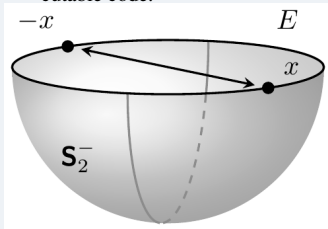
```
% [1 packages: tikz]
\begin{tikzpicture}[scale=3]
\clip(-2,-0.2)rectangle(2,0.8);
\draw[step=.5cm,gray,very thin](-1.4,-1.4)grid(1.4,1.4);
\filldraw[fill=green!20,draw=green!50!black](0,0)--(3mm,0mm)arc[start angle=0,
end angle=30,radius=3mm]--cycle;
\draw[>->](-1.5,0)--(1.5,0)coordinate(x axis);
\draw[>->](0,-1.5)--(0,1.5)coordinate(y axis);
\draw(0,0)circle[radius=1cm];
...
\path[name path=sloped line](0,0)--(30:1.5cm);
\draw[name intersections={of=upward line and sloped line,by=t}][very thick,orange]
(1,0)--node[right=1pt,fill=white]{\displaystyle\tan\alpha=\frac{\sin\alpha}{\cos\alpha}}
\color{blue}\cos\alpha$(t);
\draw(0,0)--(t);
\foreach\x\text in{-1,-0.5/-\frac{1}{2},1}\draw(\x cm,1pt)--(\x cm,-1pt)node[
anchor=north,fill=white]{\text{x}};
\foreach\y\text in{-1,-0.5/-\frac{1}{2},0.5/\frac{1}{2},1}\draw(1pt,\y cm)--(-1
pt,\y cm)node[anchor=east,fill=white]{\text{y}};
\end{tikzpicture}
```

Rendered Image:



3D Shapes Case 1

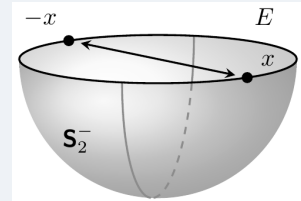
Q: Recreate this diagram using $\LaTeX$. Provide the full, executable code.



A:

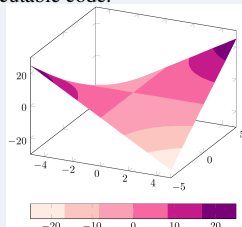
```
% [2 packages: tikz,bm,pgfplots, tikz-3dplot]
\tdplotsetmaincoords{80}{140}
\begin{tikzpicture}[scale=2,tdplot_main_coords]
\tikzset{dot/.style={circle,fill,minimum size=4.5pt,inner sep=0pt,outer sep=0pt}}
\hemispherebehind/.style={ball color=gray!20!white,opacity=0.3},hemispherefront/.
style={ball color=gray!65!white,opacity=0.3},circlearc/.style={thick,gray
!90},circlearchidden/.style={thick,dashed,gray!90},equator/.style={thick,
black},diameter/.style={thick,black,stealth-stealth,shorten <=5pt,shorten
>=5pt}}
\newcommand{\hemispherefront}{(1cm,0)arc(0:-180:1cm and 1.8
mm)arc(180:0:1cm and -1cm)}
\newcommand{\hemispherebehind}{(1cm,0)arc(0:-180:1cm and -1.8mm)arc
(-180:0:1cm and 1cm)}
\newcommand{\equator}{(-1,0,0)arc(0:360:-1)}
\tdplotsetthetaplanecoords{35}\path[tdplot_rotated_coords,name path=semicircle]
(0,-1,0)arc(90:-90:-1);
...
\draw[circlearchidden,intersection segments={of=semicircle and hemisphere,
sequence={L2}}];
\draw[circlearc,intersection segments={of=semicircle and hemisphere,sequence={L
3}}];
\draw[equator]\equator node[pos=1,label={90:$E$}]{};
\draw[diameter](0,-1,0)node[dot,label={120:$-x$}]{}--(0,1,0)node[dot,label
={30:$x$}]{};
\node at(75,0,-.5){\bm{\mathsf{S}}_2^-};
\end{tikzpicture}
```

Rendered Image:



3D Shapes Case 2

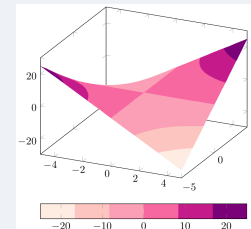
Q: Recreate this diagram using $\LaTeX$. Provide the full, executable code.



A:

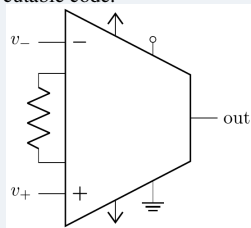
```
% [1 packages: pgfplots]
\begin{tikzpicture}
\begin{axis}[
colormap access=piecewise constant,
colormap/RdPu-6,
colorbar horizontal,
]
\addplot3 [
surf,
shader=interp,
] {x*y};
\end{axis}
\end{tikzpicture}
```

Rendered Image:



Circuit Diagrams Case 1

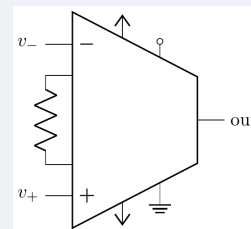
Q: Recreate this diagram using $\LaTeX$. Provide the full, executable code.



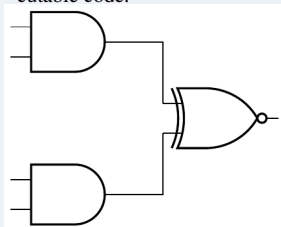
A:

```
% [3 packages: fontenc, inputenc, circuitikz]
\begin{circuitikz}
\draw
(0,0) node[inst amp ra] (opamp) {}
(opamp.+) node[left] {$v_+$}
(opamp.-) node[left] {$v_-$}
(opamp.out) node[right] {out}
(opamp.up) node[vcc] {}
(opamp.down) node[vee] {}
(opamp.refv down) node[ground] {}
(opamp.refv up) to[short, -o] ++(0,0.3)
(opamp.ra-) to[R] (opamp.ra+);
\end{circuitikz}
```

Rendered Image:

**Circuit Diagrams Case 2**

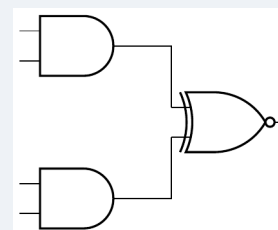
Q: Recreate this diagram using $\LaTeX$. Provide the full, executable code.



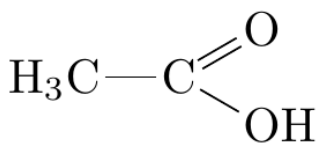
A:

```
% [4 packages: fontenc, inputenc, lmodern...]
\begin{circuitikz}
\draw
(0,2) node[and port] (myand1) {}
(0,0) node[and port] (myand2) {}
(2,1) node[xnor port] (myxnor) {}
(myand1.out) -| (myxnor.in 1)
(myand2.out) -| (myxnor.in 2);
\end{circuitikz}
```

Rendered Image:

**Chemistry Case 1**

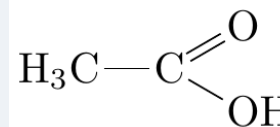
Q: Recreate this diagram using $\LaTeX$. Provide the full, executable code.



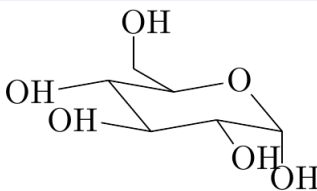
A:

```
% [1 packages: chemfig]
\chemfig{H_3C-C(=[:30]O)-[:30]OH}
\flipnext
\chemfig{H_3C-C(=[:30]O)-[:30]OH}\medskip
\chemfig{H_3C-C(=[:30]O)-[:30]OH}
\flipnext
\chemfig{H_3C-C(=[:30]O)-[:30]OH}
```

Rendered Image:

**Chemistry Case 2**

Q: Recreate this diagram using $\LaTeX$. Provide the full, executable code.



A:

```
% [3 packages: amsmath, amssymb, xcolor, chemfig]
\chemfig{?(-[:190]OH)-[:50](-[:170]OH)-[:10](-[:55,0.7]OH)
-[:10](-[:6,0.7]OH)-[:130]O-[:190]?(-[:150,0.7]-[:2,0.7]OH)}
```

Rendered Image:

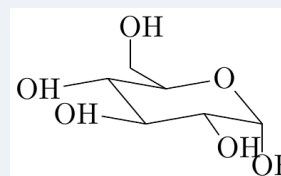


Figure B.1 Representative D2C-P (Diagram-to-Code Parsing) examples. Given an input diagram, the task requires generating complete, compilable $\LaTeX$ /TikZ code that faithfully reproduces the original.

B.4 D2C-E Templates

The D2C-E employs a pool of **17 editing task templates** across four evaluation dimensions: *color* (C, 2 templates), *text* (T, 4 templates), *scope* (S, 8 templates), and *layout* (L, 3 templates). Each template specifies a natural-language editing instruction with placeholders (in square brackets) that are instantiated with diagram-specific

Table B.4 Editing template pool of D2C-E. 17 templates span four evaluation dimensions. Placeholders in brackets are instantiated with diagram-specific elements during generation.

ID	Task Name	Template
Color		
C1	Element Color Change	Change the <code>[color_attr]</code> of <code>[target_element]</code> to <code>[new_color]</code> .
C2	Text Color Change	Change the text color of <code>[target_text]</code> to <code>[new_color]</code> .
Text		
T1	Label/Annotation Rename	Rename the label <code>[old_text]</code> to <code>[new_text]</code> .
T2	Data Value Modification	Change the data value at <code>[position]</code> from <code>[old_value]</code> to <code>[new_value]</code> .
T3	Component Parameter Mod.	Change the parameter of <code>[component]</code> from <code>[old_param]</code> to <code>[new_param]</code> .
T4	Title/Legend Text Mod.	Change the <code>[text_type]</code> text from <code>[old_text]</code> to <code>[new_text]</code> .
Scope		
S1	Element Position Move	Move <code>[target_element]</code> to <code>[new_position]</code> .
S2	Rotation/Mirror Transform	Rotate <code>[target]</code> by <code>[degrees]°</code> <code>[direction]</code> . / Mirror <code>[target]</code> along the <code>[axis]</code> axis.
S3	Axis Range Scaling	Change the <code>[axis]</code> axis range from <code>[old_range]</code> to <code>[new_range]</code> .
S4	Element Position Swap	Swap the positions of <code>[element_A]</code> and <code>[element_B]</code> .
S5	Element Deletion	Remove <code>[target_element]</code> from the diagram.
S6	Element Addition	Add <code>[new_element]</code> to the diagram.
S7	Node Shape Replacement	Change the shape of <code>[target_node]</code> from <code>[old_shape]</code> to <code>[new_shape]</code> .
S8	Component Type Replacement	Replace <code>[old_component]</code> with <code>[new_component_type]</code> .
Layout		
L1	Structure Type Conversion	Convert <code>[current_type]</code> to <code>[target_type]</code> .
L2	Conditional Filtering	Filter the data to only show <code>[condition]</code> .
L3	Reference Element Addition	Add <code>[reference_element]</code> to the diagram.

elements during the agentic pipeline (§B.2). Table B.4 presents the complete template pool, and Table B.5 shows per-domain applicability.

B.5 D2C-E Examples

Figure B.2 presents representative D2C-E cases. Each case shows the original diagram, an editing instruction, the modified TikZ code, and the rendered output after editing.

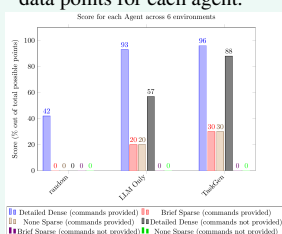
Table B.5 Editing dimensions for each type of diagram. ✓ indicates the dimension applies to that domain.

Task	Charts	P. Geom.	3D	Graph	Chem.	Circuit	#
C1	✓	✓	✓	✓	✓	✓	6
C2	✓	✓	✓	✓	✓	✓	6
T1	✓	✓	✓	✓	✓	✓	6
T2	✓						1
T3						✓	1
T4	✓						1
S1	✓	✓		✓		✓	4
S2		✓	✓	✓	✓		4
S3	✓						1
S4		✓		✓		✓	3
S5	✓	✓	✓	✓	✓	✓	6
S6	✓	✓	✓	✓	✓	✓	6
S7				✓			1
S8						✓	1
L1	✓			✓		✓	3
L2	✓			✓		✓	3
L3	✓	✓					2
Per domain	12	9	6	11	6	11	

Diagram-to-Code Editing (D2C-E)

Charts Case 1

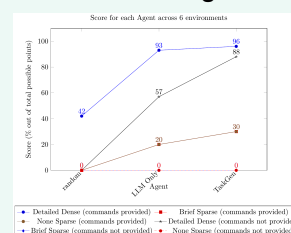
Q: Convert this grouped bar chart into a line chart, where each series in the legend is represented by a separate line connecting the data points for each agent.



