mxSRBench: Towards Spatial Understanding and Diagrammatic Reasoning in `mxGraph` for Large Language Models

Anonymous ACL submission

Abstract

While LLMs have advanced code synthesis and semantic reasoning, their ability to bridge the spatial–semantic gap in diagrams remains poorly understood. Diagrams demand satisfaction of semantic topology and metric geometry, yet existing benchmarks, focused on VQA or pixel generation, fail to isolate the resulting symbolic reasoning failures. We introduce mxSRBench, a benchmark of 4,081 instruction–XML pairs built on `mxGraph`, a high-fidelity symbolic language that explicitly encodes spatial logic. mxSRBench offers three tiers: **T1** validates XML and structural renderability; **T2** probes spatial reasoning; and **T3** evaluates professional diagram synthesis across 11 types. For T2, we establish an **Evaluation Protocol** covering three core capabilities: **(1) Understanding** (interpreting spatial logic from XML), **(2) Generation** (grounding textual spatial constraints into valid coordinate XML), and **(3) Multihop Reasoning** (compositional multi-step spatial inference). Systematic benchmarking of LLMs/VLMs reveals a spatial–semantic gap: top models achieve up to 91.7% in understanding but degrade sharply on constrained generation and multihop reasoning, confirming they are sophisticated logicians yet poor drafters. mxSRBench provides a quantitative, diagnostic foundation for neuro-symbolic and tool-augmented diagram generation.

1 Introduction

Recent rapid progress in LLMs (Achiam et al., 2023; Touvron et al., 2023; Guo et al., 2025)—including reasoning (Wei et al., 2022; Yao et al., 2023; Wang et al., 2022), code synthesis (Chen et al., 2021b; Nijkamp et al., 2022; Li et al., 2023a), and in-context learning (Brown et al., 2020; Dong et al., 2024; Zhou et al., 2024)—suggests their potential to generate diagram specifications from natural language. Leveraging LLMs to create editable structured diagrams—with explicit structure and editability—addresses key limits of current methods: template tools lack flexibility (Shbita et al., 2025), layout engines produce rigid results (Yang et al., 2024), and AI systems generate non-editable raster images (Zhu et al., 2026). LLM-based generation can reduce costs across office, industrial, and academic scenarios by enabling iterative editing, automatic alignment, and rapid refinement.

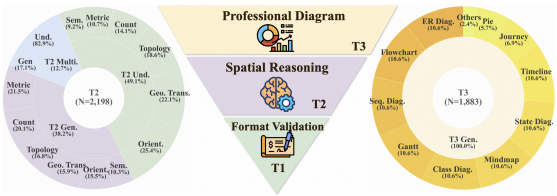


Figure 1: Distribution of mxSRBench across T2 (N=2,198) and T3 (N=1,883) tracks, spanning 6 dimensions and 11 diagram types.

However, this demands "diagrammatic spatial reasoning" from LLMs, requiring generated diagrams to simultaneously ensure: (1) **Semantic and structural integrity**: elements align with user instructions and remain logically self-consistent. (2) **Layout coherence and readability**: the model manages global aesthetics—alignment, spacing, and overlap avoidance—for visual clarity. (3) **Precise spatial constraints**: consistent relative positions (e.g., containment, proximity) are maintained under complex, multi-constraint reasoning. Evaluation thus requires outputs that are not only syntactically well-formed, but also editable, structurally faithful, and geometrically verifiable. This raises a critical question for benchmarking: ***What diagram representation can effectively encapsulate semantic hierarchy and layout coherence?***

To address this, LLMs must produce an intermediate representation as the diagram blueprint. We categorize candidate formats by their emphasis on semantic structure or spatial layout (Table 1): (1) **Semantic-centric DSLs** (e.g., Mermaid, DOT) provide robust logical structures but delegate layout to black-box renderers, limiting spatial control and

Table 1: Taxonomy of diagram generation paradigms. Unlike *Semantic-First* paradigms that sacrifice layout control, or *Layout-First* ones that lack explicit topology, `mxGraph` strikes a balance. * Native Doc refers to PPTX or Figma.

Criterion	Semantic-First (Textual)		Layout-First (Geometric)			Balanced
	DOT	DSL (Mermaid)	SVG	Diffusion	Native Doc*	`mxGraph` (Ours)
Representation	Text Spec	Logic Code	Vector Paths	Raster Image	Tool Objects	**Graph Objects**
Topological Logic	**Explicit**	**Explicit**	Implicit	None	Weak	**Explicit**
Layout Precision	Low (Auto)	Low (Auto)	**High**	Stochastic	**High**	**High**
Native Editability	Limited	Limited	No	No	Yes	**Full (Dynamic)**
Structure Verifiable	Yes	Yes (Typed)	Limited	No	Yes	**Yes**
Practical Deployment	Medium	Medium	Medium	Low	High	**High (Industrial)**
Evaluation Cost	Low	Low	Medium	High	Medium	**Low**

geometric evaluation. (2) **Layout-centric formats** (e.g., SVG, diffusion models (Zhang et al., 2023; Rombach et al., 2022), native documents like PPTX/Figma) offer high visual fidelity but either lack explicit structural metadata or impose prohibitive syntactic complexity, impeding reliable generation and automated assessment.

We bridge these paradigms with `mxGraph`. It preserves explicit semantic hierarchies while exposing verifiable geometric coordinates. Crucially, its hierarchical cell structure encodes containment and grouping through explicit parent-child relationships. For a nested flowchart where sub-process A contains three decision nodes, SVG demands complex geometric checks to verify containment; in `mxGraph`, containment is directly encoded via `parent` attributes, enabling hierarchical reasoning without coordinate computation. Likewise, ID-based edges (`source`/`target`) make topology directly queryable, whereas SVG hides connectivity in implicit path geometries. This explicit, queryable structure, together with a concise XML syntax well within current LLM capabilities, supports reliable spatial reasoning (empirically validated in Section 4) and scalable, standardized evaluation across diagramming tasks.

To quantify LLM proficiency in synthesizing editable structured diagrams, we introduce mxSRBench[1], a multi-dimensional benchmark evaluating: (1) **Basic correctness**: *syntactic validity and structural fidelity to user instructions*; (2) **Spatial reasoning**: *handling atomic and compositional spatial constraints with global consistency*; (3) **Professional adaptation**: *generating diverse, domain-specific diagrams adhering to industry conventions*. mxSRBench comprises 4,081 instruction–XML pairs and a three-tier suite (Figure 1): **T1** validates XML renderability; **T2** probes spatial reasoning across six dimensions via three subtasks—Understanding (interpreting spatial logic from given XML), Generation (grounding textual constraints into coordinates), and Multihop Reasoning (integrating multiple relations into a globally consistent layout); **T3** tests professional synthesis over 11 diagram types. We employ **Decomposed Requirements Following Ratio (DRFR)** (Qin et al., 2024) to objectively assess per-constraint spatial satisfaction, moving beyond subjective visual similarity.

Table 2 positions mxSRBench against prior spatial reasoning benchmarks. Image and video benchmarks encode spatial relations implicitly in pixels, while SVG-, program-, or logic-form benchmarks expose some structure but are often limited in task coverage, reasoning dimensions, or evaluation granularity. In contrast, mxSRBench combines explicit coordinates, topological relations, understanding and generation tasks, six spatial dimensions, and per-constraint diagnostics, while covering 11 professional diagram categories.

Our contributions are as follows:

- **Benchmark and Data:** We introduce mxSRBench, a 4,081-sample `mxGraph` benchmark for diagrammatic spatial reasoning with explicit symbolic and geometric structure.
- **Tiered Evaluation and Diagnostics:** We design a three-tier suite covering format validation, spatial reasoning, and professional synthesis, with T2 decomposed into Understanding, Generation, and Multihop modes.
- **Constraint-Centric Metrics and Findings:** We use **DRFR** for objective constraint satisfaction and benchmark diverse LLMs/MLLMs, revealing weaknesses in coordinate grounding and topological consistency.

2 Related Work

Our work builds on spatial reasoning, diagram generation, and controllable generation, which we discuss below.

[1]Datasets are available at [link].

Table 2: Comparison with existing spatial reasoning benchmarks. **Explicit Spatial** indicates whether the benchmark exposes structured coordinate/topological data (✓) versus implicit pixel-level representations (✗); △ denotes partial (e.g., bounding boxes only). **#Spatial Dims** counts the distinct spatial reasoning dimensions evaluated. **Decomp. Metric** indicates whether per-constraint decomposed evaluation is used (△: task-specific but not per-constraint).

Benchmark	Representation	#Samples	Explicit Spatial	Understand	Generate	#Spatial Dims	Decomp. Metric
VSI-Bench (Yang et al., 2025b)	Video	>5,000	✗	✓	✗	3	✗
SpatialSense (Yang et al., 2019)	Image + BBox	17,498	△	✓	✗	1	✗
T2I-CompBench (Huang et al., 2023)	Image	6,000	✗	✗	✓	2	✗
SVGenius (Chen et al., 2025)	SVG Code	2,377	✓	✓	✓	4	△
Geometry3K (Lu et al., 2021)	Logic Form	3,002	✓	✓	✗	2	✗
GeoQA (Chen et al., 2021a)	Program	4,998	✓	✓	✗	2	✗
mxSRBench (Ours)	`mxGraph`	**4,081**	✓	✓	✓	**6**	✓

2.1 Spatial Reasoning and Understanding

Early work in spatial reasoning established foundational concepts for topological inference using qualitative formalisms such as Region Connection Calculus (Cohn et al., 1997). Recent vision-language models(Antol et al., 2015; Yang et al., 2025b; Shiri et al., 2024) have advanced spatial understanding, with SpatialVLM (Chen et al., 2024a), SpatialRGPT (Cheng et al., 2024a), and SpatialBench (Xu et al., 2025) addressing 3D reasoning, depth integration, and systematic evaluation respectively. However, these models focus on spatial understanding rather than generation, lacking support for producing structured, editable diagram specifications.

Existing benchmarks exhibit limitations in evaluation granularity. VSI-Bench (Yang et al., 2025b) and SpatialSense (Yang et al., 2019) focus on video/image-based spatial tasks, while GSR-Bench (Rajabi and Kosecka, 2024) and What'sUp (Kamath et al., 2023) probe grounded understanding in VLMs. Text-based benchmarks including SpaRC (Rizvi et al., 2024) and StepGame (Li et al., 2024) test spatial inference without visual grounding. For geometric reasoning, GeoQA (Chen et al., 2021a), Geometry3K (Lu et al., 2021), UniGeo (Chen et al., 2022), G-LLaVA (Gao et al., 2025) evaluate problem solving but focus primarily on understanding. 3D spatial benchmarks such as SPARE3D (Han et al., 2020) and SQA3D (Ma et al., 2022) assess spatial QA from 3D scenes. For image generation, DrawBench (Saharia et al., 2022), PartiPrompts (Yu et al., 2022) assess compositional capabilities, while T2I-CompBench (Huang et al., 2023) and T2I-CompBench++ (Huang et al., 2025) specifically evaluate attribute binding, spatial relationships, and numeracy in text-to-image synthesis. Crucially, these benchmarks adopt implicit pixel-level or pure-text representations, making it difficult to diagnose whether failures stem from spatial reasoning deficits or coordinate-level grounding errors. mxSRBench addresses this gap using structured `mxGraph` XML with explicit coordinates and topological relations, enabling fine-grained error attribution across six spatial dimensions.

2.2 Structured and Controllable Generation

Layout-guided image generation has advanced through diffusion-based approaches, including GLIGEN (Li et al., 2023b), LayoutDiffusion (Zheng et al., 2023), HiCo (Cheng et al., 2024b), and BoxDiff (Xie et al., 2023) for layout control, and ControlNet (Zhang et al., 2023) with T2I-Adapter (Mou et al., 2024) for spatial conditioning. However, these methods produce raster outputs without structural metadata, limiting both automated assessment and downstream editability.