A:

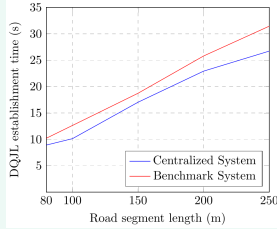
```
% [5 packages: fontenc, inputenc, xcolor...]
\begin{tikzpicture}
\begin{axis}[width=\textwidth, height=0.7\textwidth, xlabel={Agent}, ylabel={
Score (% out of total possible points)}, symbolic x coords={random, LLM
Only, TaskGen}, xtick=data, nodes near coords, ymin=0, ymax=110, legend
style={at={(0.5,-0.25)}, anchor=north, legend columns=2}, enlarge x limits
={abs=1.5cm}, ylabel near ticks, xticklabel style={rotate=45, anchor=east},
title={Score for each Agent across 6 environments}]
\addplot coordinates {(random,42) (LLM Only,93) (TaskGen,96)};
\addplot coordinates {(random,0) (LLM Only,20) (TaskGen,30)};
\addplot coordinates {(random,0) (LLM Only, 20) (TaskGen,30)};
\addplot coordinates {(random,0) (LLM Only,57) (TaskGen,88)};
\addplot coordinates {(random,0) (LLM Only,0) (TaskGen,0)};
\addplot coordinates {(random,0) (LLM Only,0) (TaskGen,0)};
\legend{Detailed Dense (commands provided), Brief Sparse (commands provided
), None Sparse (commands provided), Detailed Dense (commands not
provided), Brief Sparse (commands not provided), None Sparse (commands
not provided)}
\end{axis}
\end{tikzpicture}
```

Rendered Image:



Charts Case 2

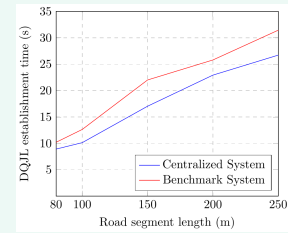
Q: Change the data value at the third data point of the Benchmark System from (150, 18.73) to (150, 22.00).



A:

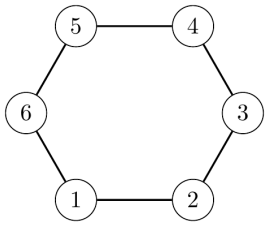
```
% [6 packages: pgfplots, amsmath, amssymb...]
\begin{tikzpicture}
\begin{axis}[xlabel={Road segment length (m)},ylabel={DQJL establishment
time (s)},xmin=80,xmax=250,ymin=0,ymax=35,xtick
={80,100,150,200,250},ytick={5, 10, 15, 20, 25, 30, 35},legend pos=south
east,ymajorgrids=true,xmajorgrids=true,grid style=dashed\addplot[color=
blue]coordinates{(80, 8.91)(100, 10.13)(150, 17.03)(200, 22.92)(250, 26.71)
};
\addplot coordinates{(80, 10.2)(100, 12.63)(150, 22.00)(200, 25.79)(250, 31.43)
};
\legend{Centralized System, Benchmark System}
\end{axis}
\end{tikzpicture}
```

Rendered Image:



Graph Structures Case 1

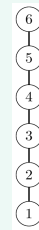
Q: Convert the current hexagonal cycle layout into a linear vertical layout where the nodes are connected in sequence from 6 at the top to 1 at the bottom.



A:

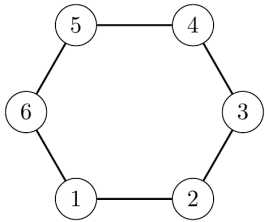
```
% [3 packages: amsmath, amssymb, tikz,color]
\begin{tikzpicture}[scale=0.65]
\node [draw, circle] (1) at (0,0) {$1$};
\node [draw, circle] (2) at (0,1.5) {$2$};
\node [draw, circle] (3) at (0,3) {$3$};
\node [draw, circle] (4) at (0,4.5) {$4$};
\node [draw, circle] (5) at (0,6) {$5$};
\node [draw, circle] (6) at (0,7.5) {$6$};
\draw [thick] (6)---(5)---(4)---(3)---(2)---(1);
\end{tikzpicture}
```

Rendered Image:



Graph Structures Case 2

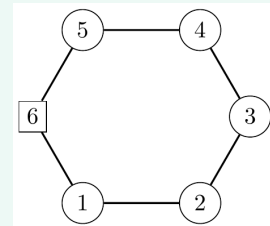
Q: Change the shape of node 6 from a circle to a rectangle.



A:

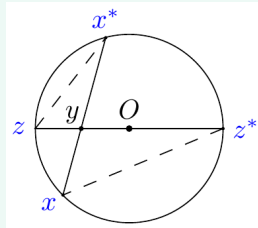
```
% [3 packages: amsmath, amssymb, tikz,color]
\begin{tikzpicture}[scale=0.65]
\node [draw, circle] (1) at (-1.35,0) {$1$};
\node [draw, circle] (2) at (1.35,0) {$2$};
\node [draw, circle] (3) at (2.5,2) {$3$};
\node [draw, circle] (4) at (1.35,4) {$4$};
\node [draw, circle] (5) at (-1.35,4) {$5$};
\node [draw, rectangle] (6) at (-2.5,2) {$6$};
\draw [thick] (1)---(2)---(3)---(4)---(5)---(6)---(1);
\end{tikzpicture}
```

Rendered Image:



Planar Geometry Case 1

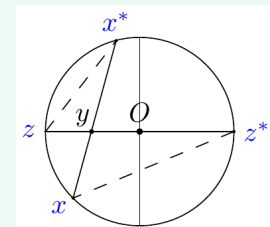
Q: Add a vertical diameter passing through the center point O to the diagram.



A:

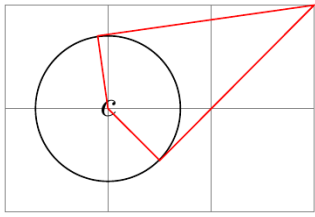
```
% [1 packages: lmodern,tikz]
\begin{tikzpicture}[scale=.3]
\draw(0,0)circle(4.01cm);
\draw(-1.387)---(-2.83,-2.83);
\draw(4.01,0)---(-4.01,0);
\draw(0,4.01)---(0,-4.01);
\draw[dash pattern=on 5pt off 5pt](-4.01,0)---(-1.387);
\draw[dash pattern=on 5pt off 5pt](4.01,0)---(-2.83,-2.83);
...
\fill(4.01,0)circle(2pt)node[blue,right]{$z^*$};
\fill(-4.01,0)circle(1.5pt)node[blue,left]{$z$};
\fill(-2.05,0)circle(3pt);
\node[above left]at(-1.7,-.2){$y$};
\fill(0,0)circle(4pt)node[above]{$O$};
\end{tikzpicture}
```

Rendered Image:



Planar Geometry Case 2

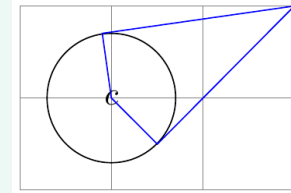
Q: Change the stroke color of the red path connecting the tangents and the circle center to blue.



A:

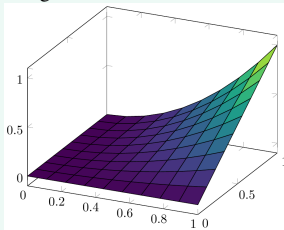
```
% [1 packages: tikz]
\begin{tikzpicture}
\draw[help lines] (0,0) grid (3,2);
\coordinate (a) at (3,2);
\node[circle,draw] (c) at (1,1) [minimum size=40pt] {$c$};
\draw[blue] (a) -- (tangent cs:node=c,point={(a)},solution=1) --
(c.center) -- (tangent cs:node=c,point={(a)},solution=2) -- cycle;
\end{tikzpicture}
```

Rendered Image:



3D Shapes Case 1

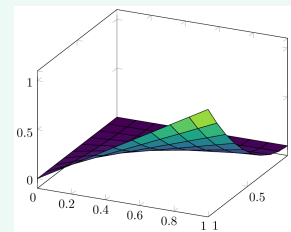
Q: Rotate the 3D surface plot by 90 degrees around the vertical axis.



A:

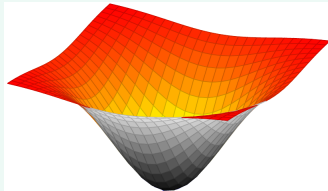
```
% [1 packages: pgfplots]
\begin{tikzpicture}
\begin{axis}[colormap/viridis, view={115}{30}]
\addplot3 [
surf,
shader=flat,
draw=black,
samples=10,
domain=0:1,
] {x^2*y};
\end{axis}
\end{tikzpicture}
```

Rendered Image:



3D Shapes Case 2

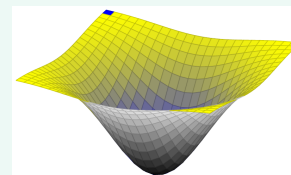
Q: Change the interior colormap of the 3D surface from 'hot' to 'plasma'.



A:

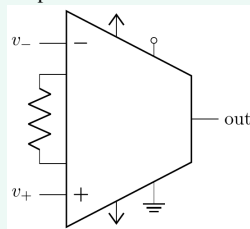
```
% [1 packages: pgfplots]
\begin{tikzpicture}
\begin{axis}[
hide axis,
xlabel=$x$,ylabel=$y$,
mesh/interior colormap name=plasma,
colormap/blackwhite,
]
\addplot3 [domain=-1.5:1.5,surf]
{-exp(-x^2-y^2)};
\end{axis}
\end{tikzpicture}
```

Rendered Image:



Circuit Diagrams Case 1

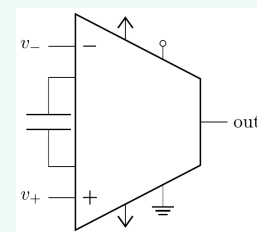
Q: Replace the resistor connected between the ra- and ra+ pins with a capacitor.



A:

```
% [3 packages: fontenc, inputenc, circuitikz]
\begin{circuitikz}
\draw
(0,0) node[inst amp ra] (opamp) {}
(opamp.+) node[left] {$v_+$}
(opamp.-) node[left] {$v_-$}
(opamp.out) node[right] {out}
(opamp.up) node[vcc] {}
(opamp.down) node[vcc] {}
(opamp.refv down) node[ground] {}
(opamp.refv up) to[short, -o] ++(0,0.3)
(opamp.ra-) to[C] (opamp.ra+);
\end{circuitikz}
```

Rendered Image:



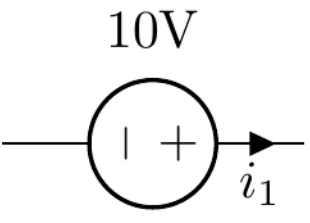
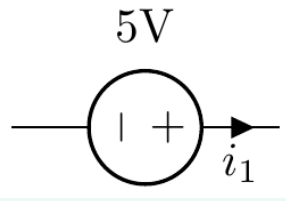
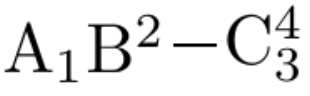
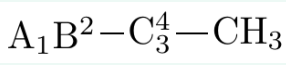
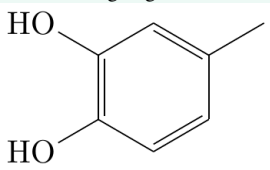
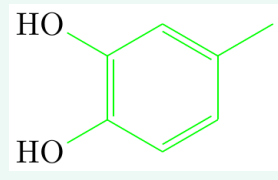
Circuit Diagrams Case 2 Q: Change the parameter of the voltage source from 10V to 5V. 	A: <pre>% [4 packages: fontenc, inputenc, lmodern...] \begin{circuitikz}[american] % Standard independent voltage source with current arrow \draw (0,0) to[V=5V, i_=\$i_1\$] (2,0); \end{circuitikz}</pre>	Rendered Image: 
Chemistry Case 1 Q: Add a single bond and a methyl group (-CH3) attached to 'C_3^4' to the diagram. 	A: <pre>% [1 packages: chemfig] \chemfig{A_1B^2-C_3^4-CH_3}</pre>	Rendered Image: 
Chemistry Case 2 Q: Change the stroke color of the benzene ring to green. 	A: <pre>% [1 packages: chemfig] \setchemfig{atom sep=2em,bond offset=1pt,bond style={green}} \chemfig{*6((-HO)-(-)-(-)-(-HO)=)}</pre>	Rendered Image: 

Figure B.2 Representative D2C-E (Diagram-to-Code Editing) examples. Given an input diagram and an editing instruction, the task requires modifying the TikZ code to produce the desired change while preserving unaffected elements.

B.6 DQA Templates

The DQA uses 60 manually designed question templates across the six diagram domains, including descriptive questions (23 templates) that assess the model's ability to identify domain-specific symbols and extract information; standard reasoning questions (18 templates) that require numerical computation with domain-specific formulas; and what-if reasoning questions (19 templates) that require predicting answers conditioned on hypothetical element modifications. Each template encodes the relevant domain knowledge. Table B.6 summarizes the per-domain distribution. We instruct models with the expected output format at inference, and each question specifies one of three output formats:

- **OI-NUM:** A single numerical value with units if applicable (e.g., “100 Ω”, “90 deg”, “3.14”). Fallback: NOT_PRESENT if information is absent from the diagram.
- **OI-TERM:** A standard scientific term or exact label (e.g., “Parallelogram”, “Diode”, “Hydroxyl”). Fallback: NOT_PRESENT.
- **OI-LIST:** A list of strings sorted alphabetically (e.g., [‘Node A’, ‘Node C’]). Fallback: empty list [].

Table B.6 Question template and instruction counts for DQA across six diagram domains. Templates cover descriptive (Desc.) and reasoning (Reas.) types, where reasoning includes standard (Std.) and what-if sub-types.

Domain	Template Count			Total	Output Instruction		
	Desc.	Std./Reas.	What-if/Reas.		NUM	TERM	LIST
Charts	4	4	3	11	9	2	0
Planar Geom.	5	3	3	11	6	5	0
3D Shapes	2	5	4	11	7	4	0
Graph Struct.	4	2	3	9	8	0	1
Chemistry	4	2	3	9	4	5	0
Circuit	4	2	3	9	5	4	0
Total	23	18	19	60	39	20	1

Question templates. (1) Table B.7 presents 41 question templates: 23 descriptive and 18 standard reasoning. Descriptive templates assess symbol identification (e.g., recognizing a zigzag as a resistor, a right-angle square as 90°), element counting (e.g., vertices, atoms, faces), and pattern recognition (e.g., data trends, series/parallel topology). Standard reasoning templates require applying domain-specific formulas (e.g., $V = IR$ for circuits, $A = \frac{1}{2}bh$ for triangles, interior angle sum $= (n-2) \times 180^\circ$) or performing visual reasoning (e.g., shortest path length, cross-section shape identification); (2) Table B.9 presents 19 what-if templates for the reasoning questions. Each poses a hypothetical modification to the diagram (e.g., changing a data value, removing an element, scaling a dimension) and asks about the consequence. These modifications are deliberately aligned with D2C-E editing operations: for instance, a what-if question about removing a node (aligned with scope task S5) tests whether the model can reason about connectivity changes, while a question about modifying a resistance value (aligned with text task T3) tests circuit analysis under parameter changes.

Domain knowledge. Each template encodes domain-specific knowledge required for correct answers. The knowledge spans: graph theory concepts (degree, in-/out-degree, connected components, shortest path) for Graph Structures; statistical measures (mean, median, range, monotonicity) for Charts; Euclidean geometry (shape classification, perimeter, area, interior angle sum) for Planar Geometry; circuit analysis (Ohm’s law $V=IR$, series/parallel resistance, power) for Circuits; solid geometry (face/vertex counting, volume, surface area, cross-sections) for 3D Shapes; and organic chemistry conventions (implicit hydrogens from valence rules, bond types, functional group identification, molecular weight) for Chemistry. We summarize the domain knowledge per diagram type in Table 2a.

Table B.7 Question templates in DQA (Part 1: Graph Structures, Charts, Planar Geometry). “D” = descriptive; “R” = standard reasoning.

ID	Type	Name	Question Template	OI
Graph Structures				
Q1	D	Degree Query	What is the degree of the node labeled $[label]$?	NUM
Q2	D	In-degree Query	What is the in-degree of the node labeled $[label]$?	NUM
Q3	D	Out-degree Query	What is the out-degree of the node labeled $[label]$?	NUM
Q4	D	Neighbors Query	List all nodes that are direct neighbors of $[label]$.	LIST
Q5	R	Path Length	What is the length of the shortest path from $[node_A]$ to $[node_B]$?	NUM
Q6	R	Tree Height	What is the height of this tree?	NUM
Charts				
Q1	D	Max/Min Query	Which data point/category has the maximum (or minimum) value?	TERM
Q2	R	Mean Query	What is the mean (average) value of all visible data points?	NUM
Q3	R	Median Query	What is the median value of the dataset?	NUM
Q4	R	Range Query	What is the range (max – min) of the dataset?	NUM
Q5	D	Monotonicity	In the interval from $[start]$ to $[end]$, what trend does the data exhibit?	TERM
Q6	R	Extrema Location	At approximately what x-coordinate does the function reach its minimum?	NUM
Q7	D	Heatmap Value	According to the color scale, what is the approximate value at cell $([row], [col])$?	NUM
Q8	D	Polar Plot	At angle $\theta = [angle]^\circ$, what is the radial distance r ?	NUM
Planar Geometry				
Q1	D	Right Angle Symbol	What angle measure does the small square symbol at $[location]$ indicate?	NUM
Q2	D	Equal Length Marks	Sides $[A]$ and $[B]$ have identical tick marks. What does this indicate?	TERM
Q3	D	Dashed Line	What does the dashed line in this geometric diagram typically represent?	TERM
Q4	D	Triangle Type	Based on the marked information, what type of triangle is $[label]$?	TERM
Q5	D	Quadrilateral Type	Based on the marked properties, what type of quadrilateral is $[label]$?	TERM
Q6	R	Perimeter	If side $[A] = [v1]$ and side $[B] = [v2]$, what is the perimeter of rectangle $[rect]$?	NUM
Q7	R	Area	If base $[b] = [v1]$ and height $[h] = [v2]$, what is the area of triangle $[tri]$?	NUM
Q8	R	Interior Angle Sum	What is the sum of interior angles of this $[n]$ -sided polygon?	NUM

Table B.8 Question templates in DQA (Part 2: Circuit Diagrams, 3D Shapes, Chemistry). “D” = descriptive; “R” = standard reasoning.

ID	Type	Name	Question Template	OI
Circuit Diagrams				
Q1	D	Resistor ID	What component is represented by the zigzag symbol labeled $[label]$?	TERM
Q2	D	Capacitor ID	What component is represented by the two parallel lines labeled $[label]$?	TERM
Q3	D	Diode ID	What component is represented by the triangle-with-line symbol labeled $[label]$?	TERM
Q4	D	Series/Parallel	Are components $[comp_A]$ and $[comp_B]$ connected in series or parallel?	TERM
Q5	R	Total Resistance	If $[R1] = [v1]\Omega$ and $[R2] = [v2]\Omega$ are in series, what is the total resistance?	NUM
Q6	R	Ohm’s Law	If $[R]$ has resistance $[val]\Omega$ and current $[I]$ A flows through it, what is the voltage?	NUM
3D Shapes				
Q1	D	Solid Identification	What is the name of this 3D solid?	TERM
Q2	R	Face Count	How many faces does this solid have?	NUM
Q3	R	Vertex Count	How many vertices does this solid have?	NUM
Q4	D	Hidden Edges	What do the dashed lines in this 3D diagram represent?	TERM
Q5	R	Cross-section	If this $[solid]$ is cut horizontally, what is the shape of the cross-section?	TERM
Q6	R	Volume	If the cylinder has radius $[r]$ and height $[h]$, what is its volume? (Use $\pi = 3.14$)	NUM
Q7	R	Surface Area	If the cube has edge length $[e]$, what is its surface area?	NUM
Chemistry				
Q1	D	Unlabeled Vertex	What atom does the unlabeled vertex at $[location]$ represent?	TERM
Q2	D	Bond Type	What type of bond connects atom $[A]$ to atom $[B]$?	TERM
Q3	R	Implicit Hydrogens	How many implicit hydrogen atoms are bonded to the carbon at position $[pos]$?	NUM
Q4	D	Functional Group	What functional group is present at $[location]$?	TERM
Q5	D	Carbon Count	How many carbon atoms are in this molecule (including labeled and unlabeled)?	NUM
Q6	R	Molecular Formula	What is the molecular formula of this compound?	TERM

Table B.9 DQA what-if question templates. Each template specifies a hypothetical modification and asks about its consequence.

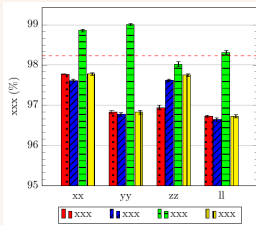
ID	Name	Question Template	OI
Graph Structures			
W1	Node Removal	Remove node $[target]$ and all its edges. How many connected components remain?	NUM
W2	Edge Addition	Add a new edge between $[node_A]$ and $[node_B]$. What is the new degree of $[node_A]$?	NUM
W3	Edge Weight Mod.	Change the edge weight from $[A]$ to $[B]$ from $[old]$ to $[new]$. What is the new total weight of the path from $[start]$ to $[end]$?	NUM
Charts			
W1	Value Modification	Change data point $[label]$ from $[old]$ to $[new]$. What is the new mean of the dataset?	NUM
W2	Data Removal	Remove data point $[target]$. What is the new maximum value?	NUM
W3	Uniform Shift	Increase all data values by $[\delta]$. What is the new range of the dataset?	NUM
Planar Geometry			
W1	Side Scaling	Scale side $[label]$ by a factor of $[k]$. What is the new perimeter?	NUM
W2	Uniform Scaling	Scale all sides of the $[shape]$ by a factor of $[k]$. By what factor does the area increase?	NUM
W3	Angle Modification	Change angle $[A]$ from $[old]^\circ$ to 90° . What type of triangle does $[tri]$ become?	TERM
Circuit Diagrams			
W1	Resistance Mod.	Change $[R]$ from $[old]$ to $[new]\Omega$. If voltage is $[V]V$, what is the new current?	NUM
W2	Short Circuit	Short-circuit $[R]$ (0Ω). What is the current through $[R'] = [val]\Omega$ at $[V]V$?	NUM
W3	Parallel Addition	Add $[val]\Omega$ in parallel with $[R]$ (also $[val]\Omega$). What is the equivalent resistance?	NUM
3D Shapes			
W1	Radius Scaling	Scale the cylinder radius by factor $[k]$, keeping height constant. By what factor does the volume increase?	NUM
W2	Uniform Scaling	Scale all dimensions of the rectangular prism by factor $[k]$. By what factor does the surface area increase?	NUM
W3	Cone Slicing	Remove the top half of the cone by slicing at mid-height. What is the remaining solid called?	TERM
W4	Stacking	Stack an identical $[shape]$ on top. What is the total volume of the combined structure?	NUM
Chemistry			
W1	Bond Type Change	Change the bond between $[A]$ and $[B]$ from single to double. How many implicit H on $[A]$?	NUM
W2	Substituent Repl.	Replace $[old_group]$ at $[pos]$ with $[new_group]$. What is the change in molecular weight?	NUM
W3	Oxygen Insertion	Insert an O atom between $[A]$ and $[B]$ to form $[A]-O-[B]$. What functional group is this?	TERM

B.7 DQA Descriptive Question Examples

Figure B.3 presents representative DQA descriptive question examples. These questions assess the model's ability to identify domain-specific symbols and extract information directly from the diagram.

Descriptive Questions in Diagram Question Answering

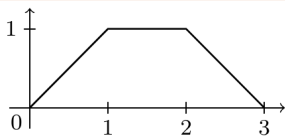
Charts Case 1



Q: Which category and bar style (by color/pattern) has the maximum value?

A: green with horizontal lines

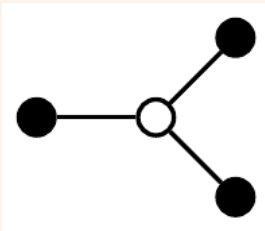
Charts Case 2



Q: In the interval from 0 to 1, what trend does the data exhibit?

A: increasing

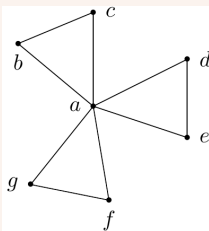
Graph Structures Case 1



Q: What is the degree of the white node?

A: 3

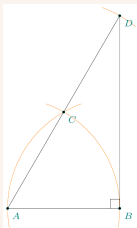
Graph Structures Case 2



Q: List all nodes that are direct neighbors of b.

A: a, c

Planar Geometry Case 1



Q: What angle measure does the small square symbol at vertex B indicate?

A: 90 deg

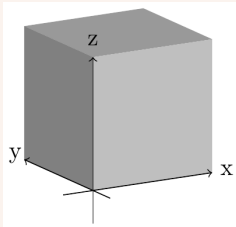
Planar Geometry Case 2



Q: Based on the marked information, what type of triangle is formed by the segments labeled $\cos \alpha$, $\sin \alpha$, and the radius of the circle?

A: right triangle

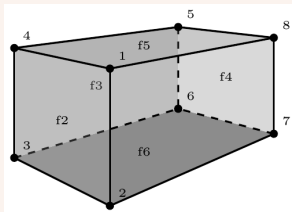
3D Shapes Case 1



Q: What is the name of this 3D solid?

A: cube

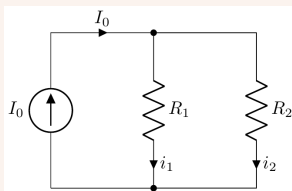
3D Shapes Case 2



Q: What do the dashed lines in this 3D diagram represent?

A: hidden edges

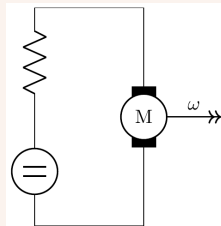
Circuit Diagrams Case 1



Q: What type of component is represented by the zigzag symbol labeled R_1 ?

A: resistor

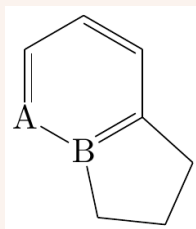
Circuit Diagrams Case 2



Q: Are the resistor and the component labeled M connected in series or parallel?