Vector graphics generation offers an alternative paradigm with explicit geometric representation. Early neural approaches including DeepSVG (Carlier et al., 2020), Im2Vec (Reddy et al., 2021), and IconShop (Wu et al., 2023) established foundations for SVG synthesis. Comprehensive benchmarks evaluate LLM capabilities: VGBench (Zou et al., 2024) assesses SVG, TikZ, and Graphviz processing, SVGenius (Chen et al., 2025) stratifies evaluation across complexity tiers, and SGP-Bench (Qiu et al., 2025) tests semantic consistency in symbolic programs. For SVG-specific evaluation, SVGEditBench (Nishina and Matsui, 2024, 2025) targets editing capabilities, Image-Text Bridging (Cai et al., 2025) probes cross-modal understanding, and SVG-Taxonomy (Xu and Wall, 2024) examines low-level visual tasks. Generation methods include StarVector (Rodriguez et al., 2025), OmniSVG (Yang et al., 2026), SVGBuilder (Chen and Pan, 2025), Chat2SVG (Wu et al., 2025), LLM4SVG (Xing et al., 2025) and DiffSketcher (Xing et al., 2023). UniSVG (Li et al., 2025) and InternSVG (Wang et al., 2025) pursue

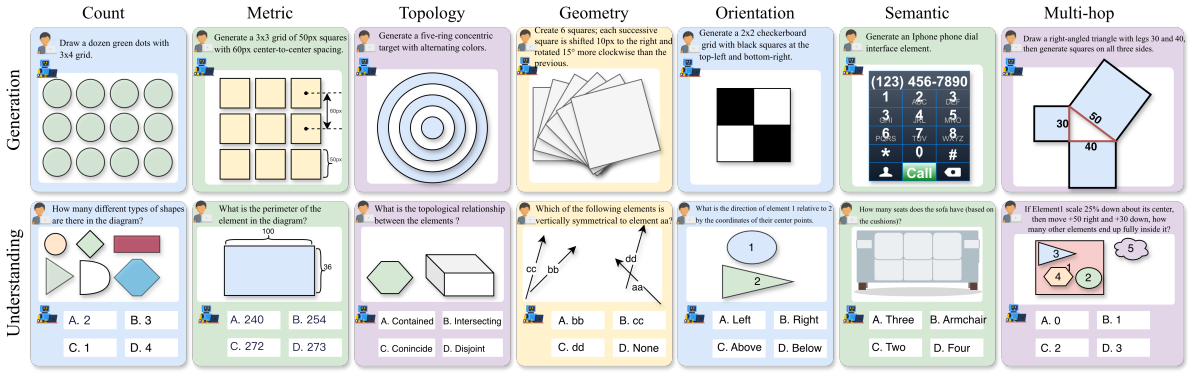


Figure 2: T2 task demos in mxSRBench. **Understanding** and **Generation** examples are shown across six spatial dimensions: Count, Metric, Topology, Geometry Transformation (Geo Trans), Orientation, and Semantic.

unified SVG understanding and generation. TextLap (Chen et al., 2024b) addresses layout planning, while StructEval (Yang et al., 2025a) evaluates structured output generation. Despite these advances, existing approaches either lack explicit topological semantics, delegate layout to black-box engines, or cover narrow task scopes. mxSRBench uniquely combines semantic hierarchies with verifiable geometric coordinates in `mxGraph` XML, enabling comprehensive spatial reasoning evaluation through the DRFR metric.

3 mxSRBench Framework

3.1 Framework Overview

3.1.1 T1: Format Validation

Format Validation assesses whether a model can produce *renderable* `mxGraph` XML. Concretely, outputs must be well-formed, schema-consistent, and loadable in draw.io. We formalize failures via a **minimum fatal error set**: the smallest set of errors that makes an XML file non-renderable. Specifically, we implement a three-pass validator as presented in Figure 3: XML physical structure validation, attribute validation for geometry and edge references, and hierarchical reference validation for parent links, self-references, and cycles.

3.1.2 T2: Spatial Reasoning Tasks

T2 evaluates whether an LLM can correctly process diagrams represented as explicit `mxGraph` XML: reading structured spatial information, writing valid editable specifications, and composing multiple spatial constraints into globally consistent solutions. **T2** targets bridging the spatial–semantic gap under coordinate-level constraints, requiring jointly handling discrete graph structure (e.g., `id`, `parent`) and continuous geometry (e.g., `x`, `y`). We evaluate T2 along three core capabilities: (1) **Understanding**—parse `mxGraph` XML to accurately infer symbolic structure and spatial/-topological relations; (2) **Generation**—translate natural-language instructions into structurally correct and geometrically faithful `mxGraph` XML; (3) **Multihop Reasoning**—assess whether models successfully solve complex cases requiring chaining multiple atomic relations and resolving long-range references while maintaining global consistency.

3.1.3 T3: Professional Diagram Generation

T3 evaluates whether an LLM can act as a reliable professional drafter: given text instruction, can it synthesize an editable `mxGraph` diagram matching domain conventions, preserving the intended semantic structure, and remaining readable under complex layout constraints? Unlike T2's atomic spatial skills, T3 targets holistic diagram synthesis where correctness depends on global composition. Our T3 test set contains 1,883 samples covering 11 professional diagram types (Table 14). Since professional diagrams involve rich, type-specific conventions that are difficult to exhaustively decompose into atomic checklists, we use a VLM-based judge that compares reference and generated renderings with type-specific rubrics and returns a similarity/quality score.

3.2 mxSRBench Dataset Construction

mxSRBench contains 4,081 instruction-XML pairs created through a "Rule Engine + LLM Rewriting" hybrid approach, achieving both spatial accuracy and linguistic diversity. We summarize data construction pipelines by task.

T2: Spatial Reasoning Data. T2 contains three complementary subsets. **(1) Understanding** synthesizes draw.io diagrams in `mxGraph` XML using

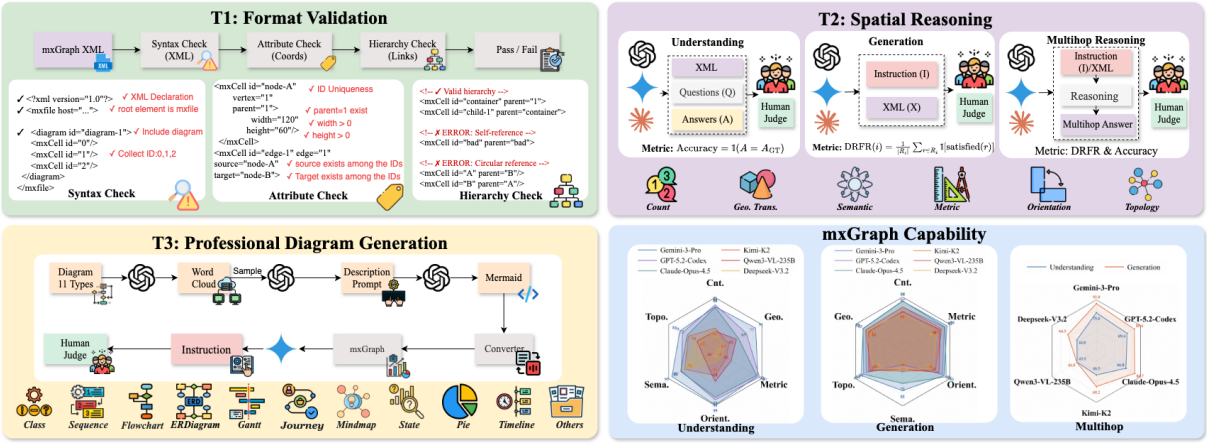


Figure 3: Overview of the mxSRBench evaluation tiers. Our benchmark spans from low-level Format Validation (**T1**) to multi-dimensional Spatial Reasoning (**T2**) and high-level Professional Diagram Synthesis (**T3**). The radar charts illustrate the performance gaps among representative models across key mxGraph capabilities.

Table 3: Performance on mxSRBench **T2** tasks. We report Understanding accuracy (%), Generation (**DRFR**), and Multihop results (Accuracy/**DRFR**). Bold entries denote the best results within closed-source and open-source models, respectively.

Model	Understanding (Accuracy)							Generation (DRFR)							Multihop	
	Count	Geo	Met	Orie	Sem	Top	Avg	Count	Geo	Met	Orie	Sem	Top	Avg	Und	Gen
Closed Source Models																
Gemini-3-Pro	88.16	**90.76**	93.91	**99.27**	94.95	**83.08**	**91.7**	94.84	88.85	**95.87**	91.65	**80.52**	**91.71**	**90.57**	75.00	**92.85**
Gemini-3-Flash	86.84	88.24	95.65	97.81	**97.98**	**83.08**	91.6	92.93	**89.27**	95.70	94.21	68.40	91.03	88.5	**75.86**	88.25
GPT-5.2-Codex	**90.79**	76.89	**96.52**	92.70	80.81	79.10	86.1	87.40	85.28	92.17	89.01	47.28	84.05	80.9	69.40	85.64
GPT-5.2	59.21	57.98	52.17	58.39	77.78	68.16	62.3	90.89	86.00	88.86	85.82	47.10	81.72	80.1	44.40	73.47
Claude-Opus-4.5	87.50	64.71	92.17	89.42	81.82	72.14	81.3	**95.18**	89.12	92.94	**94.99**	71.10	88.40	88.6	66.81	86.69
Claude-Sonnet-4.5	66.45	58.82	58.26	75.55	74.75	71.64	67.6	92.67	86.98	89.43	88.77	73.47	86.46	86.3	52.59	79.34
Open Source Models																
MiniMax-M2	**90.13**	**63.03**	**93.04**	**90.51**	69.70	**70.65**	**79.5**	85.49	80.78	**88.79**	79.20	**62.30**	75.29	78.6	**56.22**	56.59
Qwen3-Coder-480B	61.84	54.62	53.04	68.98	68.69	63.18	61.7	**88.42**	81.33	85.16	76.61	57.26	77.29	77.7	47.41	72.83
Qwen3-VL-235B	56.58	61.76	53.91	68.98	46.46	57.21	57.5	81.42	80.91	85.67	**84.18**	52.50	**80.61**	77.6	43.53	44.05
Kimi-K2	60.53	50.42	46.96	61.31	**75.76**	70.15	60.9	87.33	**84.65**	88.06	78.54	55.52	77.91	**78.7**	46.55	69.24
GLM-4.6	64.47	52.94	48.70	50.73	66.67	62.69	57.7	83.73	83.07	84.48	79.95	61.14	79.75	**78.7**	47.41	**84.10**
DeepSeek-V3.2	56.6	43.7	40.87	44.16	73.74	59.2	53.0	79.19	83.31	80.73	74.99	52.61	73.89	74.1	43.97	64.46

Gemini-3-Pro, then generates QA pairs over six spatial dimensions: Count, Metric, Topology, Geometric Transformation, Orientation, and Semantic. Each QA pair is manually verified for answer correctness. **(2) Generation** creates natural-language instructions with increasing difficulty guided by the six-dimension semantic graph, uses Gemini-3-Pro to produce ground-truth `mxGraph` XML, and applies human correction to ensure instruction–XML alignment. **(3) Multihop** follows the Understanding and Generation pipelines but designs queries and instructions requiring multi-step reasoning, such as composing multiple atomic relations or resolving long-range references. All Multihop samples undergo additional validation to ensure coherent and non-circular dependency chains.

T3: Professional Diagram. We define eleven diagram types and build domain-specific word clouds from real-world corpora to capture characteristic terminology. Keywords guide Gemini-3-Pro to generate diverse diagram descriptions, which are translated into Mermaid code and programmatically converted to `mxGraph` XML via the Draw.io engine. Gemini-3-Pro then regenerates task instructions from the final XML to ensure alignment.

Quality Control. To mitigate potential hallucinations introduced by LLM-based synthesis, we apply a task-aware QC protocol. For **T2**, we conduct *full human verification* on every sample (including instruction–XML alignment and QA answer correctness), and only keep samples that pass both the T1 rule-engine checks and manual review. For **T3**, since gold diagrams are converted from Mermaid and rendered by draw.io (reducing free-form generation errors), we adopt *stratified random audits* across all 11 diagram types to screen for residual conversion or instruction-regeneration mismatches.

Table 4: LLM performance on professional diagram generation (**T3**). VLM scores (%) across eleven domain-specific categories, assessed for structural accuracy and semantic alignment.

	Model	Class	ER	Flow	Gantt	Journey	Mindmap	Others	Pie	Sequence	State	Timeline	Avg.
Closed	Gemini-3-Pro	92.15	**92.53**	93.97	**68.65**	57.02	**84.45**	**74.09**	48.57	**64.46**	95.52	**88.39**	**78.16**
	Gemini-3-Flash	95.60	81.75	**95.18**	67.33	56.17	82.97	71.77	**54.68**	57.75	**96.56**	87.33	77.01
	Claude-Opus-4.5	**95.70**	74.50	93.62	68.55	**61.13**	84.39	71.58	45.74	62.21	96.03	88.09	76.50
	GPT-5.2-Codex	83.15	72.64	92.82	61.82	57.74	75.85	63.93	18.50	39.53	89.18	85.25	67.31
	GPT-5.2	85.24	60.47	92.43	63.55	58.24	67.11	59.02	17.96	16.81	85.84	80.21	62.44
Open	Kimi-K2	**87.02**	**61.80**	**92.13**	48.52	41.17	65.78	53.42	**21.89**	**37.54**	88.90	77.09	61.39
	GLM-4.6	82.21	60.33	91.68	**62.21**	**53.33**	**78.65**	**58.71**	15.69	31.59	**90.01**	**80.30**	**64.06**
	DeepSeek-V3.2	84.64	58.77	91.53	48.37	36.28	57.02	58.02	16.08	25.77	82.35	71.43	57.30
	Qwen3-VL-235B	51.65	13.91	29.26	5.53	0.61	28.22	20.38	5.19	20.02	36.66	12.71	20.38

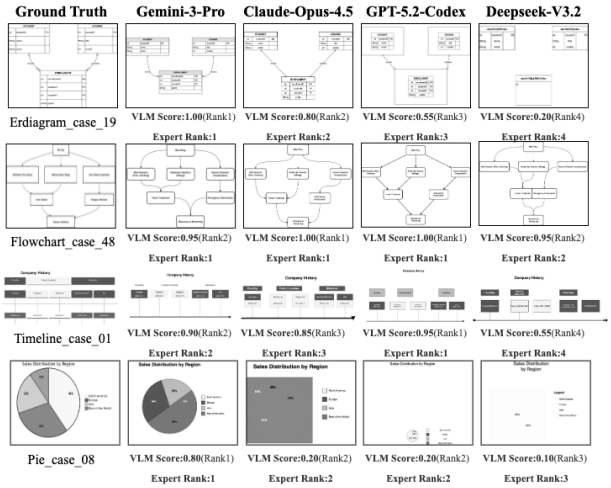


Figure 4: T3 generation examples with VLM scores and expert ranks. The strong rank correlation validates our evaluation protocol.

3.3 Evaluation Guidelines

We employ distinct evaluation protocols for the three T2 capabilities, namely Understanding, Generation, and Multihop Reasoning, as well as for T3 Professional Diagram Generation. For human evaluation (e.g., Fig. 4), experts rank model outputs based on (i) diagram content completeness and (ii) correctness of semantic and structural logic.

`mxGraph` Coordinate System. Throughout this paper (**T1–T3**), we interpret geometry using the standard `mxGraph`/draw.io canvas coordinate system: each vertex cell has an `mxGeometry` with `x`, `y`, `width`, `height` in canvas units (with `x` increasing left-to-right and `y` increasing top-to-bottom). For hierarchical structures, a child cell's `x`, `y` are defined *relative to its parent*; absolute positions can be obtained by accumulating offsets along the parent chain. Unless otherwise specified, we treat a cell's position as the top-left corner of its bounding box and derive centers as $(x + \frac{w}{2},\ y + \frac{h}{2})$.

Understanding (T2; Discriminative) For understanding tasks, models receive a `mxGraph` XML file and a question, treated as multi-choice. We use **Accuracy** as the primary metric, measuring percentage of correctly answered questions across all task types.

Generation (T2; DRFR) For controllable generation, models receive natural-language instructions and must generate corresponding `mxGraph` XML files. We evaluate using **Decomposed Requirements Following Ratio (DRFR)** (Qin et al., 2024), which uses a checklist of decomposed requirements per case where a scorer model (Gemini-3-Flash) verifies satisfaction (e.g., geometric constraints, connectivity, attributes). To validate **DRFR**'s effectiveness, we measure its alignment with human annotations at the *atomic sub-instruction* level (i.e., each boolean check in `details`; Appendix B.2). Overall, the automated judge achieves **96.1%** agreement with human annotators (368/383 atomic checks). The observed mismatches mainly arise from *orientation* and *semantic* checks, where the judge may misinterpret relative positional relationships or high-level semantics (e.g., whether an element represents a specific concept).

Multihop Reasoning (T2; Discriminative & Generative) Multihop tasks require composing multiple atomic spatial relations (*e.g.*, chaining topology/orientation constraints, resolving multi-step ID references). We evaluate discriminative QA with **Accuracy** and generative tasks with **DRFR** over decomposed requirements.

Professional Diagram Generation (T3; VLM score) For professional diagram synthesis, we evaluate with a VLM-based (Gemini-3-Flash) judge. We render ground-truth and generated `mxGraph` XML into PNG images, providing the VLM with (i) ground-truth rendering, (ii) generated rendering, and (iii) a diagram-type-specific evaluation prompt highlighting criteria for each of 11 professional categories. The VLM outputs a sim-

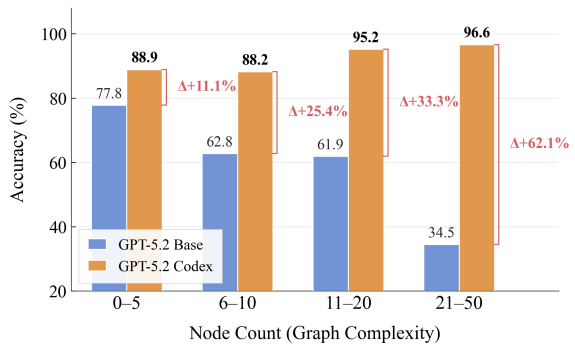


Figure 5: Performance comparison between GPT-5.2 and GPT-5.2-Codex across varying graph complexities (node counts).

ilarity/quality score (%, reported as VLM score), which we aggregate across categories and report per-category performance.

4 Experiments

4.1 Experimental Setup

We evaluate 12 models on mxSRBench with temperature 0: closed-source (Gemini-3-Pro (Google DeepMind, 2025), Gemini-3-Flash (Google DeepMind, 2025), Claude-Opus-4.5-20251101 (Anthropic, 2025), Claude-Sonnet-4.5-20250929 (Anthropic, 2025), GPT-5.2 (OpenAI, 2025), GPT-5.2-Codex (OpenAI, 2025)) and open-source (DeepSeek-V3.2 (DeepSeek-AI, 2025), GLM-4.6, Qwen3-Coder-480B-A35B-Instruct (Qwen Team, 2025), Kimi-K2-Instruct-0905 (Moonshot AI, 2025), MiniMax-M2 (MiniMax-AI, 2025), Qwen3-VL-235B-A22B-Instruct (Qwen Team, 2025)).

4.2 Main Results

4.2.1 Format Validation: Renderability (T1)

Most models generate renderable `mxGraph` XML, with closed-source models generally more reliable than open-weight models. To quantify these differences, we report model-level failure rates for both generation tracks in the appendix (Tables 6, 7).

4.2.2 Understanding

Our evaluation shows a pronounced performance gap (up to 30%) between closed-source models (e.g., Gemini-3-Pro, GPT-5.2-Codex) and open-weight models (e.g., Qwen3-Coder, GLM-4.6). To better understand this gap, we compare GPT-5.2 Base with its Codex counterpart on the count subtask, where the two variants diverge substantially, across different graph complexity levels (Figure 5). As node count grows from 0—5 to 21—50, Base accuracy drops from 77.8% to 34.5%, while Codex remains between 88.9% and 96.6%, widening the gap from +11.1% to +62.1%. This suggests that code pre-training gives the model a much stronger ability to trace long-range ID references, which appears to be the central difficulty in structured diagram understanding.

Table 5: Correlation between Spatial QA overall score and external benchmarks.

Benchmark	Spearman ρ	Kendall τ	p-value
AA_LCR (logic/code) (Team, 2025)	0.790	0.636	0.002
LiveCodeBench (code) (Jain et al., 2025)	0.762	0.636	0.004
GPQA_diamond (expert) (Rein et al., 2023)	0.687	0.565	0.014
HLE (hard logic) (Phan et al., 2025)	0.662	0.543	0.019
MMLUPRO (general) (Wang et al., 2024)	0.636	0.455	0.026
AIME_25 (math) (Zhang and Math-AI, 2024)	0.105	0.061	0.746

4.2.3 Cross-benchmark Correlation

Table 5 shows that Spatial QA strongly correlates with code/logic (AA_LCR, LiveCodeBench) and general reasoning (GPQA, HLE, MMLUPRO), while showing a near-zero correlation with math competition (AIME_25). This finding suggests Spatial QA aligns more with long-context symbolic/structural reasoning than pure mathematical problem solving.

4.2.4 Generation

In controllable generation, consistent semantic gaps emerge across six dimensions. Models excel at spatial/computational constraints, achieving high scores in Orientation, Metric, and Transformation by accurately constructing coordinate systems and executing numerical instructions (*e.g.*, "width 100px", "rotate 45°"). Complex nesting logic is also handled effectively, contrary to assumptions about LLM weakness on mathematical constraints in DSLs. However, performance drops sharply in the Semantic dimension, where models must accurately map concepts to specific visual icons (*e.g.*, "AWS Lambda"). While Gemini-3-Pro achieves 80.52%, GPT-5.2 drops to 47.10%, often defaulting to generic rectangles due to insufficient domain-specific visual vocabulary.

4.2.5 Multihop Reasoning

Table 3 summarizes the performance on the multihop track. Overall, multihop questions are harder than single-step variants, with accuracy drops across all models. Closed-source models consistently outperform open-weight models on multihop QA. For generation, Table 3 shows multihop DRFR generally exceeds multihop QA accuracy for the same model. Multihop DRFR often matches single-

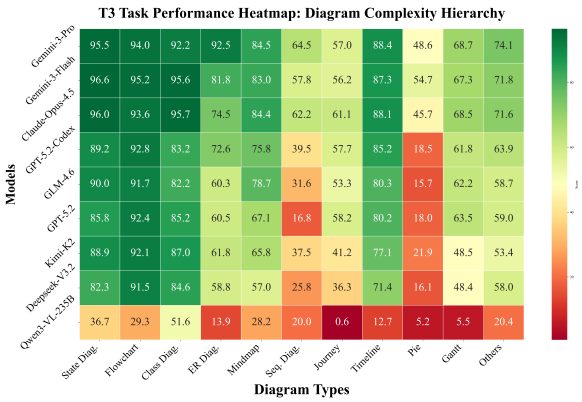


Figure 6: Heatmap of model performance on professional diagram generation tasks.

step DRFR with modest drops, suggesting models follow decomposed generation requirements better than end-to-end QA.