A: series

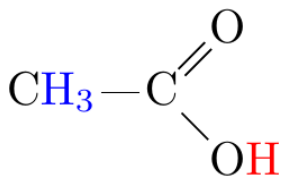
Chemistry Case 1



Q: What type of bond connects atom A to atom B ?

A: single bond

Chemistry Case 2



Q: What functional group is present at the terminal position on the right?

A: carboxyl group

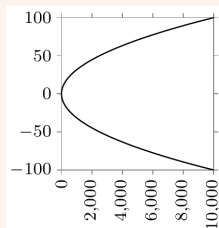
Figure B.3 Representative DQA descriptive question examples. Each case shows a diagram and a question that requires direct observation and information extraction from the visual content.

B.8 DQA Reasoning Question Examples

Figure B.4 presents representative DQA reasoning question examples. These questions require numerical computation with domain-specific formulas or predicting outcomes of hypothetical modifications.

Reasoning Questions in Diagram Question Answering

Charts Case 1



Q: What is the range (difference between maximum and minimum) of the vertical y-axis values shown in the dataset?

A: 200

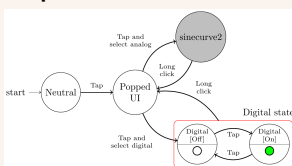
Charts Case 2



Q: Consider the following modification to the dataset in the top-right circular radar chart: Change the value of data point 'Money' from 5 to 10. (Type: T4) What is the new mean of the entire dataset?

A: 7

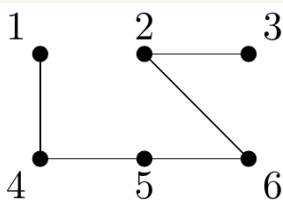
Graph Structures Case 1



Q: What is the length of the shortest path from Neutral to Digital [On]?

A: 3

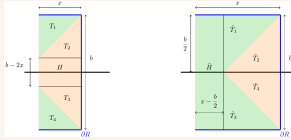
Graph Structures Case 2



Q: Consider the following modification: Remove the node 4 and all its connected edges. (Type: T5) How many connected components remain in the graph?

A: 2

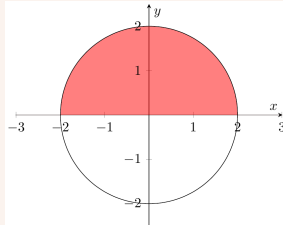
Planar Geometry Case 1



Q: In the diagram on the right, if the vertical base of triangle T~2 along the right edge is $b = 10$ units and its horizontal height from the vertex is 5 units, what is the area of triangle T~2?

A: 12.5

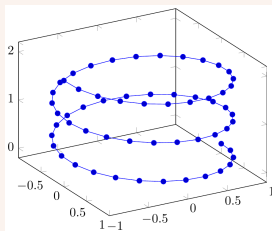
Planar Geometry Case 2



Q: Consider the following modification: Scale the radius of the circle (currently 2 units) by a factor of 2. (Type: T2) What is the new perimeter of the full circle?

A: 8π

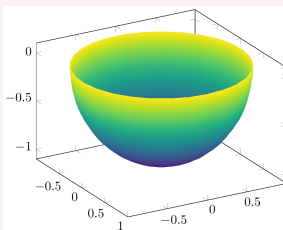
3D Shapes Case 1



Q: If the cylinder that bounds this helix has radius 1 and height 2, what is its volume? (Use $\pi = 3.14$)

A: 6.28

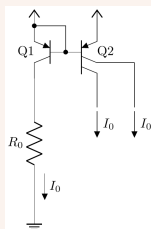
3D Shapes Case 2



Q: If this hemisphere is cut horizontally, what is the shape of the cross-section?

A: circle

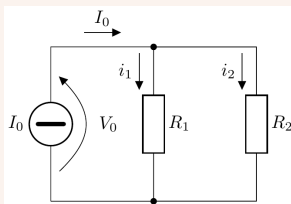
Circuit Diagrams Case 1



Q: If resistor R_0 has resistance 1000 Ohms and current $I_0 = 0.005$ A flows through it, what is the voltage across it?

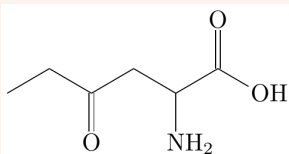
A: 5V

Circuit Diagrams Case 2



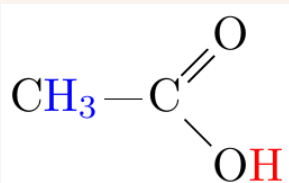
Q: Consider the following modification: Add a resistor of 100 Ohms in parallel with the existing resistor R_1 (also 100 Ohms). (Type: T9) What is the equivalent resistance of this parallel combination?

A: 50 Ohm

Chemistry Case 1

Q: Consider the following modification: Change the single bond between the two carbon atoms at the far left of the chain from a single bond to a double bond. (Type: T7) How many implicit hydrogens are now bonded to the terminal carbon atom at the far left?

A: 2

Chemistry Case 2

Q: Consider the following modification: Replace the -OH group at the bottom right with a -Cl atom. (Type: T9) What is the change in molecular weight (New - Old)?

A: 18.44 g/mol

Figure B.4 Representative DQA reasoning question examples. Each case shows a diagram and a question that requires domain-specific reasoning, computation, or hypothetical analysis beyond direct observation.

C Evaluation Methodology

C.1 Semantic Object Model (SOM) Pipeline

Our object-based metric evaluates whether the model correctly perceives basic objects that the code draws. We parse both generated and ground-truth TikZ code into a set of graphical objects—their type, text content, color, and bounding box—and compute F1 scores for each dimension. The extraction proceeds through three stages: semantic injection, compilation, and DOM-based extraction.

Stage 1: Semantic injection via `semantic_spy.sty`. Before compilation, we preprocess the TikZ source by injecting a custom L^AT_EX style file, `semantic_spy.sty`, as the last loaded package in the preamble. This package hooks into TikZ, pgfplots, and CircuiTikZ commands via `\special{dvisvgm:raw ...}` directives, which embed semantic XML tags directly into the DVI output. Specifically, it installs the following hooks:

- **TikZ core hooks.** Every `\node` is wrapped in `<gclass="tikz-node">`. The wrapper stores `data-id`, `data-shape`, and optionally `data-text`. Similarly, every `tikzpicture` and `scope` environment is wrapped with corresponding semantic group tags.
- **pgfplots data hooks.** Every axis environment is wrapped with `<gclass="pgf-axis">`. Within each `\addplot`, scatter marker hooks inject `<gclass="data-point">` tags with original coordinates (`data-x`, `data-y`) and parent-series identifiers. Each plot series is wrapped with `<gclass="pgf-series">`.
- **CircuiTikZ component hooks.** Every bipole component is wrapped with `<gclass="circuit-component">`. The wrapper carries a unique component identifier.

The preprocessor also handles code standardization: wrapping fragments in standalone documents, detecting and injecting missing packages/libraries, fixing common compilation issues (duplicate packages, xcolor option clashes, pgfplots compatibility version downgrades), and checking DVI compatibility. If the code is incompatible with DVI mode (e.g., uses `fontspec` or `xeCJK`), semantic injection is skipped and the pipeline falls back to geometry-only extraction.

Stage 2: Compilation and SVG conversion. The preprocessed code is compiled to DVI format using the `latex` engine, which preserves the `\special` directives containing the semantic tags. If DVI compilation fails, the pipeline falls back to PDF-based compilation with multi-engine fallback (`pdflatex` → `lualatex` → `xelatex`), which still produces geometry-only SVG. The DVI (or PDF) output is then converted to SVG using `dvisvgm` with parameters `--font-format=svg` (to preserve text elements), `--precision=6`, and `--zoom=1` (for 1:1 coordinate mapping). For DVI input, `dvisvgm` faithfully translates the injected `\special` tags into SVG group elements with the corresponding CSS classes and data attributes. For PDF input, `dvisvgm--pdf` produces geometric SVG without semantic annotations.

Stage 3: DOM-based SOM extraction. The generated SVG is parsed using an lxml-based DOM parser. The

Table C.1 Primary SOM element types extracted per domain. All domains share basic NODE, PATH, and TEXT types; domain-specific types arise from specialized packages.

Domain	Primary TikZ Pkg.	Key SOM Element Types
Charts	pgfplots	AXIS, DATASERIES, DATAPOINT, NODE (legend), TEXT (tick labels)
Graph Struct.	tikz	NODE (circle, rectangle, diamond), PATH (open/closed), TEXT (labels)
Planar Geom.	tikz/tkz-euclide	NODE (point markers), PATH (edges, arcs), TEXT (vertex labels, annotations)
Circuit	circuitikz	COMPONENT (R, C, L, V, etc.), NODE (junctions), TEXT (value labels)
3D Shapes	pgfplots/tikz	AXIS, PATH (surfaces, wireframes), NODE (labels), FILL/FILLDRAW
Chemistry	chemfig	NODE (atoms, functional groups), PATH (bonds), TEXT (atom labels)

extractor traverses the SVG tree recursively and classifies each element into one of the following categories:

- **Semantic elements:** identified by CSS class attributes injected by `semantic_spy.sty`. These include `tikz-node` (NODE), `tikz-path` (PATH), `pgf-axis` (AXIS), `pgf-series` (DATASERIES), `data-point` (DATAPOINT), and `circuit-component` (COMPONENT).
- **Native SVG shapes:** identified by SVG tag names (`rect`, `circle`, `ellipse`, `path`, `text`, etc.), mapped to corresponding geometric types.

For each element, the extractor records four attributes:

- **Type.** A fine-grained type key that combines the coarse element type with subtype attributes, e.g., `node:circle`, `node:rectangle`, `path:closed`, `path:open`, and `component:R` (resistor). Structural container types such as `picture`, `scope`, and `axis` are excluded from type-level evaluation because they would inflate the score.
- **Text.** Text content is extracted with the following priority: (1) the `data-text` attribute from semantic injection; (2) the `metadata` child element; (3) glyph reconstruction from `<use>` elements (see below); and (4) fallback to `<text>` or `<tspan>` children. For DATAPOINT elements, the (x, y) data values are formatted as text for matching.
- **Color.** Extracted from `fill` and `stroke` attributes for path elements, and from `fill` for text elements. Colors specified as `none` or `transparent` are excluded.
- **BBox.** Axis-aligned bounding boxes are computed from SVG geometry: directly from coordinate attributes for basic shapes (`rect`, `circle`, `ellipse`), from path data using `svgpathtools` for `<path>` elements, and recursively aggregated for group elements. Cumulative SVG transforms (`translate`, `scale`, `matrix`) are tracked through the DOM tree.

Glyph text reconstruction. A key challenge is that `dvipdfm` converts text into glyph paths stored in `<defs>` and referenced by `<use>` elements. Their IDs follow the pattern `g{font_id}-{char_code}`. To recover text content, we collect all `<use>` elements within a group, sort them by x -position (left to right), decode the character codes from glyph IDs, and concatenate the characters. For example, `<usehref="#g0-65"/>` corresponds to character code 65 (“A”). This reconstruction handles both single-character and multi-character text groups, including colored text whose parent `<g>` carries a `fill` attribute.

Per-domain element mapping. The SOM pipeline maps diagrams from different TikZ packages into a unified element schema. Table C.1 summarizes the primary element types extracted per domain.

C.2 Evaluation Metrics

Object-based metric computation. Formally, given TikZ source code c in domain d , a domain extractor $\mathcal{E}_d$ maps code to attribute sets:

$$\mathcal{E}_d(c) = \{ \mathcal{T}_d(c), \mathcal{X}_d(c), \mathcal{K}_d(c), \mathcal{B}_d(c) \}. \quad (1)$$

Here, $\mathcal{T}_d(c)$ is a multiset of primitive types; $\mathcal{X}_d(c)$ is a multiset of textual elements; $\mathcal{K}_d(c)$ is a multiset of normalized color specifications; and $\mathcal{B}_d(c)$ is a set of axis-aligned bounding boxes $b_i = (x, y, w, h)$, with (x, y) denoting the center coordinates and w and h representing the width and height.

Primitive F1 Score. Given predicted code $\hat{c}$ and ground-truth code c , we extract 4 graphic objects via $\mathcal{E}_d$. Let M_* denote the number of correct matches (defined below), $|\hat{S}|$ the predicted count, and $|S|$ the ground-truth count. The per-dimension F1 is defined as follows, where the subscript $*$ is replaced by each instance in $\{type, text, color, bbox\}$.

$$P_* = \frac{M_*}{|\hat{S}|}, \quad R_* = \frac{M_*}{|S|}, \quad F1_* = \frac{2P_*R_*}{P_* + R_*}. \quad (2)$$

The four dimensions differ only in how M_* is computed: (1) Count-based multiset matching for type. Let $\hat{n}(t)$ and $n(t)$ be the counts of type $t \in \mathcal{T}_d$ in predicted and ground-truth code:

$$M_{type} = \sum_{t \in \mathcal{T}_d} \min(\hat{n}(t), n(t)). \quad (3)$$

(2) Greedy one-to-one exact string matching for text. Each ground-truth text element matches at most one predicted element by exact string equality: $M_{text} = |\text{matched pairs}|$. (3) Permutation-based optimal assignment for color [14]. Colors are grouped by associated type of element. Within each group g , the optimal permutation π of predicted colors is selected using CIE2000 perceptual similarity:

$$M_{color} = \sum_g \max_{\pi} \sum_i \text{sim}(\pi(\hat{k}_i^g), k_i^g), \quad \text{sim}(\hat{k}, k) = \max\left(0, 1 - \frac{\Delta E_{00}(\hat{k}, k)}{100}\right). \quad (4)$$

(4) Greedy one-to-one IoU matching for BBox. For each ground-truth box b (iterated in order), we search the remaining unmatched predicted boxes and select the first $\hat{b}$ whose Intersection-over-Union (IoU) exceeds a threshold $\tau=0.3$: $\text{IoU}(\hat{b}, b) = \frac{|\hat{b} \cap b|}{|\hat{b} \cup b|} \geq \tau$. The matched predicted box is then removed from the candidate pool. The final matching count is $M_{bbox} = |\text{matched pairs}|$.