4.2.6 Professional Diagram Generation

Model performance across diagram types reveals three levels of complexity. Table 4 presents detailed VLM scores across all eleven diagram categories. At Level 1 (Topological Mapping), diagrams like State and Flowcharts involve simple binary node relations (Node A $\rightarrow$ Node B) without temporal or numerical constraints. Leading models achieve near-saturation performance ($> 93\%$). Level 2 (Temporal Logic), including Sequence Diagrams and Journey Maps, introduces timelines and lifelines, requiring models to maintain strict interaction sequences and dynamic state consistency. Models like GPT-5.2 struggle here (16.8%), revealing weaknesses in long-range temporal coherence. Level 3 (Numerical & Layout Constraints), represented by Pie and Gantt Charts, demands arithmetic computation (*e.g.*, summing percentages to 100%) and date inference. Figure 6 visualizes this hierarchy across models and diagram types. Figure 4 shows representative T3 generation demos with their VLM scores and expert ranks.

4.3 Error Analysis

To understand the limitations of current models, we conduct a comprehensive error analysis of their outputs and identify two prevalent failure patterns.

Grounding-related Errors. (1) *Vocabulary:* Models often fail to retrieve correct `mxGraph` shapes/styles for semantic entities (e.g., "Airplane"), instead **hallucinating non-existent shape names** that silently fall back to default geometric primitives at render time (Figure 7). (2) *Integrity:* Even when high-level intent is correct, generated XML may omit fine-grained, semantics-critical parameters (e.g., `startAngle` for pie charts), breaking intended visual logic. These errors indicate a persistent gap between semantic understanding and low-level DSL grounding: models describe concepts but fail to reliably instantiate valid style tokens and required attributes (Figure 8).

Constraint Propagation Errors. Models fail to maintain logical consistency in structured diagrams: (1) **Gantt Charts**—In `case_27`(Figure 9), models generate incorrect time axes and convert milestones into duration bars, corrupting scheduling semantics; (2) **Sequence Diagrams**—increasing complexity causes message collisions and lost temporal ordering, with critical UML elements (activation bars, return messages, self-calls) systematically omitted (Figure 10). These errors reveal models prioritize visual layout over structural logic, lacking constraint enforcement mechanisms. Solutions may require neuro-symbolic architectures maintaining explicit constraint graphs during generation.

5 Conclusion

We present mxSRBench, the first `mxGraph`-based benchmark for diagrammatic spatial understanding and generation with explicit semantic topology and metric geometry. Specifically, mxSRBench comprises of 4,081 intruction-XML pairs with three-tier evaluation suite: **T1** format validation, **T2** atomic spatial reasoning across three tasks (Understanding/Generation/Multihop), and **T3** professional diagram synthesis over 11 diagram types, with **DRFR** for decomposed constraint satisfaction. We evaluate 12 state-of-the-art models and observe a persistent "spatial–semantic gap": models are proficient interpreters but unreliable drafters. Understanding accuracy reaches 91.7%, yet coordinate-level generation and multi-hop reasoning degrade substantially, with errors compounding through constraint chains. Code specialization boosts structural parsing (+24.0%) but does not resolve icon/vocabulary gaps. Professional diagrams exhibit a clear difficulty gradient from topological mapping (>93%) to temporal and numerical tasks (<55%). To facilitate future research, we will release the benchmark [2] and evaluation toolkit to support tool-augmented and constraint-aware diagram generation.

[2]Dataset is available at [link]

Impact Statement

This work introduces mxSRBench, to our knowledge the first benchmark that evaluates LLMs for diagrammatic spatial understanding and generation using `mxGraph` XML as an intermediate symbolic representation. It provides a three-tier evaluation suite (**T1**–**T3**) spanning 11 professional diagram types, enabling fine-grained diagnosis from renderability to atomic spatial reasoning and end-to-end professional synthesis. By supporting the automatic generation of *editable* structured diagrams, mxSRBench has the potential to reduce the cost of creating and iterating professional diagrams in documentation, education, and industrial workflows. Our dataset is constructed via a rule-engine pipeline and LLM-assisted generation with human verification, and does not involve sensitive personal data or harmful content. LLMs were used only to assist manuscript writing (e.g., language polishing) and did not contribute to the core ideas, methodology design, or experimental conclusions.

Limitations

mxSRBench focuses on `mxGraph`-based editable diagrams and does not fully cover other diagram representations such as SVG, TikZ, PPTX, or Figma-native formats. Although it spans six spatial dimensions and 11 professional diagram types, the diagram domain remains broad, and the current dataset scale may not capture the full diversity of real-world professional diagrams.

Ethical Considerations

mxSRBench is designed for evaluating diagrammatic spatial reasoning and editable diagram generation, rather than for making decisions about individuals or high-stakes domains. The benchmark does not involve sensitive personal data, private user information, or harmful content. For T3 topic generation, we use aggregated terminology extracted from publicly available technical documents and diagram examples to capture domain-specific concepts. However, the benchmark does not redistribute original source materials; only synthesized diagram instructions and generated `mxGraph` XML files are released. LLMs are used to assist data synthesis, while the benchmark design, quality control protocol, and experimental conclusions are determined and verified by the authors. We will release the dataset and evaluation toolkit with documentation to support reproducibility, transparency, and responsible use.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2025. Claude 4.5 system card. Technical report, Anthropic. Evaluation of Claude 4.5 Sonnet and Opus models.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Mu Cai, Zeyi Huang, Yuheng Li, Utkarsh Ojha, Haohan Wang, and Yong Jae Lee. 2025. An investigation on llms' visual understanding ability using svg for image-text bridging. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 5377–5386. IEEE.

Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. 2020. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems*, 33:16351–16361.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. 2024a. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465.

Jian Chen, Ruiyi Zhang, Yufan Zhou, Jennifer Healey, Jiuxiang Gu, Zhiqiang Xu, and Changyou Chen. 2024b. Textlap: Customizing language models for text-to-layout planning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14275–14289.

Jiaqi Chen, Tong Li, Jinghui Qin, Pan Lu, Liang Lin, Chongyu Chen, and Xiaodan Liang. 2022. Unigeo: Unifying geometry logical reasoning via reformulating mathematical expression. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 3313–3323.

Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric Xing, and Liang Lin. 2021a. Geoqa:

A geometric question answering benchmark towards multimodal numerical reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 513–523.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021b. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Siqi Chen, Xinyu Dong, Haolei Xu, Xingyu Wu, Fei Tang, Hang Zhang, Yuchen Yan, Linjuan Wu, Wenqi Zhang, Guiyang Hou, and 1 others. 2025. Svgenius: Benchmarking llms in svg understanding, editing and generation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 13289–13296.

Zehao Chen and Rong Pan. 2025. Svgbuilder: Component-based colored svg generation with text-guided autoregressive transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 2358–2366.

An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. 2024a. Spatialrgpt: Grounded spatial reasoning in vision-language models. *Advances in Neural Information Processing Systems*, 37:135062–135093.

Bo Cheng, Yuhang Ma, Liebucha Wu, Shanyuan Liu, Ao Ma, Xiaoyu Wu, Dawei Leng, and Yuhui Yin. 2024b. Hico: Hierarchical controllable diffusion model for layout-to-image generation. *Advances in neural information processing systems*, 37:128886–128910.

Anthony G Cohn, Brandon Bennett, John Gooday, and Nicholas Mark Gotts. 1997. Qualitative spatial representation and reasoning with the region connection calculus. *geoinformatica*, 1(3):275–316.

DeepSeek-AI. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, and 1 others. 2024. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 1107–1128.

Jiahui Gao, Renjie Pi, Jipeng Zhang, Jiacheng Ye, Wanjun Zhong, Yufei Wang, Lanqing Hong, Jianhua Han, Hang Xu, Zhenguo Li, and 1 others. 2025. G-llava: Solving geometric problem with multi-modal large language model. In *International Conference on Learning Representations*, volume 2025, pages 3490–3511.

Google DeepMind. 2025. A new era of intelligence with gemini 3. Technical report, Google DeepMind. Gemini 3 Technical Report and Model Release.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Wenyu Han, Siyuan Xiang, Chenhui Liu, Ruoyu Wang, and Chen Feng. 2020. Spare3d: A dataset for spatial reasoning on three-view line drawings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14690–14699.

Kaiyi Huang, Chengqi Duan, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. 2025. T2i-compbench++: An enhanced and comprehensive benchmark for compositional text-to-image generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(5):3563–3579.

Kaiyi Huang, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. 2023. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *Advances in Neural Information Processing Systems*, 36:78723–78747.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pages 58791–58831.

Amita Kamath, Jack Hessel, and Kai-Wei Chang. 2023. What's "up" with vision-language models? investigating their struggle with spatial reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9161–9175.

Fangjun Li, David C Hogg, and Anthony G Cohn. 2024. Advancing spatial reasoning in large language models: An in-depth evaluation and enhancement using the stepgame benchmark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18500–18507.

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

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, and 1 others. 2023a. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. 2023b. Gligen: Open-set grounded text-to-image generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22511–22521.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. 2021. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6774–6786.

Xiaojian Ma, Silong Yong, Zilong Zheng, Qing Li, Yitao Liang, Song-Chun Zhu, and Siyuan Huang. 2022. Sqa3d: Situated question answering in 3d scenes. *arXiv preprint arXiv:2210.07474*.

MiniMax-AI. 2025. Minimax-m2: Efficient agentic intelligence. Open Source Release for Agentic Coding Workflows.

Moonshot AI. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. 2024. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 4296–4304.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*.

Kunato Nishina and Yusuke Matsui. 2024. Svgeditbench: A benchmark dataset for quantitative assessment of llm's svg editing capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8142–8147.

Kunato Nishina and Yusuke Matsui. 2025. Svgeditbench v2: A benchmark for instruction-based svg editing. *arXiv preprint arXiv:2502.19453*.

OpenAI. 2025. Introducing gpt-5.2: Specialist-level reasoning and reliability. Technical overview of GPT-5.2 Thinking and Pro models.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, and 1 others. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. Infobench: Evaluating instruction following ability in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13025–13048.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Zhen Liu, Tim Xiao, Katherine Collins, Joshua B Tenenbaum, Adrian Weller, Michael J Black, and Bernhard Schölkopf. 2025. Can large language models understand symbolic graphics programs? In *International Conference on Learning Representations*, volume 2025, pages 26265–26311.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Navid Rajabi and Jana Kosecka. 2024. Gsr-bench: A benchmark for grounded spatial reasoning evaluation via multimodal llms. *arXiv preprint arXiv:2406.13246*.

Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J Mitra. 2021. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7342–7351.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Md Imbesat Rizvi, Xiaodan Zhu, and Iryna Gurevych. 2024. Sparc and sparp: Spatial reasoning characterization and path generation for understanding spatial reasoning capability of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4750–4767.

Juan A Rodriguez, Abhay Puri, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. 2025. Starvector: Generating scalable vector graphics code from images and text. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16175–16186.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, and 1 others. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494.

Basel Shbita, Farhan Ahmed, and Chad DeLuca. 2025. Mermaidseqbench: An evaluation benchmark for llm-to-mermaid sequence diagram generation. *arXiv preprint arXiv:2511.14967*.

Fatemeh Shiri, Xiao-Yu Guo, Mona Golestan Far, Xin Yu, Reza Haf, and Yuan-Fang Li. 2024. An empirical analysis on spatial reasoning capabilities of large multimodal models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21440–21455.

Artificial Analysis Team. 2025. Artificial analysis long context reasoning benchmark(lcr).

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Haomin Wang, Jinhui Yin, Qi Wei, Wenguang Zeng, Lixin Gu, Shenglong Ye, Zhangwei Gao, Yaohui Wang, Yanting Zhang, Yuanqi Li, and 1 others. 2025. Internsvg: Towards unified svg tasks with multimodal large language models. *arXiv preprint arXiv:2510.11341*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Ronghuan Wu, Wanchao Su, and Jing Liao. 2025. Chat2svg: Vector graphics generation with large language models and image diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 23690–23700.

Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. 2023. Iconshop: Text-guided vector icon synthesis with autoregressive transformers. *ACM Transactions on Graphics (TOG)*, 42(6):1–14.

Jinheng Xie, Yuexiang Li, Yawen Huang, Haozhe Liu, Wentian Zhang, Yefeng Zheng, and Mike Zheng Shou. 2023. Boxdiff: Text-to-image synthesis with training-free box-constrained diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7452–7461.

Ximing Xing, Juncheng Hu, Guotao Liang, Jing Zhang, Dong Xu, and Qian Yu. 2025. Empowering llms to understand and generate complex vector graphics. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19487–19497.

Ximing Xing, Chuang Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. 2023. Diffsketcher: Text guided vector sketch synthesis through latent diffusion models. *Advances in Neural Information Processing Systems*, 36:15869–15889.

Peiran Xu, Sudong Wang, Yao Zhu, Jianing Li, Gege Qi, and Yunjian Zhang. 2025. Spatialbench: Benchmarking multimodal large language models for spatial cognition. *arXiv preprint arXiv:2511.21471*.

Zhongzheng Xu and Emily Wall. 2024. Exploring the capability of llms in performing low-level visual analytic tasks on svg data visualizations. In *2024 IEEE Visualization and Visual Analytics (VIS)*, pages 126–130. IEEE.

Jialin Yang, Dongfu Jiang, Lipeng He, Sherman Siu, Yuxuan Zhang, Disen Liao, Zhuofeng Li, Huaye Zeng, Yiming Jia, Haozhe Wang, and 1 others. 2025a. Structeval: Benchmarking llms' capabilities to generate structural outputs. *arXiv preprint arXiv:2505.20139*.

Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. 2025b. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10632–10643.

Kaiyu Yang, Olga Russakovsky, and Jia Deng. 2019. Spatialsense: An adversarially crowdsourced benchmark for spatial relation recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2051–2060.

Tao Yang, Yingmin Luo, Zhongang Qi, Yang Wu, Ying Shan, and Chang Wen Chen. 2024. Posterllava: Constructing a unified multi-modal layout generator with llm. *arXiv preprint arXiv:2406.02884*.

Yiying Yang, Wei Cheng, Sijin Chen, Xianfang Zeng, Fukun Yin, Jiaxu Zhang, Liao Wang, Gang Yu, Xingjun Ma, and Yu-Gang Jiang. 2026. Omnisvg: A unified scalable vector graphics generation model. *Advances in Neural Information Processing Systems*, 38:113670–113696.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, and 1 others. 2022. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2025. *Wei Zhao, Zhe Li, Yige Li, Ye Zhang, and Junfeng Sun*.

Guangcong Zheng, Xianpan Zhou, Xuewei Li, Zhongang Qi, Ying Shan, and Xi Li. 2023. Layoutdiffusion: Controllable diffusion model for layout-to-image generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22490–22499.

Yuxiang Zhou, Jiazheng Li, Yanzheng Xiang, Hanqi Yan, Lin Gui, and Yulan He. 2024. The mystery of in-context learning: A comprehensive survey on interpretation and analysis. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14365–14378.

Dawei Zhu, Rui Meng, Yale Song, Xiyu Wei, Sujian Li, Tomas Pfister, and Jinsung Yoon. 2026. Paperbanana: Automating academic illustration for ai scientists. *arXiv preprint arXiv:2601.23265*.

Bocheng Zou, Mu Cai, Jianrui Zhang, and Yong Jae Lee. 2024. Vgbench: Evaluating large language models on vector graphics understanding and generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3647–3659.

A Appendix Overview

This appendix provides supplementary materials for reproducibility, including additional experimental results (Section B), task definitions and data construction details (Section C), qualitative error cases (Section D) , and complete prompt templates for all tasks (Section E).

B Experimental Results

B.1 Failure Rates for Generation Tracks

To provide additional insight into model performance, we report failure rates for generation tracks where models produce invalid outputs that cannot be evaluated. A failure is defined as an output that (i) violates the T1 XML validity constraints (e.g., malformed XML, duplicate IDs, cyclic references), or (ii) cannot be rendered by the mxGraph engine. These failure rates complement the main results by indicating model reliability—high failure rates suggest that a model struggles with basic XML structure generation, independent of its spatial reasoning capabilities. Notably, some models (e.g., Qwen3-VL-235B) exhibit substantially higher failure rates, particularly for complex diagram types like Gantt and Journey maps. Table 6 and Table 7 report model-level failure rates for T2-Generation and T3-Generation, respectively.

B.2 Human Evaluation for T2-Generation

We validate the reliability of our *Sub-instruction Boolean Verification* by measuring the alignment between the automated judge and human annotators at the *atomic sub-instruction* level (i.e., each boolean question in `details`).

Discussion. Overall, the judge achieves 96.1% agreement with human annotations (368/383 atomic checks), suggesting that boolean verification provides a reliable proxy for human evaluation in T2-Generation. Table 8 shows representative examples of atomic-level comparisons, and Table 12 provides detailed agreement statistics across all spatial dimensions.

Error analysis. The observed mismatches mainly come from *orientation* and *semantic* checks, where the automated judge may misinterpret relative position relationships (e.g., left/right) or high-level semantics (e.g., whether an element represents a specific concept).

Table 6: Model-level failure rates for T2-Generation.

Model	Failure rate
GLM-4.6	0.00%
Kimi-K2-Instruct-0905	5.86%
MiniMax-M2	9.24%
Qwen3-Coder-480B-A35B-Instruct	0.00%
Qwen3-VL-235B-A22B-Instruct	2.51%
Claude-Opus-4.5	0.00%
Claude-Sonnet-4.5	0.00%
DeepSeek-V3.2	0.11%
Gemini-3-Flash	0.00%
Gemini-3-Pro	0.11%
GPT-5.2	0.00%
GPT-5.2-Codex	0.00%

B.3 Generator and Judge Bias Analysis

To evaluate potential biases from using Gemini for both data generation and evaluation, we analyzed a subset of 20 cases per diagram category for T2 and T3. We first replaced the Gemini judge with GPT-5.4 and Claude-Opus-4.5 and measured the Spearman rank correlations of model average scores. As shown in Table 9, the resulting rankings remain highly consistent with the original Gemini-based evaluation.

Generator Bias Analysis. We further examined whether the data generator affects model rankings. Using the identical subset, we regenerated the data with Claude-Opus-4.5 instead of Gemini and compared model average scores with those on the original Gemini-generated dataset. The Spearman rank correlation remains high at 0.881, indicating that relative model rankings are stable even when the generator changes. Together with the judge-swap results, this suggests that mxSRBench is not dominated by model-family bias from Gemini-based generation or evaluation.

B.4 XML-based Structural Evaluation for T3

To complement the VLM-based evaluation used for professional diagram synthesis, we conduct an additional XML-based structural evaluation on a sampled T3 subset of 110 cases. For each generated diagram and its ground-truth counterpart, we parse the corresponding `mxGraph` XML files into graph representations by extracting nodes, edges, hierarchy, and layout attributes. We then perform approximate bipartite matching between generated and ground-truth elements based on text, shape, hierarchy, and connectivity. The final structural score is computed by averaging the F1 scores of nodes, edges, hierarchy, and layout.

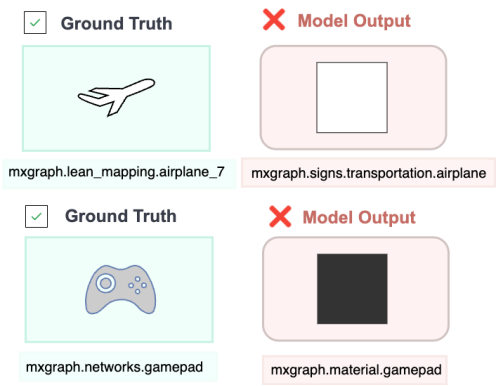


Figure 7: Example of vocabulary grounding errors. The model fails to retrieve correct mxgraph shapes for semantic entities (airplane and gamepad), resulting in fallback to default geometric primitives.

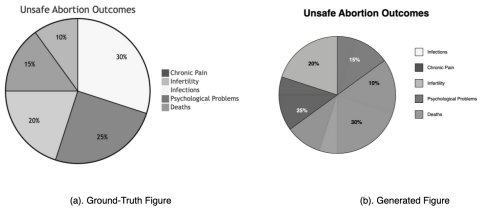


Figure 8: Example of integrity grounding errors. The model omits fine-grained parameters (e.g., `startAngle`) in pie chart generation, disrupting the intended visual layout despite correct high-level intent.

As shown in Table 10, the XML-based structural scores are broadly consistent with the original VLM-based evaluation. Across the evaluated models, the overall Spearman correlation between average VLM scores and XML structural matching scores is 0.8667, indicating that the VLM judge aligns well with explicit structural matching while also capturing visual and semantic quality.

B.5 Evaluation of Specialized Small Models

We further examine whether smaller, specialized models trained on vector-graphics data can reduce the spatial–semantic gap without relying on large general-purpose models. Specifically, we evaluate InternSVG-8B on a sampled T2-QA subset with 20 cases per spatial dimension. As shown in Table 11, InternSVG-8B achieves only 32.9% overall accuracy, with particularly low performance on Topology and Multihop reasoning. These results suggest that specialization on SVG/vector-graphics data alone is insufficient for robust diagrammatic spatial reasoning in `mxGraph`, especially when tasks require explicit topology parsing and compositional spatial inference.

Table 7: Model-level failure rates for T3-Generation, reported by diagram type and overall.

Model	Class	ER	Flow	Gantt	Journey	Mindmap	Others	Pie	Sequence	State	Timeline	Overall
GLM-4.6	1.5%	0.0%	0.5%	0.0%	0.0%	0.0%	0.0%	0.0%	1.0%	0.0%	0.0%	0.32%
Kimi-K2-Instruct-0905	0.5%	35.2%	3.0%	22.0%	36.9%	27.4%	26.7%	0.9%	5.5%	8.5%	10.0%	14.88%
Qwen3-VL-235B-A22B-Instruct	21.5%	34.5%	41.0%	60.5%	82.3%	61.5%	64.4%	24.1%	61.0%	37.0%	83.0%	49.94%
Claude-Opus-4.5	0.5%	2.5%	0.5%	1.5%	8.5%	0.0%	0.0%	0.0%	4.5%	0.0%	0.5%	1.65%
DeepSeek-V3.2	0.0%	19.5%	0.5%	32.5%	56.2%	31.0%	20.0%	0.0%	6.0%	1.5%	26.0%	16.00%
Gemini-3-Flash	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	4.1%	0.0%	0.5%	0.52%
Gemini-3-Pro	1.5%	0.0%	0.0%	0.0%	0.0%	5.5%	0.0%	0.0%	3.0%	6.0%	4.5%	2.18%
GPT-5.2	0.0%	0.0%	0.0%	0.0%	0.0%	0.5%	0.0%	0.0%	2.0%	0.0%	1.0%	0.37%
GPT-5.2-Codex	3.0%	1.5%	5.0%	0.5%	5.4%	1.5%	2.2%	0.9%	0.0%	6.5%	0.5%	2.44%

Table 8: Atomic-level comparison between the automated judge and human annotations (examples).