The primary perception-level metric is the average across the four objects:

$$\text{F1}_{avg} = \frac{1}{4} (\text{F1}_{type} + \text{F1}_{text} + \text{F1}_{color} + \text{F1}_{bbox}). \quad (5)$$

Code-based metric computation. We use CrystalBLEU [36] to measure token-level similarity between generated and ground-truth TikZ code. CrystalBLEU extends standard BLEU-4 by filtering out trivially shared n-grams—high-frequency, low-information tokens such as `{`, `}`, `[`, `]`, `,`, `;`, `=`, `\begin`, and `\end`—that would inflate scores without reflecting meaningful code similarity. Tokenization is performed using the Pygments `TexLexer`, which correctly segments L^AT_EX control sequences, and tokens of type `Whitespace` and `Comment` are discarded.

For the editing task, CrystalBLEU is also decomposed into preserve-only and edit-only variants. The edit-only CrystalBLEU performs multiset subtraction of source n-grams from both GT and generated n-grams before computing BLEU, isolating the contribution of newly introduced code. The preserve-only variant keeps only n-grams that intersect with the source (Counter intersection), measuring how faithfully the model retains unchanged code portions.

Image-based metric computation. We compile both generated and ground-truth TikZ code into PNG images (300 DPI, cropped whitespace) and compare them using four established image similarity metrics. The rendering pipeline uses `latexmk` with multi-engine fallback (`pdflatex` $\rightarrow$ `lualatex` $\rightarrow$ `xelatex`), followed by PDF cropping (via `pdfCropMargins`) and `pdftoppm` conversion to PNG. All images are resized to 256×256 pixels with white background before metric computation.

- **SSIM** [32]. Structural Similarity Index Measure, computed via `torchmetrics` with data range normalized to $[0, 1]$. Compares brightness, contrast, and structural patterns between two images (higher is better).
- **CLIP Score** [33]. Cosine similarity between CLIP ViT-B/32 image embeddings of the generated and ground-truth images. Captures high-level semantic similarity in a shared vision-language embedding space (higher is better).
- **LPIPS** [34]. Learned Perceptual Image Patch Similarity, computed using an AlexNet backbone. Measures perceptual distance using deep features that correlate with human perceptual judgments (lower is better).
- **FID** [35]. Fréchet Inception Distance, a distribution-level metric computed over all generated and ground-truth images within each model’s evaluation set using `torchmetrics` with Inception v3 features (2048-dim). Requires ≥ 2 images per set (lower is better).

In the main results (Table 5), we report two composite image-level scores: SC (the average of SSIM and CLIP, higher is better) and FL (the average of FID and LPIPS, lower is better).

Edge cases. When both the predicted and ground-truth sets are empty for a given dimension (e.g., neither code contains any text elements), we define $\text{F1} = 1.0$ (perfect match), since the model correctly produces no elements of that type. When the color group contains ≤ 6 elements, we use exhaustive permutation search; for larger groups (> 6 elements), we switch to greedy approximation to maintain computational tractability.

Preserve/edit splits in D2C-E. For the diagram-to-code editing task, a single overall score is insufficient: an edit typically changes only a small portion of the diagram, so a model that ignores the editing instruction and simply reproduces the original diagram could already score high. We therefore split evaluation into two complementary parts:

(1) Preserve-only evaluation measures how well the model retains elements that should remain unchanged after editing. For each dimension, we keep only elements present in both the generated code and the original source code (intersection), then compute F1 score against the ground-truth preserved elements. Formally, for a given dimension:

- GT preserve set: GT elements that match source elements.
- Gen preserve set: Generated elements that match source elements.
- F1 score is computed between the GT preserve set and the Gen preserve set.

(2) Edit-only evaluation isolates the quality of the actual edits. For each dimension, elements common between the source and the ground-truth are subtracted from both the ground-truth and the generated code before computing F1 score. This eliminates the contribution of unchanged elements and focuses exclusively on what the editing instruction asks to change. Formally:

- Common set: Elements present in both source and GT (computed via greedy matching).
- GT edit set: GT elements after removing the common set.
- Gen edit set: Generated elements after removing the common set.
- F1 score is computed between the GT edit set and the Gen edit set.

We apply the same procedure for object- and code-based metric computation on both preserve and edit splits.

C.3 DQA LLM-as-a-Judge Details

We evaluate DQA responses through a two-stage pipeline: rule-based matching followed by LLM-as-a-judge fallback for ambiguous cases.

Stage 1: Answer extraction. The model’s raw output is processed to extract the final answer. Models are instructed to return JSON `{"answer": "...", "reasoning": "..."}` , but actual outputs may contain markdown code blocks, extra text, or malformed JSON. The extractor tries four strategies in order: (1) direct JSON parsing; (2) extraction from markdown `“`json ... `”` code blocks; (3) finding the outermost `{...}` braces; (4) regex fallback for `"answer": "..."` patterns. Thinking tags (`<think>...</think>`) are stripped before extraction.

Stage 2: Rule-based matching. The extracted answer is compared against the ground truth using type-specific rules based on the output instruction (OI) type:

- OI-NUM (5,013 questions, 70.2%): Numeric comparison with absolute tolerance ($\leq 10^{-6}$) or relative tolerance ($\leq 1\%$), with unit normalization (e.g., “degrees” = “deg” = “°”; “amps” = “A”).
- OI-TERM (1,779 questions, 24.9%): Case-insensitive exact string match after normalization (removing quotes, collapsing whitespace, stripping trailing punctuation).
- OI-LIST (354 questions, 5.0%): Order-independent set comparison. Lists are parsed from JSON arrays or CSV format, and items are normalized before set equality check.

Each rule produces a confidence score. When confidence is high (≥ 0.85), the rule verdict is accepted directly. When confidence is low (e.g., partial containment for terms, or unparseable candidate), the case is escalated to Stage 3.

Stage 3: LLM-as-a-judge fallback. For cases where rule-based matching yields low confidence, we use an LLM judge. Given a question, the ground-truth answer, and the model’s response, we prompt the judge LLM to classify the response as CORRECT, INCORRECT, or UNATTEMPTED. The judge prompt is type-aware: it includes OI-specific guidelines (e.g., for OI-NUM, small rounding differences are acceptable; for OI-LIST, extra or missing items make the answer incorrect). The judge outputs a structured two-line response: (1) an evaluation explaining the reasoning, and (2) a label. The final binary score is 1 for CORRECT and 0 otherwise. The full judge prompt template is shown below:

Table C.2 Human agreement validation of the DQA LLM judge. κ_{J-H} : Cohen’s Kappa between the LLM judge and human majority vote. κ_{H-H} : Fleiss’ Kappa among three human annotators. Acc: percentage agreement between judge and human majority. [†]Stage 2 κ values are deflated by extreme label skew (99%+ correct); raw accuracy better reflects the near-perfect agreement on this subset.

Subset	N	κ_{J-H}	κ_{H-H}	Acc (%)
All	200	0.937	0.915	97.5
Stage 2 (Rule)	130	0.000 [†]	0.663 [†]	99.2
Stage 3 (LLM)	70	0.839	0.805	94.3
OI-NUM	133	0.961	0.949	98.5
OI-TERM	53	0.911	0.809	96.2
OI-LIST	14	0.811	1.000	92.9

DQA LLM-as-a-Judge Prompt

Role: You are an expert judge specialized in evaluating the correctness of answers to scientific diagram understanding questions.

Task: Classify the model’s response into one of three categories:

1. **Correct:** The model answer contains the core information of the ground truth and is semantically consistent.
2. **Incorrect:** The model answer contradicts the ground truth.
3. **Unattempted:** The model explicitly states it cannot answer, or the response is empty.

Answer Type: {output_instruction}
{OI-specific guidelines}

Output Format:

1. **Evaluation:** [Brief explanation of reasoning]
2. **Label:** [Correct / Incorrect / Unattempted]

Input:

Question: {question}

Model Answer: {model_answer}

Ground Truth Answer: {ground_truth_answer}

C.4 LLM Judge Human Agreement Validation

To verify the reliability of our LLM-as-a-judge evaluation, we conduct a human agreement study following prior work that validates automated evaluators against independent human annotations and human-majority labels using agreement-based metrics [63].

Sampling. We stratify-sample 200 cases from all 7,146 DQA questions along two dimensions: (1) evaluation stage, 70 cases resolved by the LLM judge (Stage 3) and 130 by rule-based matching (Stage 2), which ensures coverage of the most subjective cases; and (2) output instruction type, 133 OI-NUM, 53 OI-TERM, and 14 OI-LIST, approximately following the corpus proportions. We use GPT-5.2 model outputs as the representative responses for annotation.

Annotation protocol. Three graduate-student annotators independently label each (question, model_answer, ground_truth) triple as CORRECT, INCORRECT, or UNATTEMPTED, following the same OI-specific grading guidelines used by the LLM judge (§C.3). Annotators are shown the diagram image, question text, ground-truth answer, and model answer, but not the automated judge’s verdict.

Agreement metrics. We report Cohen’s Kappa (κ) and raw accuracy. Specifically, κ_{J-H} measures agreement between the LLM judge and the human majority vote, while κ_{H-H} measures Fleiss’ Kappa among the three human annotators as a human reference. We additionally report raw accuracy because κ can be deflated under extreme label imbalance, even when agreement is very high [63].

Results. As shown in Table C.2, the LLM judge closely matches human consensus overall, achieving $\kappa_{J-H} = 0.937$ with 97.5% accuracy, indicating almost perfect agreement. This is comparable to the human agreement level ($\kappa_{H-H} = 0.915$), suggesting that the automated judge reliably tracks majority human judgments. For the 130 Stage 2 cases, only one judge–human disagreement occurs (99.2% accuracy); the near-zero κ is an artifact of extreme label skew rather than poor agreement. For the 70 Stage 3 cases—the most subjective subset—the judge still achieves $\kappa_{J-H} = 0.839$ (94.3% accuracy), close to the corresponding human agreement

Table D.1 Generation configurations of evaluated models. “API” denotes cloud-hosted model endpoints; “vLLM” denotes models deployed with vLLM. T : sampling temperature; $L_{\max}$: maximum output tokens.

Model	Inference	T	$L_{\max}$	Notes
Closed-Source				
Gemini-3.1 Pro	API	1.0	16,384	reasoning_effort = medium
Gemini-3.0 Pro	API	1.0	16,384	
Gemini-3.0 Flash	API	1.0	16,384	
GPT-5.2	API	1.0	16,384	
Claude-4.6 Opus	API	1.0	16,384	
Seed-2.0 Pro	API	1.0	16,384	
Open-Source				
Qwen3.5-397B-A17B	API	1.0	16,384	1×A800
Qwen3-VL-235B-A22B	API	1.0	6,144	
Kimi-K2.5	API	1.0	16,384	
Qwen3-VL-8B	vLLM	1.0	8,192	
InternVL3-38B	vLLM	1.0	8,192	
TikZero+ 10B	vLLM	0.8	4,096	

($\kappa_{H-H} = 0.805$). Across output instruction types, agreement is highest for OI-NUM ($\kappa_{J-H} = 0.961$) and lowest for OI-LIST ($\kappa_{J-H} = 0.811$), consistent with the greater ambiguity of list-valued answers.

D Implementation Details

D.1 Model Configurations

Table D.1 summarizes the generation configurations for all 12 evaluated models. Closed-source models and several large open-source models are accessed via their respective provider APIs. The remaining open-source models are deployed locally or remotely using vLLM (v0.14.0). All models use a sampling temperature of $T=1.0$ except TikZero+ 10B ($T=0.8$). The maximum output length $L_{\max}$ is set to 16,384 tokens for API-based models to accommodate long TikZ programs, while locally-served models use smaller limits (4,096–8,192) due to GPU memory constraints.

D.2 Inference and Compilation

Inference pipeline. All models receive diagram images encoded as base64 data URIs within multimodal messages. For D2C-P and D2C-E, the system prompt instructs the model to output complete $\text{\LaTeX}$ code inside a single “`“ $\text{\LaTeX}$  code block, starting from \documentclass and ending with \end{document}”`. For DQA, models return structured JSON with `answer` and `reasoning` fields. Generated code is extracted from the model response via regex-based markdown code-block parsing. Each inference call is subject to a 300-second timeout; failed calls are retried up to 3 times with exponential backoff (2, 4, 8 s delays).

Hardware. All experiments are conducted on a single machine with 8×NVIDIA A800 80 GB GPUs. Open-source models served locally with vLLM each occupy one GPU. Image-level metric computation (SSIM, LPIPS, CLIP) runs on a dedicated GPU to avoid memory contention with inference. The DQA LLM judge (Qwen3-Next-80B-A3B, §C.3) is deployed across 4 GPUs with tensor parallelism via vLLM.

$\text{\LaTeX}$ compilation. Generated TikZ code is compiled into PDF using a multi-engine fallback strategy: `pdflatex` is attempted first, followed by `lualatex` and `xelatex` if earlier engines fail. Each compilation attempt has a 60-second timeout. Compiled PDFs are cropped using `pdfCropMargins` and converted to 256×256 PNG images (300 DPI) via `pdftoppm` for image-level metric computation. For object-level F1 evaluation, code is separately compiled to DVI format and converted to SVG for Structured Object Model (SOM) extraction.

Failure handling. When compilation fails or times out, all upward metrics ($F1_{avg}$, CrystalBLEU, SSIM, CLIP) default to 0, and downward metrics (LPIPS) default to 1.0, ensuring a fixed denominator across all models. Inference failures, including API errors, empty responses, or timeouts after all retries, receive the same penalty scores. This deterministic treatment prevents missing data from inflating any model’s average performance.

E Evaluation Prompts

We list 16 evaluation settings (**S1-S16**) in Table 4. Herein, we provide the full system prompts per setting. Specifically, each inference call follows a two-message structure: (1) a system message containing the task prompt (presented below), and (2) a user message containing the diagram image (base64-encoded) together with a task-specific text query (e.g., “Convert this diagram to LaTeX TikZ code” for D2C-P, an editing instruction for D2C-E, or a question for DQA). Placeholders enclosed in braces ($\{\dots\}$) are filled at inference time with instance-specific data: $\{\text{perception_data}\}$ is a structured JSON list of visual primitives extracted from ground-truth annotations.

E.1 Foundational Evaluation Prompts

The three foundational settings use system prompts that instruct the model to perform the task directly without additional context or tools.

S1 / S6: D2C-P Direct Coding / D2C-E Direct Editing

You are a LaTeX expert. Follow these rules strictly:

1. Output the COMPLETE LaTeX code inside a single “`latex` code block.
2. Start from `\documentclass` and end with `\end{document}`.
3. Do NOT output any text, explanation, or reasoning outside the code block.

S1 and S6 share the same system prompt; the task is distinguished by the user message: S1 sends only the diagram image, while S6 additionally includes an editing textual instruction.

S12: DQA Direct Answering

You are a scientific diagram analysis expert. Answer the question based on the image and follow the required answer format exactly.

E.2 Agentic Evaluation Prompts

The agentic settings augment the foundational prompts with one or more capabilities: context utilization (providing structured object data), tool use (TikZ documentation search via MCP), state management (requiring intermediate code generation), and planning (coordinating multiple capabilities). Below, prompts are organized by task.

E.2.1 Diagram-to-Code Parsing (S2–S5)

S2: + Objects (Context Utilization)

You are a LaTeX expert specializing in scientific diagrams.

Visual Perception of This Diagram

The following is a structured analysis of the visual elements in this diagram, extracted from ground truth data. These are the primitive building blocks that compose the diagram — your generated code must faithfully reproduce all of them.

$\{\text{perception_data}\}$

Instructions

- The perception data above describes the fundamental visual primitives: their types, colors, text content, and spatial positions.
- Treat these as the complete inventory of what the diagram contains. Every element listed must be present in your output.
- Named colors (e.g., red, blue, gray) are standard color names.
- Output the COMPLETE code inside a single “`latex` code block. Start from `\documentclass` and end with `\end{document}`.

S3: + TikZ Search Tool (Tool Use)

You are a LaTeX expert with access to a TikZ documentation tool.

Available Tool

- **SearchLaTeXKnowledgeBase**: Search TikZ/PGF/pgfplots documentation for syntax and examples.

Workflow

1. Observe the diagram image carefully
2. If you need specific TikZ syntax or package usage, use the **SearchLaTeXKnowledgeBase** tool
3. Generate the COMPLETE LaTeX code inside a single “`latex` code block

4. Start from `\documentclass` and end with `\end{document}`

S4: + Model Generated Objects (State Management)

You are a LaTeX expert. Before writing code, analyze the image systematically.

Step 1 — Perception (inside `<perception>` tags):

1. Element types
2. Text content
3. Colors
4. Spatial layout

Step 2 — Code Generation: Based on your perception, output the COMPLETE LaTeX code inside a single ““`latex` code block. Start from `\documentclass` and end with `\end{document}`.

S5: + Objects & TikZ Search Tool (Planning)

You are a LaTeX expert with access to a TikZ documentation tool.

Visual Perception of This Diagram

The following is a structured analysis of the visual elements in this diagram, extracted from ground truth data. These are the primitive building blocks that compose the diagram — your generated code must faithfully reproduce all of them.

{perception_data}

Available Tool

- SearchLaTeXKnowledgeBase: Search TikZ/PGF/pgfplots documentation for syntax and examples.

Instructions

- The perception data above describes the fundamental visual primitives of this diagram. Every element listed must be present in your output.
- Named colors (e.g., red, blue, gray) are standard color names.
- Use the SearchLaTeXKnowledgeBase tool when you need specific syntax or package usage.
- Output the COMPLETE code inside a single ““`latex` code block. Start from `\documentclass` and end with `\end{document}`.

E.2.2 Diagram-to-Code Editing (S7–S11)

S7: + Objects (Context Utilization)

You are a LaTeX expert.

Visual Perception of This Diagram

The following is a structured analysis of the visual elements in this diagram, extracted from ground truth data. These are the primitive building blocks that compose the diagram.

{perception_data}

Instructions

- The perception data above describes the fundamental visual primitives: their types, colors, text content, and spatial positions.
- Use these primitives to understand the diagram’s structure, then apply the user’s modification instruction.
- Output the final modified code inside a single ““`latex` code block.
- Start from `\documentclass` and end with `\end{document}`.

S8: + TikZ Search Tool (Tool Use)

You are a LaTeX expert with access to a TikZ documentation tool.

Available Tool

- SearchLaTeXKnowledgeBase: Search TikZ/PGF/pgfplots documentation for syntax and examples.

Workflow

1. Observe the diagram image carefully
2. If you need specific TikZ syntax or package usage for the edit task, use the SearchLaTeXKnowledgeBase tool
3. Apply the user’s modification instruction
4. Output the final modified code inside a single ““`latex` code block
5. Start from `\documentclass` and end with `\end{document}`

S9: + Required TikZ Codes (State Management)

You are a LaTeX expert. Complete this task in two steps within a single response.

Step 1 — Reconstruct Original Code: Look at the diagram image and generate complete LaTeX code for the original diagram. Output it inside `<original_reconstruction>` tag.

Step 2 — Apply Modification: Based on Step 1, apply the user instruction. Output the final modified code inside a single ““`latex` code block.

Rules:

- The ““`latex` code block is the final answer.

- It must start from `\documentclass` and end with `\end{document}`.

S10: + Optional TikZ Codes (Planning)

You are a LaTeX expert. You may optionally reconstruct the code before editing.

If you find it helpful, you can first reconstruct the original diagram code inside `<original_reconstruction>` tags. This is entirely optional — skip it if you can apply the edit directly.

Then, apply the user's modification instruction. Output the final modified code inside a single `““latex` code block.

Rules:

- The `““latex` code block is the final answer.
- It must start from `\documentclass` and end with `\end{document}`.

S11: + Optional Codes & TikZ Search Tool (Planning)

You are a LaTeX expert with access to a TikZ documentation tool.

Available Tool

- **SearchLaTeXKnowledgeBase:** Search TikZ/PGF/pgfplots documentation for syntax and examples.

Workflow

You may optionally reconstruct the original diagram code inside `<original_reconstruction>` tags before editing. This is entirely optional — skip it if you can apply the edit directly.

If you need specific TikZ syntax or package usage, use the **SearchLaTeXKnowledgeBase** tool.

Then, apply the user's modification instruction. Output the final modified code inside a single `““latex` code block.

Rules:

- The `““latex` code block is the final answer.
- It must start from `\documentclass` and end with `\end{document}`.

E.2.3 Diagram Question Answering (S13–S16)

S13: + Objects (Context Utilization)

You are a scientific diagram analysis expert.

Visual Perception of This Diagram

The following is a structured analysis of the visual elements in this diagram, extracted from ground truth data. These are the primitive building blocks that compose the diagram.

`{perception_data}`

Instructions

- The perception data above describes the fundamental visual primitives: their types, colors, text content, and spatial positions.
- Use these primitives along with the image to answer the question.
- Follow the answer format specified in the question.

S14: + Required TikZ Codes (State Management)

You are a scientific diagram analysis expert.

Step 1 — Code Reconstruction: Generate the complete LaTeX code for this diagram. Output it inside `<internal_code_representation>` tag.

Step 2 — Answer the Question: Using both image and code, answer the question. Follow the answer format specified by the question. Output final answer after `</internal_code_representation>`.

S15: + Optional TikZ Codes (Planning)

You are a scientific diagram analysis expert.

If you find it helpful, you can first reconstruct the diagram's LaTeX code inside `<internal_code_representation>` tags to aid your analysis. This is entirely optional — skip it if you can answer directly from the image.

Then, answer the question based on the image (and your code reconstruction if you made one). Follow the answer format specified by the question. Output your final answer after any code reconstruction tags.

S16: + Optional Codes & TikZ Search Tool (Planning)

You are a scientific diagram analysis expert with access to a TikZ documentation tool.

Available Tool

- **SearchLaTeXKnowledgeBase:** Search TikZ/PGF/pgfplots documentation for syntax and examples.

Workflow

If you find it helpful, you can first reconstruct the diagram's LaTeX code inside `<internal_code_representation>` tags. This is entirely optional.

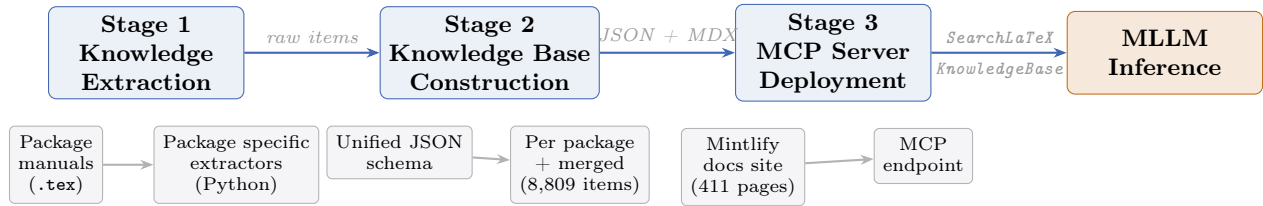


Figure F.1 Three stage construction pipeline for the MCP based TikZ documentation server. L^AT_EX source files from 19 package manuals are parsed into a unified JSON knowledge base and then deployed as a Mintlify hosted MCP server. During inference, models query the `SearchLaTeXKnowledgeBase` tool on demand.

Table F.1 Extraction strategies for the six core TikZ packages aligned with Diagram-MMU’s diagram domains.

Package	Domain(s)	#Files	Doc. Format	Extraction Strategy
tikz-pgf	Graph, Geom., 3D	~150	codeexample	Recursive scan of PGF manual tree
pgfplots	Charts, 3D	~80	codeexample	Recursive file traversal with axis detection
circuitikz	Circuit	4	LT _X example	Component catalog + showexpl examples
tkz-euclide	Planar Geom.	32	tkzexample	Multi file orchestration across submodules
chemfig	Chemistry	5	\example macro	Custom delimiter parsing
tikz-network	Graph Struct.	1	docspec/lstlisting	Command and option extraction

Table F.2 Example `SearchLaTeXKnowledgeBase` queries for benchmark domains.

Domain	Example Query	Returned Types
Charts	"addplot bar chart stacked"	Executable examples, command specs
Graph Struct.	"tikz node positioning arrow styles"	Executable examples, command specs
Planar Geom.	"tkz-euclide angle bisector mark right angle"	Executable examples, command specs
Circuit	"circuitikz resistor capacitor parallel"	Component definitions, examples
3D Shapes	"pgfplots 3d surface plot axis options"	Executable examples, command specs
Chemistry	"chemfig bond angle double bond ring"	Executable examples, key value options

If you need to understand specific TikZ syntax or package conventions, use the `SearchLaTeXKnowledgeBase` tool. Then, answer the question based on the image (and your analysis). Follow the answer format specified by the question. Output your final answer after any code reconstruction tags.

F MCP-Based TikZ Documentation Server

F.1 Server Architecture and Implementation

As described in §3.5, we build the TikZ search tool as an MCP [27] server to give models on demand access to curated TikZ documentation, avoiding the noise of web search and the inefficiency of loading full manuals.

Pipeline overview. The server is built in three stages: knowledge extraction from official L^AT_EX source files, unified knowledge base construction, and Mintlify based MCP deployment, as shown in Figure F.1.