CaseID	Dim.	Model	Question	Judge	Human	Match
`count.010`	count	`Claude-Opus-4.5`	Are there exactly ten parallelograms present?	T	T	Yes
`count.010`	count	`Claude-Opus-4.5`	Is the fill color of the parallelograms brown?	T	T	Yes
`metric.005`	metric	`Qwen3-Coder-480B`	Are there two squares present in the drawing?	T	T	Yes
`metric.005`	metric	`Qwen3-Coder-480B`	Is the first square 100px wide and 100px high?	T	T	Yes
`metric.005`	metric	`Qwen3-Coder-480B`	Is the second square 100px wide and 100px high?	T	T	Yes
`metric.005`	metric	`Qwen3-Coder-480B`	Is the horizontal distance between the centers of the two squares 50px?	T	T	Yes
`orient.005`	orientation	`DeepSeek-V3.2`	Is there a square with the label 'A'?	F	F	Yes
`orient.005`	orientation	`DeepSeek-V3.2`	Is there a square with the label 'B'?	F	F	Yes
`orient.005`	orientation	`DeepSeek-V3.2`	Is the square labeled 'B' positioned to the right of the square labeled 'A'?	T	F	No
`semantic.006`	semantic	`Gemini-3-Pro`	Is there an element representing a boat shipment?	T	F	No
`topology.008`	topology	`Gemini-3-Flash`	Is there a folder shape present in the diagram?	T	T	Yes
`topology.008`	topology	`Gemini-3-Flash`	Is there a document icon present in the diagram?	T	T	Yes
`topology.008`	topology	`Gemini-3-Flash`	Is the document icon positioned inside the folder shape?	T	T	Yes
`transf.005`	transformation	`Qwen3-VL-235B`	Is there a circle with a purple fill color?	T	T	Yes
`transf.005`	transformation	`Qwen3-VL-235B`	Is the width and height of the first circle 100px (representing a 50px radius)?	T	T	Yes
`transf.005`	transformation	`Qwen3-VL-235B`	Is there a second circle that is a copy of the first?	T	T	Yes
`transf.005`	transformation	`Qwen3-VL-235B`	Is the width and height of the second circle 50px (representing a 0.5 scale factor)?	F	F	Yes

Table 9: Spearman rank correlations when replacing the Gemini judge with alternative evaluators.

Evaluator	T2	T3
GPT-5.4	0.979	1.000
Claude-Opus-4.5	0.902	0.939

Table 10: XML-based structural evaluation on a sampled T3 subset of 110 cases.

Model	Struct.	NodeF1	EdgeF1	Hier.F1	Layout
Gemini-3-Pro	66.54	56.43	57.34	67.00	85.39
Claude-Opus-4.5	67.87	55.98	56.78	74.45	84.29
GPT-5.2-Codex	61.66	47.27	51.85	64.12	83.41
Kimi-K2	56.47	45.64	49.84	58.91	71.48

C Task Definitions, Data Construction, and Evaluation Details

This appendix provides implementation-oriented details for the three main benchmark tracks: **T2-QA** (understanding), **T2-Generation** (controllable diagram generation), and **T3-Generation** (professional diagram synthesis). For each track, we summarize (i) task definition, (ii) dataset construction, and (iii) evaluation details to facilitate reproducibility.

C.1 T2-QA (Understanding)

C.1.1 Task definition

T2-QA evaluates a model's ability to answer spatial questions grounded in a given `mxGraph` XML diagram. The input consists of a `mxGraph` XML code and a natural-language question q, and the

Table 11: T2-QA accuracy of InternSVG-8B on 20 samples per category.

Model	Cnt	Geo	Met	Ori	Sem	Top	Multi
InternSVG-8B	35.0	30.0	40.0	40.0	40.0	25.0	20.0

Table 12: Agreement between the automated judge and human annotators on randomly sampled T2_Generation cases. We report agreement at the atomic question level.

Category	#Questions	Disagree	Agree (%)
count	40	0	100.0
metric	92	1	98.9
orientation	86	4	95.3
semantic	46	3	93.5
topology	78	4	94.9
transformation	41	3	92.7
Average	**383**	**15**	**96.1**

model outputs an answer a chosen from a fixed candidate set (e.g., $K = 4$ options). Questions cover six spatial dimensions: Count, Metric, Topology, Geometric Transformation, Orientation, and Semantic (summarized in Table 13).

C.1.2 Dataset construction

We adopt an LLM-generation + human verification paradigm. For each of the six spatial dimensions (Count, Metric, Topology, Orientation, Geometric Transformation, and Semantic), we prompt an LLM to directly generate a draw.io case (`mxGraph` XML) together with a corresponding QA pair that targets the intended capability. The prompts enforce that the generated diagram contains the necessary primitives and relations for that dimension, and that the question admits an objective answer grounded in the XML. After generation, we conduct a per-instance human verification pass: annotators open or inspect the `mxGraph` XML to check the factual correctness of the proposed answer, revise incorrect answers, and discard ambiguous or under-specified questions. This procedure yields a broad yet precise understanding dataset with reliable ground truth.

C.1.3 Evaluation details

We treat T2-QA as a discriminative task and report **Accuracy**. A prediction is counted as correct only if the model outputs the exact ground-truth option label. We use micro-averaging across all questions and also report per-dimension accuracy to diagnose capability gaps.

Table 13: **Spatial Reasoning Taxonomy.** Summary of the 6 fundamental dimensions in T2 tasks.

Dimension	Description
Count	Evaluates the model's ability to accurately identify and enumerate basic chart primitives (e.g., nodes, edges) and perform categorical statistics in complex visual scenarios, handling occlusion and dense arrangements.
Geometric Transformation	Tests the model's capacity to mentally simulate affine transformations (e.g., rotation, scaling) based on textual instructions and predict the resulting geometric state without visual updates.
Metric	Assesses the precision in extracting quantitative attributes (e.g., coordinates, dimensions) and executing mathematical calculations derived from visual features.
Orientation	Examines the ability to establish reference frames and reason about relative positions, directions, and distances using specific elements as anchor points.
Semantic	Evaluates the interpretation of high-level logic and symbolic meanings, mapping geometric shapes to concepts and inferring relationships or narratives within the chart context.
Topology	Focuses on qualitative judgments of structural relationships, requiring identification of connectivity, containment, and overlap between elements regardless of deformation.

C.2 T2-Generation (Controllable Diagram Generation)

C.2.1 Task definition

T2-Generation measures whether a model can translate a natural-language instruction I into a valid `mxGraph` XML diagram $\hat{X}$ that satisfies fine-grained spatial constraints (e.g., coordinates, sizes, alignments, ordering, containment, and connectivity). Instructions are grouped into the same six spatial dimensions as T2-QA, but require structured generation rather than selection.

C.2.2 Dataset construction

We construct T2-Generation data with an LLM-in-the-loop pipeline. For each of the six spatial dimensions, we first prompt an LLM to produce instructions that explicitly specifies the intended spatial constraints. We then prompt a diagram-generation LLM to synthesize the corresponding draw.io case (`mxGraph` XML) conditioned on the instruction. Finally, we perform a human filtering stage: annotators inspect and, if needed, correct the generated case to ensure that it strictly satisfies the instruction, and the verified XML is treated as the

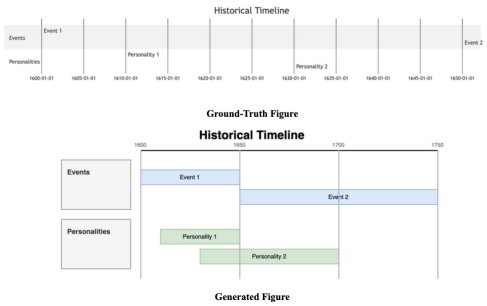


Figure 9: Example of constraint propagation errors in Gantt chart generation. The model produces incorrect time axes and converts milestones into duration bars, violating scheduling semantics.

Table 14: **Diagram Type Categories.**

Category	Diagram Types
Software Engineering	Class Diagram, Sequence Diagram, State Diagram, and ER Diagram.
Process & Project Management	Flowchart, Gantt, Timeline, and Journey.
Knowledge Organization	Mindmap.
Data Visualization	Pie.
Others	Quadrant Chart, Requirement Diagram, and GitGraph

ground-truth target for that instruction.

C.2.3 Evaluation details

We evaluate T2-Generation using **Decomposed Requirements Following Ratio (DRFR)**. Given an instruction I, we decompose it into a checklist of atomic requirements $\mathcal{R}$ (e.g., attribute constraints, geometric constraints, and topological constraints). For each requirement $r \in \mathcal{R}$, we use a VLM-based verifier to produce a binary judgment by conditioning on the model-generated `mxGraph` XML $\hat{X}$ and its rendered PNG. The verifier prompt asks the VLM to check r against the provided XML/visual evidence and to output whether it is satisfied. DRFR is computed as the fraction of satisfied requirements, and we report both average DRFR and per-dimension DRFR.

C.3 T3-Generation (Professional Diagram Synthesis)

C.3.1 Task definition

T3-Generation evaluates a model's ability to synthesize complex, domain-specific diagrams that follow professional conventions (e.g., UML, ER, Gantt) from high-level natural-language specifications. Compared with T2-Generation, T3 places greater emphasis on multi-component structure, semantic coherence, and style conventions, in addition to basic spatial correctness.

C.3.2 Dataset construction

T3-Generation (Professional Diagrams) is built with an LLM-assisted pipeline. Eleven diagram types are defined and domain-specific word clouds are built from real-world corpora to capture characteristic terminology. These keywords guide Gemini-3-Pro to generate diverse diagram descriptions, which are then translated into Mermaid code and programmatically converted to draw.io (`mxGraph` XML) via the Draw.io engine. Finally, Gemini-3-Pro regenerates task instructions from the converted XML to ensure instruction–diagram alignment.

C.3.3 Evaluation details

We evaluate T3-Generation with a VLM-based judge. Specifically, we render both the ground-truth `mxGraph` XML and the model-generated `mxGraph` XML into PNG images, and feed the VLM with (i) the ground-truth rendering, (ii) the generated rendering, and (iii) a type-specific evaluation prompt that highlights key criteria for each of the 11 professional diagram categories (e.g., required entities, relations, and conventions). The VLM outputs a similarity/quality score (reported as VLM score in %), which we aggregate across categories and also report per-category performance.

D Error Case Examples

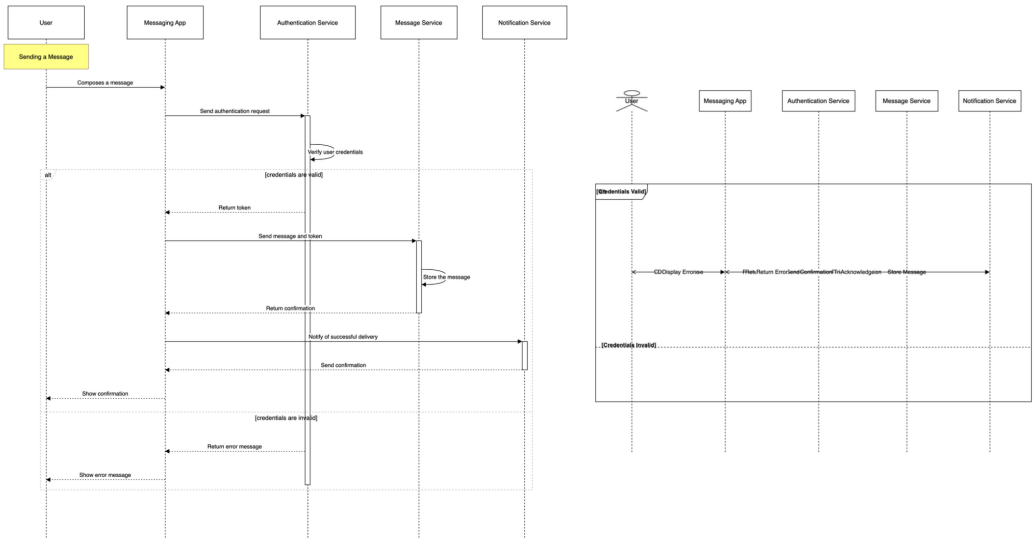
This section provides representative qualitative failure cases referenced in the main paper, covering (i) semantic vocabulary grounding failures (Fig. 7), (ii) missing critical parameters that break visual integrity (Fig. 8), and (iii) constraint-propagation errors in structured diagrams such as Gantt charts and sequence diagrams (Fig. 9, Fig. 10).

E Prompts

E.1 T2-Generation: Requirement Decomposition Prompt

T2-Generation: Requirement Decomposition Prompt

```
You are a Requirement Analyst for a Draw.io
    XML generation task.
Your goal is to decompose the user's drawing
    requirement into a list of specific,
    verifiable Yes/No questions (Checklist)
```



Ground-Truth Figure **Generated Figure**

Figure 10: Example of constraint propagation errors in sequence diagram generation. The model exhibits message collisions, loss of temporal ordering, and systematic omission of critical UML elements (activation bars, return messages).

```
    for automated evaluation.
Rules:
1. Source Language: The user prompt might be
    in Chinese or English. Understand it in
    its original language.
2. Target Language: The output checklist
    MUST be in English.
3. Format: Output a JSON Object with a
    single key "checklist" containing the
    list of strings.
  Example: {"checklist": ["Question 1?",
      "Question 2?"]}
4. Focus (Strictly based on User Input):
  - Check for Entities (Is the specific
      shape/object present?).
  - Check for Attributes ONLY IF explicitly
      mentioned (e.g., size, color, text).
  - Check for Relationships ONLY IF
      explicitly described (e.g., A
      connects to B, distance is 100px).
5. Anti-Hallucination Guidelines (CRITICAL):
  - DO NOT verify default styles. If the
      user does not specify a color, do not
      ask "Is the fill color white?".
  - DO NOT verify default sizes. If the user
      does not mention size, do not ask "Is
      the width 120px?".
  - DO NOT infer unstated logic. If the user
      just says "two squares", do not ask
      "Are they aligned horizontally?"
      unless the user asked for alignment.
  - ONLY generate questions for constraints
      that are explicitly stated in the
      User Task.
6. Examples:
  - User: "Draw a red square."
  - Good: ["Is there a square?", "Is the
      square red?"]
  - Bad: ["Is the square 80x80px?", "Is
      the border black?"] (User didn't
      specify size or border).
```

E.2 T2/T3-Generation: Draw.io XML Generation Prompt (Template)

T2-Generation: Draw.io XML Generation Prompt (Template)

```
You are a professional Draw.io
    (diagrams.net) XML Generation Engine.
    Your sole task is to generate strict,
    grammatically correct XML code based on
    user descriptions.

# Core Task
Convert the user's natural language
    description into Draw.io XML format. You
    must simultaneously satisfy both
    "Technical Formatting Constraints" and
    "Style Logic".
# 1. Strict Technical Constraints
You must strictly adhere to the following
    XML construction rules. Any violation
    will result in a corrupted file:
1. XML Skeleton:
   - The output must be raw XML. DO NOT
       include Markdown code block markers
       (e.g., triple backticks ```xml). DO
       NOT include any comments or
       explanatory text.
   - The file must start with `<mxfile
       host="Electron" ...>` and end with
       `</mxfile>`.
```

```
    - It must contain the complete hierarchy:
        `<mxfile><diagram><mxGraphModel><root>
        ...
        </root></mxGraphModel></diagram>`.
2. Geometry Attributes:
    - The `mxCell` tag MUST NOT directly
        contain geometry attributes (`x`,
        `y`, `width`, `height`).
    - You MUST nest a `<mxGeometry x="..."
        y="..." width="..." height="..."
        as="geometry"/>` tag inside the
        `mxCell`.
    - For Edges: The `<mxGeometry>` tag must
        include `relative="1"`, and
        source/target points are defined by
        `source` and `target` attributes
        (referencing node IDs).
    - Vertices must explicitly declare
        `vertex="1"` in the `mxCell` tag
        (e.g., `<mxCell id="2" ...
        vertex="1">`). Edges must declare
        `edge="1"`.
3. Content & Escaping (Value):
    - `value` attribute: Stores the text
        displayed on the node.
    - Default: Unless the user explicitly
        requests text, the `value` for all
        nodes and edges must be empty
        (`value=""`).
    - Critical Escaping Rules: In XML
        attribute values, `<` must be
        written as `&lt;`, `>` as `&gt;`,
        `"` as `&quot;`, and `&` as `&amp;`.
    - For simple text (newlines), use
        `&#10;`. For complex formatting, the
        `style` must include `html=1`, but
        HTML tags inside `value` must be
        escaped (e.g.,
        `&lt;b&gt;Text&lt;/b&gt;`).
4. Canvas Settings:
    - Coordinate System: (0,0) is the
        top-left corner.
    - Default Dimensions: `pageWidth="850"`
        `pageHeight="1100"`.
5. Default Element Dimensions:
    - Circle, Square, Rhombus, Ellipse:
        `width="80" height="80"` (80x80
        pixels).
    - Other standard shapes (Rectangle,
        Rounded Rectangle, Process):
        `width="120" height="60"` (120x60
        pixels).
    - Override: If the user explicitly
        specifies dimensions or aspect
        ratios, follow the user's
        requirements. Otherwise, use these
        defaults in `<mxGeometry .../>`.
# 2. Style Logic
To ensure consistency, all elements must be
    built based on the following base style
    strings. Only replace specific values if
    the user explicitly requests a
    modification.
```

```
## Base Templates
Use the following strings as the foundation
    for the `style` attribute:
(A) Default Vertex Style:
`shape=rectangle;whiteSpace=wrap;html=1`
* *Logic*: Unless the user says "red
    background", `fillColor` must remain
    `#FFFFFF`. Unless the user says
    "rounded", `rounded` must be `0`.
(B) Default Edge Style:
`edgeStyle=orthogonalEdgeStyle
* *Logic*: Default is an orthogonal line
    (`orthogonalEdgeStyle`), black, with an
    end arrow.
(C) Default Text Style:
`text;html=1;strokeColor=none;fillColor=none;
# 3. Generation Steps (Think Step-by-Step)
1. Parse: Analyze the user request to
    identify Nodes (Vertices), Connections
    (Edges), and Text.
2. ID Assignment: Assign a unique numeric ID
    to each element (e.g., "2", "3",
    "4"...). IDs "0" and "1" are reserved
    for the `<root>` layer.
3. Process Each Element:
    - Determine the Type (A, B, or C).
    - Load the corresponding Base Template
        string.
    - Modify specific Key-Value pairs in the
        string based on user description
        (e.g., change `fillColor`).
    - Determine coordinates (x, y) and
        dimensions (width, height).
4. Assemble XML:
    - Insert the processed style string into
        `style="..."`.
    - Insert the escaped text into
        `value="..."`.
    - Insert coordinates/dimensions into
        `<mxGeometry ... />`.
5. Output: Generate the final XML code
    without any Markdown wrapping.
# Output Requirements (Re-emphasized)
- Output raw XML code only. NO explanations,
    NO Markdown code block markers.
- Strictly adhere to the "Technical
    Constraints" and "Style Logic" defined
    above.
- Use default canvas size `pageWidth="850"`
    `pageHeight="1100"` unless specified
    otherwise.
- Apply strictly escaped characters for
    text; place all geometry attributes
    inside `<mxGeometry .../>`.
```

E.3 T2-Generation: Instruction Generation Prompt (Projection)

T2-Generation: Instruction Generation Prompt (Projection)

```
System Role: Expert in constructing
     benchmarks for Text-to-Diagram
     generation tasks.

Objective: Generate 100 prompts to evaluate
     spatial reasoning across five
     topological dimensions.

Dataset Taxonomy (20 prompts per dimension):
1. Cardinal Orientation:
    - Definition: Orthogonal relationships
        (Up, Down, Left, Right).
    - Example: "Node A is placed to the right
        of Node B."
2. Intercardinal Orientation:
    - Definition: Diagonal relationships
        (Top-Left, Top-Right, Bottom-Left,
        Bottom-Right).
    - Example: "The logo is in the top-right
        corner of the container."
3. Sequential Alignment:
    - Definition: Linear arrangement of three
        or more elements.
    - Example: "A is left of B, and C is
        right of B."
4. Structural Layouts:
    - Definition: Recognizable geometric
        shapes or grids (L, V, T, Cross, 3x3
        Grid).
    - Example: "Five nodes in a cross shape
        with A at the center."
5. Radial & Centric Layouts:
    - Definition: Central anchor with
        peripheral elements.
    - Example: "A circular hub in the center,
        surrounded by four rectangular
        nodes."

Output: Numbered list (1-100), grouped by
     dimensions, unambiguous and solvable.
```

E.4 T2-Generation: Automated Judge Prompt

T2-Generation: Automated Judge Prompt

```
You are a QA Specialist for Draw.io diagrams.
Your task is to verify if a given Draw.io
     diagram satisfies a specific Yes/No
     question.
You are provided with TWO sources of
     information:
1. The Diagram Image: A visual rendering of
     the diagram.
2. The XML Code: The underlying source code
     of the diagram.
Rules:
1. Primary Source - The Image: Generally,
     prioritize the visual information
     provided by the image. Use it to judge
     existence, shape, color, relative
     positions, connections, and overall
     layout.
   - Do not rely on the XML for visual
       attributes that can be directly
       observed (e.g., "Is the box red?",
       "Is A above B?", "Is there a
       circle?"). The XML might contain
       hidden or overridden attributes that
       do not reflect the final render.
2. Secondary Source - The XML (ONLY for
     Specific Metrics): Consult the XML code
     ONLY when the question requires
     verifying precise numerical values or
     hidden properties that cannot be
     visually estimated accurately.
   - Specific Metrics: Rotation angles (e.g.,
       "rotated by 45 degrees"), exact
       dimensions (width, height, length),
       coordinates, distances, area,
       perimeter, specific style parameter
       values, or metadata.
   - Example: For "Is the width 100px?",
       visually you cannot measure pixels,
       so you MUST check the XML
       `width="100"` attribute.
   - Example: For "Is it rotated 45
       degrees?", check the XML
       `rotation="45"` style.
3. Conflict Resolution:
   - If the question is about visual
       appearance (e.g., color, shape),
       trust the Image.
   - If the question is about precise metrics
       (e.g., size, angle), trust the XML.
4. Answer Format: You must output a JSON
     object with a single key "answer".
   - If the condition is met: {"answer":
       "YES"}
   - If not met: {"answer": "NO"}
5. Ignore Irrelevant Details: Focus only on
     what the question asks.

Does the diagram satisfy the question?
     Respond with JSON.
```

E.5 T2-QA: Answering Prompt

T2-QA: Answering Prompt

```
You are a diagram analysis assistant. Answer
     the question based on the Draw.io
     mxGraph XML.

[Diagram XML]
{drawio_code}
[Context Rules]
1. "element X" refers to the element with
     value "X".
2. Containment != Intersection.
```

```
3. Adjacency != Separation.
[Question]
{question}
[Options]
{options_text}
[Strict Output Instructions]
1. Analyze the topology carefully.
2. Choose one letter (A, B, C, or D).
3. Output ONLY the letter. Do not include
    any reasoning, headers, or punctuation.
```

E.6 T2-QA: Draw.io Generation Prompt (Topology)