Stage 1: Knowledge extraction. We extract structured knowledge from the official L^AT_EX source files of each TikZ package manual. Since packages use different documentation formats, we implement package specific extractors to parse examples, commands, options, and component catalogs. In total, the framework processes about 270 source files from 19 packages using seven Python scripts. Table F.1 summarizes the extraction strategy for the six core packages.

Stage 2: Knowledge base construction. All extracted items are normalized into a unified JSON schema. Each item stores its type, package, name, description, relevant code or syntax, and source file. The merged knowledge base contains 8,809 items in total, mainly executable examples and command specifications.

Stage 3: MCP server deployment. The structured knowledge base is converted into 411 Mintlify documentation pages and exposed through an MCP endpoint. The server provides a single tool, `SearchLaTeXKnowledgeBase`, which models can call during inference to retrieve relevant documentation fragments.

Query interface. The tool accepts a natural language query and optional filters, and returns ranked results such as code examples, command syntax, and usage notes. It supports keyword search, exact phrase search, command lookup, and Boolean operators. Table F.2 shows representative queries across benchmark domains.

Table F.3 Indexed TikZ package documentation. Only the most important statistics are shown.

Package	Benchmark Domain(s)	Total	Exec.	Cmd.	%
Core Packages					
tikz-pgf	Graph, Geom., 3D	3,696	2,872	808	42.0
pgfplots	Charts, 3D	1,472	1,294	167	16.7
circuitikz	Circuit	816	501	–	9.3
tkz-euclide	Planar Geom.	467	357	110	5.3
chemfig	Chemistry	290	248	–	3.3
tikz-network	Graph Struct.	71	53	18	0.8
<i>Core subtotal</i>		<i>6,812</i>	<i>5,325</i>	<i>1,103</i>	<i>77.3</i>
Extended Packages					
xspace	All	1,107	107	1,000	12.6
pst-solides3d	3D Shapes	262	262	–	3.0
fullpage	All	193	7	186	2.2
amscd	Graph Struct.	140	140	–	1.6
tkz-base	Planar Geom.	97	97	–	1.1
tkz-graph	Graph Struct.	92	92	–	1.0
Supplemental Packages					
tikz-cd	Graph Struct.	31	31	–	0.4
comment	All	25	25	–	0.3
tikz-3dplot	3D Shapes	21	21	–	0.2
tikz-qtree	Graph Struct.	15	15	–	0.2
soul	All	8	4	4	0.1
forest	Graph Struct.	5	5	–	0.1
tcolorbox	All	1	1	–	0.0
Total		8,809	6,132	2,293	100

Integration with MLLMs. During agentic evaluation, models receive the `SearchLaTeXKnowledgeBase` tool description in the system prompt and may invoke it as needed. The MCP design is model agnostic, so any MCP compatible client can connect to the same server without model specific tool implementations.

F.2 Indexed Documentation List

The knowledge base indexes official documentation from 19 L^AT_EX packages. The six core packages are directly aligned with the six diagram domains in Diagram-MMU. The remaining packages either map to one of these six domains or provide domain agnostic support that can be used across all benchmark categories. Table F.3 summarizes the indexed packages. For utility packages that are not tied to a single diagram type, we label the benchmark domain as *All* rather than leaving the field empty.

Core packages. The six core packages directly correspond to the six diagram domains in Diagram-MMU and contribute 77.3% of all indexed items. Among them, `tikz-pgf` and `pgfplots` provide the largest share of documentation, while `circuitikz`, `tkz-euclide`, `chemfig`, and `tikz-network` cover circuit, planar geometry, chemistry, and graph structure respectively.

Extended and supplemental packages. Among the remaining packages, several can still be mapped to benchmark domains. `pst-solides3d` and `tikz-3dplot` support 3D Shapes, while `amscd`, `tkz-graph`, `tikz-cd`, `tikz-qtree`, and `forest` support Graph Struct. `tkz-base` is mapped to Planar Geom. The rest, including `xspace`, `fullpage`, `comment`, `soul`, and `tcolorbox`, are domain agnostic utility packages, so we label them as *All*.

Documentation sources. All items are extracted from official package documentation only. We do not include third party tutorials, forum posts, or community generated content, so the knowledge base remains



Figure G.1 (Updated) $F1_{avg}$ heatmap per diagram type: left panel for D2C-P, middle for preserve-only D2C-E, and right for edit-only D2C-E. Color encodes score magnitude (red→white→green); the dashed line separates closed-source (top) from open-source (bottom) models. 3D shapes is the most challenging domain across all models, while chemistry is relatively the easiest.

authoritative and consistent.

G Additional Experimental Results

G.1 Foundational Results Per Diagram Type

We report Diagram-to-Code Parsing (D2C-P) results for 12 models and Diagram-to-Code Editing (D2C-E) results for 11 models, excluding TikZero+10B as it was not trained on editing samples. Figure G.1 shows $F1_{avg}$ scores over each diagram domain: left panel for D2C-P, middle for preserve-only D2C-E, and right for edit-only D2C-E. The heatmap uses a diverging color scale from red (low) through white to green (high). Figure G.2 reports DQA accuracy across 6 diagram domains.

For diagram-to-code parsing and editing, all models struggle on 3D Shapes: D2C-P scores drop as low as 11.2% (TikZero+10B), and even the best model stays below 40%, indicating that current MLLMs fail to perceive three-dimensional spatial relationships and translate them into TikZ code. Most models perform well on chemistry diagrams, as the ChemFig package encodes molecular topology explicitly through bond-chain syntax. Graph diagrams are the relatively easiest editing domain for current MLLMs, likely because models easily ground the editing instructions into node-edge topology. Closer inspection of the D2C-E heatmaps shows that preserve-only scores are higher than edit-only scores. For example, in charts, edit-only scores fall 10–20 points below preserve-only, indicating that current models retain untouched elements well but struggle to apply the instructed modifications. Moreover, the gap between open-source and closed-source models is larger on chemistry and circuit diagrams, especially for D2C-P.

G.2 F1 Scores Per Object

Figure 6 shows F1 score breakdowns for six representative models across three radars: D2C-P (left), D2C-E preserve-only (center), and D2C-E edit-only (right). Each radar plots five axes: $F1_{avg}$, $F1_{type}$, $F1_{bbox}$, $F1_{color}$, and $F1_{text}$, with axis scales fitted to each radar’s value range.

A consistent finding is that $F1_{bbox}$ is dramatically lower than other object attributes. In D2C-P, all model polygons collapse toward the center on the $F1_{bbox}$ axis. This asymmetry persists in D2C-E preserve-only evaluation and becomes even more extreme in edit-only evaluation, confirming that spatial positioning remains the primary bottleneck across both parsing and editing tasks. Comparing the three panels shows a clear degradation: preserve-only scores are substantially higher than D2C-P on most axes, especially text and color where models benefit from retaining existing code, while edit-only scores drop sharply. Among the six models, Gemini-3.0 Pro achieves the largest polygon area across all three panels, while Qwen3-VL-8B shows the most compact polygons.

Table G.1 reports the per-object F1 scores on D2C-P, code-level (CrystalBLEU) and image-level (SSIM, CLIP, FID, LPIPS) metrics, complementing the aggregated $F1_{avg}$ in Table 5. $F1_{bbox}$ is the lowest across all models, confirming the spatial-positioning bottleneck identified in Figure 6.

Table G.1 Three-level metric results of 12 models on D2C-P. ↓ means lower is better. **Bold** is best in class; underline is second best.

Model	Object-Level					Code-Level	Image-Level			
	F1 _{type}	F1 _{text}	F1 _{color}	F1 _{bbox}	F1 _{avg}	CBLEU	SSIM	CLIP	FID↓	LPIPS↓
Closed-Source										
Gemini-3.1 Pro	59.9	61.7	61.2	16.9	49.94	32.88	54.01	70.64	7.66	43.15
Gemini-3.0 Pro	66.8	<u>69.2</u>	69.9	<u>12.7</u>	54.64	<u>32.76</u>	<u>62.13</u>	82.75	<u>8.88</u>	35.14
Gemini-3.0 Flash	<u>63.3</u>	66.1	65.3	10.1	51.19	30.67	60.45	81.63	9.21	<u>36.59</u>
GPT-5.2	62.7	68.4	<u>66.2</u>	8.0	51.35	28.81	62.66	88.29	12.74	37.53
Claude-4.6 Opus	62.7	71.1	66.0	9.2	<u>52.23</u>	31.27	61.11	<u>86.36</u>	13.79	42.63
Seed-2.0 Pro	59.8	62.4	60.3	7.8	47.57	32.56	60.14	84.93	14.66	44.30
Open-Source										
Qwen3.5-397B-A17B	61.8	66.2	63.6	14.0	51.38	32.92	61.86	85.83	<u>11.06</u>	<u>39.28</u>
Qwen3-VL-235B-A22B	<u>66.2</u>	<u>70.9</u>	<u>66.8</u>	<u>16.6</u>	<u>55.11</u>	37.92	<u>61.87</u>	<u>87.93</u>	12.92	44.86
Kimi-K2.5	69.0	73.3	70.5	17.1	57.48	<u>36.13</u>	64.35	90.26	10.03	37.95
Qwen3-VL-8B	52.7	58.4	54.5	13.0	44.66	33.31	54.78	78.84	15.90	55.52
InternVL3-38B	38.1	41.2	38.7	7.9	31.47	30.91	50.75	71.28	19.67	62.41
TikZero+ 10B	25.5	6.8	26.4	3.1	15.43	17.19	33.99	52.59	24.55	78.41

Table G.2 Three-level metric results of 11 models on D2C-E with preserve-only (p) and edit-only (e) splits. ↓ means lower is better. **Bold** is best in class; underline is second best.

Model	Preserve-Only					Edit-Only					CBLEU (p/e)		Image-Level			
	F1 _{type}	F1 _{text}	F1 _{color}	F1 _{bbox}	F1 _{avg}	F1 _{type}	F1 _{text}	F1 _{color}	F1 _{bbox}	F1 _{avg}	p	e	SSIM	CLIP	FID↓	LPIPS↓
<i>Closed-Source</i>																
Gemini-3.1 Pro	70.4	70.7	69.1	32.2	60.60	43.6	60.4	52.1	7.3	40.84	42.02	2.98	57.63	75.32	4.62	<u>41.80</u>
Gemini-3.0 Pro	77.4	78.4	75.7	<u>28.9</u>	65.12	<u>39.9</u>	64.9	<u>51.2</u>	<u>5.0</u>	<u>40.24</u>	44.77	<u>2.34</u>	61.21	<u>83.55</u>	5.84	39.29
Gemini-3.0 Flash	71.2	72.7	69.0	24.2	59.29	37.1	58.7	48.8	4.3	37.24	42.21	2.17	57.17	77.71	<u>5.54</u>	43.13
GPT-5.2	66.8	70.3	64.5	20.8	55.59	28.7	53.3	40.3	2.3	31.15	38.75	1.15	54.71	75.94	8.61	47.77
Claude-4.6 Opus	<u>75.1</u>	<u>78.2</u>	<u>72.7</u>	24.3	<u>62.56</u>	25.4	<u>60.5</u>	39.0	3.0	31.95	<u>43.59</u>	1.42	<u>59.54</u>	84.02	8.75	46.14
Seed-2.0 Pro	64.9	66.2	61.3	19.8	53.03	28.7	50.0	39.5	2.3	30.14	37.49	1.41	51.48	72.24	10.15	54.21
<i>Open-Source</i>																
Qwen3.5-397B-A17B	<u>74.0</u>	<u>75.4</u>	71.1	30.5	62.75	<u>35.2</u>	60.1	45.3	4.7	<u>36.35</u>	43.66	<u>1.91</u>	59.14	<u>81.72</u>	<u>7.08</u>	44.18
Qwen3-VL-235B-A22B	75.3	77.0	70.3	<u>32.6</u>	63.79	30.0	57.9	40.9	<u>4.8</u>	33.38	<u>43.05</u>	1.85	58.06	82.08	8.63	50.03
Kimi-K2.5	73.8	74.9	<u>70.5</u>	33.3	<u>63.12</u>	35.7	60.4	<u>44.6</u>	5.4	36.52	42.11	2.02	<u>58.77</u>	80.98	5.89	<u>45.11</u>
Qwen3-VL-8B	24.9	26.3	23.5	10.4	21.26	8.6	17.2	12.8	1.2	9.95	16.29	0.51	22.07	31.81	18.16	82.64
InternVL3-38B	53.6	55.3	49.1	20.9	44.72	20.5	38.5	29.4	2.3	22.69	33.13	1.29	48.37	67.54	13.39	65.13

Table G.2 reports the per-object F1 scores on D2C-E with preserve-only and edit-only splits, alongside code-level (CrystalBLEU) and image-level (SSIM, CLIP, FID, LPIPS) metrics. For preserve-only elements, models achieve moderate scores on type, text, and color, but F1_{bbox} remains the lowest. For edit-only elements, all scores drop substantially, e.g., F1_{type} shows the largest absolute gap, indicating that models struggle to instantiate correct object types for required editing elements.

G.3 DQA Analysis Per Diagram Type

Table G.3 reports per-domain DQA accuracy across all 11 evaluated models (TikZero+ 10B is excluded as it is not trained on question answering). Fig. G.2 provides a complementary visual comparison using grouped bar charts.