T2-QA: Draw.io Generation Prompt (Topology)

```
Generate a complete Draw.io XML file with at
    least 4 elements demonstrating
    containment, adjacency, separation, and
    intersection. No restrictions on shape
    or color. Use this example as a
    reference and create a randomized result:
<mxGraphModel dx="723" dy="719" grid="1"
    gridSize="10" guides="1" tooltips="1"
    connect="1" arrows="1" fold="1" page="1"
    pageScale="1" pageWidth="850"
    pageHeight="1100" background="#ffffff"
    math="0" shadow="0">
Requirements: No comments. Set value
    attributes sequentially (1, 2, 3...).
    Diagram id and mxfile etag must be empty
    strings. Use inline CSS styles instead
    of HTML attributes. Vertices must
    include vertex="1". Output full skeleton
    <mxfile><diagram><mxGraphModel><root>
    and close all tags. Output only XML. All
    geometry must be in <mxGeometry>.
```

E.7 T2-QA: QA Pair Generation Prompt (Topology)

T2-QA: QA Pair Generation Prompt (Topology)

```
# Task
Based on the given DrawIO code and the
    topology relationships (separation,
    intersection, adjacency, and
    containment), generate questions asking
    about the number of elements under each
    topology relationship condition. For
    example, you might ask:
- "If element '1' (the large gray rectangle)
    is moved vertically downwards by 21
    units, how many elements will be
    geometrically contained within it?"
- For the separation category, generate
    questions related to how many elements
    will remain separated after
    transformations. You can refer the
    example but do not copy it and you
    should become creative and random.

# DrawIO Code
```xml
[INSERT_DRAWIO_XML_CODE_HERE]
```

Reference Questions (Optional)
[INSERT_FEW_SHOT_EXAMPLES_HERE]
(e.g., Question: ..., Options: ..., Answer:
    ...)

Requirements
1. Keep Core Knowledge Consistent: The
    question should test similar topology
    relationship knowledge as the reference
    questions (e.g., disjoint, contained,
    overlapping, intersecting, etc.)
2. Moderately Increase Difficulty: Add one
    of the following complexities to the
    original question:
  - Multi-step reasoning (e.g., "what
      happens after
      moving/rotating/scaling")
  - Quantitative calculations (e.g.,
      distance, area, percentage)
  - Multiple constraint conditions
  - Prediction or optimization problems
3. Don't Deviate from the Theme:
  - Don't introduce physical laws (except
      basic geometric motion)
  - Don't introduce overly specialized
      concepts
  - Stay within topology relationships and
      position reasoning
4. Must Include Explanation Field:
   - Mark difficulty features (e.g., "[x]
       Multi-step motion tracking")
  - Detailed calculation steps
  - Clear reasoning process

Output Format (JSON)
Please strictly follow this JSON format,
    without any other text:
{
  "Question": "Question content in
      English...",
  "Option": [
    "A. Option A in English...",
    "B. Option B in English...",
    "C. Option C in English...",
    "D. Option D in English..."
  ],
  "Answer": "A",

}

Important Notes
- Question must be in English: Complete
    question sentence in English
- Options must be in English: All 4 options
    (A/B/C/D) in English
- Answer must be one of A/B/C/D
- Do not add any explanation text outside
    the JSON
```

```
Please generate the question:
```

E.8 T3-Generation:Instruction Generation Prompt

T3-Generation:Instruction Generation Prompt

```
# Role
You are a professional Draw.io
    (diagrams.net) XML Reverse Engineering
    Expert. Your task is to read a piece of
    Draw.io XML code and "translate" it into
    a structurally clear,
    instruction-explicit natural language
    task description.

# Goal
This task description will be used to guide
    another AI to generate XML code.
    Therefore, your description must
    accurately capture the diagram's
    content, logical structure, visual
    style, and connectivity.
The specific chart type is: {CHART_TYPE}.
    Please strictly use this type in your
    output.

# Analysis Steps (Chain of Thought)
Before generating the description, please
    perform the following analysis:
1. Extract Core Entities (Vertices):
   - Extract text from the `value` attribute
       (ignore HTML tags, keep only core
       text).
   - Analyze `style` attributes: Shape
       (`shape`), Fill Color (`fillColor`),
       Border Color (`strokeColor`), and
       Special Styles (e.g., `swimlane`).
2. Analyze Connections (Edges):
   - Determine flow direction based on
       `source` and `target` IDs.
   - Analyze text on the connection lines
       (if any).
   - Analyze line styles
       (Solid/Dashed/Orthogonal/Curved).
3. Infer Layout Logic:
   - Observe `x` and `y` coordinates. Are
       elements arranged Top-to-Bottom?
       Left-to-Right? Or is there a
       nesting/container relationship?

# Output Format
Please generate the task description
    following the structure below (do not
    output the XML analysis process, only
    the description text):

---
Chart Type: {CHART_TYPE}

Overall Layout: [Describe how elements are
    arranged, e.g., All elements are
    arranged sequentially from top to bottom]

Visual Style:
- [Describe general color or shape styles,
    e.g., Nodes mainly use rounded
    rectangles, white background, key nodes
    are red]

Node Content & Details:
1. [Node ID or Name]: [Description]
   - Shape: [e.g., Rhombus/Rectangle]
   - Color: [If there is a specific color,
       please specify, e.g., Red background]
   - Text Content: "[Put text here]"

2. ... (List all main nodes)

Connections:
- [Node A] -> [Node B]: [Describe arrow
    direction, mention if there is text on
    the line]
- [Node C] -> [Node D]
---

# Rules & Constraints
1. Ignore Metadata: Do not mention technical
    details like `<mxfile host...>` or
    `id="0"`.
2. Color Translation: If `fillColor` is a
    Hex code (e.g., #FF0000), try to
    translate it into natural language
    (e.g., "Red"). If it is a complex custom
    color, keep the Hex code.
3. Container Recognition: If the XML
    contains `swimlane` or `group`, you must
    explicitly describe this containment
    relationship (e.g., "There is a
    container named 'Student', which
    contains... attributes").
4. Simplify: If multiple nodes share the
    exact same style, you can combine the
    description (e.g., "Create 3 rectangle
    nodes, with text A, B, and C
    respectively").
```

E.9 T3 Data Construction: Word Cloud Keyword Extraction Prompt

T3-Generation: VLM Judge Prompt (Class Diagram)

```
Based on the provided diagram image,
    extract and list the most important
    keywords and concepts that represent
    the core elements of this
    system/process.

Requirements:
- Identify 15-25 key terms from the diagram
- Focus on technical terms, component
    names, and important concepts
```

```
- Include both nouns (components, tools)
    and verbs (actions, processes)
- Weight terms by importance (you can
    indicate this with frequency)
- Output as a simple comma-separated list

Format: keyword1, keyword2, keyword3, ...
```

E.10 T3 Data Construction: Diagram Description Generation Prompt

T3-Generation: VLM Judge Prompt (Class Diagram)

```
Analyze the provided diagram image and
    generate a comprehensive description of
    the system or process it represents.

Requirements:
- Provide a clear overview of what this
    diagram illustrates (2-3 sentences)
- Explain the main workflow or process flow
    (step-by-step if applicable)
- Identify key components, tools, or
    technologies involved
- Describe the relationships and data flow
    between components
- Highlight any important inputs, outputs,
    or transformations
- Keep the tone professional and technical
- Length: 200-300 words

Structure:
1. Overview
2. Main Process/Workflow
3. Key Components
4. Purpose/Outcome
```

E.11 T3 Data Construction: Mermaid Diagram Generation Prompt

T3-Generation: VLM Judge Prompt (Class Diagram)

```
Based on the provided diagram image,
    recreate it as a Mermaid diagram using
    appropriate Mermaid syntax.

Requirements:
- Analyze the diagram type (flowchart,
    sequence diagram, graph, etc.)
- Use the most suitable Mermaid diagram type
- Preserve all nodes, components, and
    connections shown in the image
- Maintain the logical flow and
    relationships
- Include all labels and text annotations
- Use clear and descriptive node IDs
- Add styling if the original has visual
    distinctions (colors, shapes)

Output only the Mermaid code block,
    starting with:
```mermaid
[diagram type]
[diagram content]
```

Choose from these common Mermaid types
    based on the image:
- graph TD/LR (flowchart
    top-down/left-right)
- sequenceDiagram
- classDiagram
- stateDiagram-v2
- journey
```
```

E.12 T3-Generation: VLM Judge Prompt (Class Diagram)

T3-Generation: VLM Judge Prompt (Class Diagram)

```
# Role
You are a senior UML diagram visual
    reviewer. Your task is to compare two
    UML class diagrams:
1. Ground Truth Image (Standard Answer):
    The correct reference diagram.
2. Generated Image (Test Diagram): The
    AI-generated diagram.

# Input Data
1. Ground Truth Image (Standard Reference -
    GT)
2. Generated Image (Test Generation - Gen)

# Evaluation Goal
Please ignore font styles, background
    colors, or pixel-level alignment
    differences. Focus on evaluating
    whether the semantic logic and visual
    specifications are consistent with the
    Ground Truth (GT).

# Scoring Criteria (Total 100 Points)

Please score based on visual comparison:

## 1. Entity & Content Completeness
    (Content Completeness) - 40 Points
*Check text content (OCR capability test)*
* Class/Interface Existence: Do all classes
    and interfaces in GT appear in the
    generated diagram? (Deduct 10 points
    for each omission)
* Text Accuracy: Are the spellings of class
    names, attributes (fields), and method
    names exactly the same? Are visibility
    symbols (+, -, #) missing? (Deduct 2
    points for each error)
* Type Markers: Do interfaces have
    <interface> or <<interface>> visual
    markers?
```
```

```
## 2. Relationship Semantic Consistency
    (Relationship Semantics) - 40 Points
*This is the most critical part of UML
    evaluation, please carefully
    distinguish the shapes of line
    terminals*
* Line Type: Solid vs. Dashed. If GT is
    dashed (e.g., implementation,
    dependency) and the generated diagram
    draws it as solid, it is considered a
    serious error. (Deduct 5 points for
    each instance)
* Arrow Shape: Please strictly distinguish
    the following styles by visual features:
   * Inheritance: Empty Triangle
   * Association: Open Arrow / No Arrow
   * Aggregation: Empty Diamond
   * Composition: Filled Diamond
   * Criterion: If the arrow shape in the
       generated diagram is inconsistent
       with GT (e.g., drawing "Filled
       Diamond" as "Empty Diamond"), this
       is a serious semantic error.
       (Deduct 10 points for each instance)
* Direction: Is the direction of the arrow
    consistent with GT? (Deduct 10 points
    for reversed direction)

## 3. Layout & Visual Structure (Layout &
    Structure) - 20 Points
*Evaluate the readability of the diagram*
* Topological Structure: Although
    coordinates do not need to be exactly
    the same, the relative positional
    relationship should be reasonable. For
    example, if the Parent class is at the
    top in GT, it should not be at the
    bottom in the generated diagram.
* Occlusion & Overlap: Are there connecting
    lines crossing text or boxes covering
    each other, making them unreadable?
    (Deduct 20 points for serious occlusion)
* Line Tidiness: Are connecting lines
    correctly connected to the edges of
    boxes, rather than hanging in the air
    or inserted inside boxes?

# Output Format (JSON)

Please output the following JSON format:

```json
{
  "score": {
    "content_completeness": 0,
    "relationship_semantics": 0,
    "layout_structure": 0,
    "total": 0
  },
  "audit_report": {
    "missing_elements": ["List missing
        classes or methods, enter None if
        none"],
    "relationship_errors": ["List specific
        class pairs with wrong arrow shapes
        or line types, e.g.: 'User->Order
        should be Composition (Filled
        Diamond), but generated as
        Aggregation (Empty Diamond)'"],
    "visual_defects": "Briefly describe
        occlusion, overlap, or text
        garbling issues"
  },
  "verdict": "Production Ready / Needs
      Correction / Unusable"
}
```
```