Graph structures is the easiest domain for most models, with top-performing models exceeding 90% accuracy (e.g., Gemini-3.1 Pro at 91.34%). This is likely because graph-related questions often involve counting nodes or identifying connectivity patterns, which are relatively straightforward visual reasoning tasks.

3D Shapes is consistently the hardest DQA domain, with even the best models scoring below 69%. The difficulty stems from the need to reason about three-dimensional spatial relationships, such as viewing angles and surface properties, from rendered diagrams.

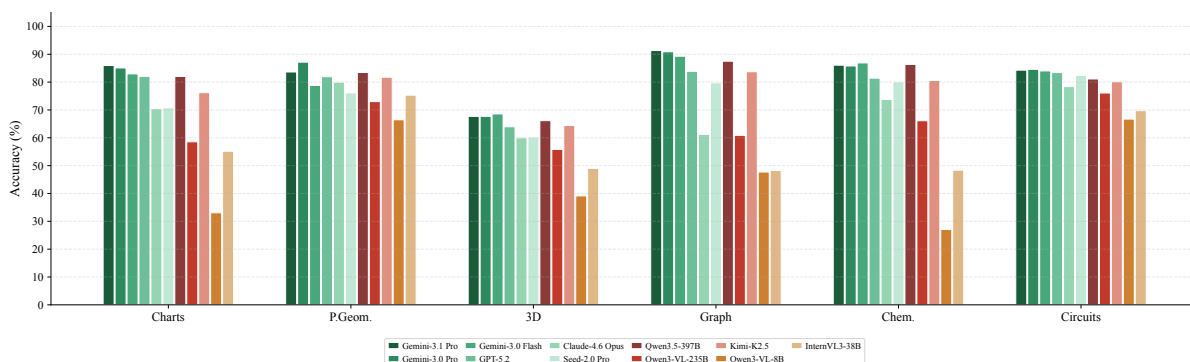
Among open-source models, Qwen3-VL-8B shows particularly uneven domain performance: it achieves 66.71% on Circuits but only 27.05% on chemistry, suggesting that smaller models lack generalization across diverse diagram types. Conversely, Qwen3.5-397B-A17B maintains relatively balanced accuracy (66–88%) across all domains, approaching closed-source performance.

G.4 Agentic Tool-Call Statistics

Table G.4 reports tool-calling statistics in the agentic settings (S3/S8/S16 in Table 4).

Table G.3 DQA accuracy (%) per diagram type. **Bold** is best in class; underline is second.

Model	Charts	P. Geom.	3D	Graph	Chem.	Circuits	Avg.
<i>Closed-Source</i>							
Gemini-3.1 Pro	85.95	83.63	67.69	91.34	86.07	84.29	86.29
Gemini-3.0 Pro	<u>85.08</u>	87.16	67.69	<u>90.88</u>	85.79	84.58	86.46
Gemini-3.0 Flash	82.95	78.80	68.57	89.29	86.89	84.01	84.07
GPT-5.2	82.08	81.93	63.96	83.87	81.42	83.43	81.67
Claude-4.6 Opus	70.42	79.96	60.00	61.22	73.77	78.39	68.75
Seed-2.0 Pro	70.75	76.12	60.22	79.73	80.05	82.42	75.90
<i>Open-Source</i>							
Qwen3.5-397B-A17B	82.03	83.45	66.15	87.50	86.34	81.12	83.42
Qwen3-VL-235B-A22B	58.55	72.99	55.82	60.88	66.12	76.08	63.60
Kimi-K2.5	<u>76.20</u>	<u>81.75</u>	<u>64.40</u>	<u>83.73</u>	<u>80.60</u>	<u>80.12</u>	<u>79.74</u>
Qwen3-VL-8B	33.06	66.46	39.12	47.70	27.05	66.71	47.12
InternVL3-38B	55.17	75.31	49.01	48.23	48.36	69.74	56.39

**Figure G.2** (Updated) DQA accuracy (%) per diagram type (an alternative visualization of Table G.3). Bars distinguish closed-source from open-source models.

Adoption. All six models invoke the tool to some degree, but adoption varies widely (0.2%–96.5%) across tasks and models. On D2C-P and D2C-E, Claude-4.6 Opus adopts tools most aggressively (93.7%/96.5%), followed by Gemini-3.0 Pro (45.0%/48.0%) and Qwen3-VL-8B (38.7%/43.8%), while Seed-2.0 Pro is most conservative (8.3%/10.0%). DQA triggers much lower tool use across all models, since question-answering rarely requires code-level documentation lookup. Within D2C-E, domains requiring specialized packages, e.g., circuit (`circuitikz`, up to 80%) and chemistry (`chemfig`, up to 65%), elicit the highest adoption, while graph diagrams drawn with basic `tikz` commands show the lowest (1–13% for most models).

Intensity. Claude-4.6 Opus averages $\bar{n}=2.83$ calls per question. We inspect its calling pattern: it first queries package syntax, then specific commands, final usage examples. This structured pattern correlates with its consistent gains under tool use. Gemini-3.1 Pro shows the highest average calls among tool-invoking questions ($\bar{n}_+=2.37$ in D2C-E) despite low adoption (15%), indicating excessive querying loops, e.g., repeated searches without synthesizing results, leading to context rot at the token limit. Qwen3-VL-8B shows the opposite: moderate adoption (43.8%) but low $\bar{n}_+=1.12$, typically issuing only a single query per question.

Response and execution rates. Table G.5 compares two rates between direct (S1/S6) and tool-use agentic settings (S3/S8) for D2C-P/D2C-E: (1) Response rate (%): the percentage of questions where the model outputs a complete response; failures occur when the model exceeds the context length or enters tool-calling loops without producing final code; (2) Execution rate (%): the percentage of generated code that compiles successfully; failures occur when the output code contains syntax errors. Gemini-3.1 Pro suffers the most: tool access reduces the response rate by 25.0% (D2C-P) and 23.8% (D2C-E) due to the querying loops described above. Claude-4.6 Opus benefits most: tool access improves execution rate by +4.2% on D2C-E. Other models show minimal changes.

Table G.4 Tool-calling statistics in tool-use agentic settings (**S3/S8** = +Tool; **S16** = +Tool&TikZ Codes). N is the number of questions; Adopt. is the percentage of questions where the model invokes the tool at least once (≥ 1) out of all N questions; $\bar{n}$ is the total call count divided by N ; $\bar{n}_+$ is the total call count divided by the number of questions where the model called the tool at least once.

Model	D2C-P (S3 , $N=300$)			D2C-E (S8 , $N=600$)			DQA (S16 , $N=600$)		
	Adopt. (%)	$\bar{n}$	$\bar{n}_+$	Adopt. (%)	$\bar{n}$	$\bar{n}_+$	Adopt. (%)	$\bar{n}$	$\bar{n}_+$
Claude-4.6 Opus	93.7%	2.70	2.88	96.5%	2.83	2.93	2.7%	0.05	1.75
Gemini-3.0 Pro	45.0%	0.75	1.67	48.0%	0.79	1.65	14.0%	0.22	1.56
Qwen3-VL-8B	38.7%	0.43	1.10	43.8%	0.49	1.12	0.2%	0.00	1.00
GPT-5.2	26.3%	0.49	1.87	23.3%	0.43	1.84	5.7%	0.08	1.47
Gemini-3.1 Pro	18.7%	0.44	2.34	15.0%	0.35	2.37	1.2%	0.02	1.43
Seed-2.0 Pro	8.3%	0.09	1.04	10.0%	0.10	1.02	0.8%	0.01	1.00

Table G.5 Response and execution rates (%) in direct (**S1/S6**) vs. tool-use settings (**S3/S8**). Δ : change from direct to +Tool ($\uparrow$ gain, $\downarrow$ drop).

Model	D2C-P						D2C-E					
	Resp. Rate.			Exec. Rate.			Resp. Rate.			Exec. Rate.		
	S1	S3	Δ	S1	S3	Δ	S6	S8	Δ	S6	S8	Δ
Claude-4.6 Opus	100.0	100.0	0.0	91.3	93.3	+2.0	100.0	99.7	-0.3	89.2	93.3	+4.2
Seed-2.0 Pro	100.0	100.0	0.0	89.7	92.3	+2.7	99.7	100.0	+0.3	90.3	89.2	-1.2
GPT-5.2	100.0	97.7	-2.3	95.3	91.7	-3.7	100.0	98.2	-1.8	91.3	92.7	+1.3
Gemini-3.0 Pro	100.0	99.7	-0.3	86.3	83.7	-2.7	100.0	99.3	-0.7	90.0	86.8	-3.2
Gemini-3.1 Pro	100.0	75.0	-25.0	73.3	60.7	-12.7	100.0	76.2	-23.8	79.7	63.2	-16.5
Qwen3-VL-8B	99.7	100.0	+0.3	88.0	87.3	-0.7	99.3	100.0	+0.7	86.8	86.3	-0.5

H Limitations and Future Work

Agentic evaluation. As acknowledged (§3.5), we do not claim Diagram-MMU evaluates the full spectrum of agentic ability in the vibe writing framework. Vibe writing platforms (e.g., Prism [12]) are becoming increasingly popular, as any researcher can prompt models in natural language to generate academic papers directly in $\text{\LaTeX}$ format. Common needs include inserting a diagram into the manuscript as compilable $\text{\LaTeX}$ TikZ code, applying specific edits based on the paper content, or describing the domain-specific meaning of diagram symbols. Our evaluation settings target such multimodal aspects of vibe writing through diagram-to-code and diagram understanding tasks.

However, the vibe writing workspace involves broader capabilities such as text drafting, citation management, and layout formatting, which remain outside our scope and would require designing agentic evaluation settings for purely textual abilities. Moreover, broader agentic capabilities such as long-horizon task decomposition, multi-tool orchestration across heterogeneous environments, self-correction via execution feedback, and interactive collaboration with human users are also not covered by the current benchmark.

Code representation. Diagram-MMU adopts $\text{\LaTeX}$ TikZ as the code representation, whose output compiles inline within $\text{\LaTeX}$ authoring environments. But, in practice, researchers also create scientific diagrams with Python libraries (e.g., Matplotlib, Seaborn), SVG/HTML tools, or other $\text{\LaTeX}$ drawing packages (e.g., `pstricks`, `xy-pic`), which are currently not covered by the evaluation settings of Diagram-MMU.

Future directions. In future work, we plan to extend agentic settings to evaluate: (1) broader textual and layout capabilities in scientific writing, such as text-to-TikZ generation, table construction from data, and mathematical derivation formatting; and (2) richer agentic workflows, including execution feedback loops, iterative human-in-the-loop refinement, and multi-tool orchestration, that reflect the real-world demands of vibe writing. For code representation, cross-language conversion pipelines that translate Python or SVG diagrams into TikZ would expand coverage while preserving inline $\text{\LaTeX}$ compilation, and supporting diagram import via image upload would allow vibe writing workspaces to ingest figures. Alternatively, extending evaluation to additional $\text{\LaTeX}$ drawing packages beyond TikZ (e.g., `pstricks`, `xy-pic`) would broaden coverage within the $\text{\LaTeX}$ ecosystem directly. For diagram understanding and reasoning, we plan to move toward expert-level questions, and to incorporate richer context such as figure captions from the surrounding paper, enabling evaluation of diagram comprehension grounded in the full manuscript.

We hope future versions of Diagram-MMU can provide more comprehensive insights and guide targeted improvements by pinpointing where models fail: whether in foundational abilities (perception, coding, knowledge, reasoning) or agentic abilities (context utilization, tool use, state management, planning). We do not foresee negative societal consequences; the benchmark is constructed from publicly available TikZ documentation and contains no private or sensitive data.

I Acknowledgement

We thank all 13 annotators for their contributions to data quality assurance; together, they verified over 2,000 scientific diagrams through cross-validation across six domains. Three annotators are co-authors of this paper. Among the remaining ten contributors, one is a research scientist at NetMind.ai, eight are PhD students at Nanjing University of Science and Technology (NJUST), and one is a PhD student at Westlake University. We gratefully acknowledge these ten collaborators and detail below the number of annotated samples and the domains each covered:

- Xinyu Miao from NJUST — 419 cases, covering 3D Shape, Charts, Chemical Expressions, Graph Structures, Planar Geometry
- Yiyao Gao from NJUST — 331 cases, covering 3D Shape, Charts, Graph Structures, Planar Geometry
- Ruolin Wang from NJUST — 271 cases, covering 3D Shape, Charts, Graph Structures, Planar Geometry
- Xinyao Hu from NetMind.ai — 208 cases, covering 3D Shape, Charts, Graph Structures, Planar Geometry
- Jinhong Yang from NJUST — 200 cases, covering Graph Structures
- Zongxin Zhu from NJUST — 194 cases, covering Charts, Graph Structures, Planar Geometry
- Yu Wang from NJUST — 181 cases, covering Graph Structures, Planar Geometry
- Xinyu Zhang from NJUST — 170 cases, covering Charts, Circuit Diagrams, Graph Structures
- Xueji Fang from Westlake University — 107 cases, covering 3D Shape, Charts, Graph Structures, Planar Geometry
- Wei Tang from NJUST — 11 cases, covering 3D Shape, Charts