MXSRBENCH: TOWARDS SPATIAL UNDERSTANDING AND DIAGRAMMATIC REASONING IN mxGraph FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

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 3,990 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.

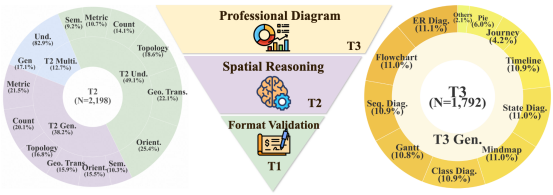

Figure 1: Distribution of mxSRBench across T2 (N=2,198) and T3 (N=1,792) tracks, spanning 6 dimensions and 11 diagram types.

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

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

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**

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 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 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 3,990 instruction–XML pairs and a three-tier suite (Figure 1): **T1** validates XML renderability; **T2** probes spatial reasoning across six dimensions via three sub-tasks—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 3,990-sample `mxGraph` benchmark for diagrammatic spatial reasoning with explicit symbolic and geometric structure.

[1]Datasets are available at [link].

- **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.

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`	**3,990**	✓	✓	✓	**6**	✓

2 Related Work

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

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 & 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.

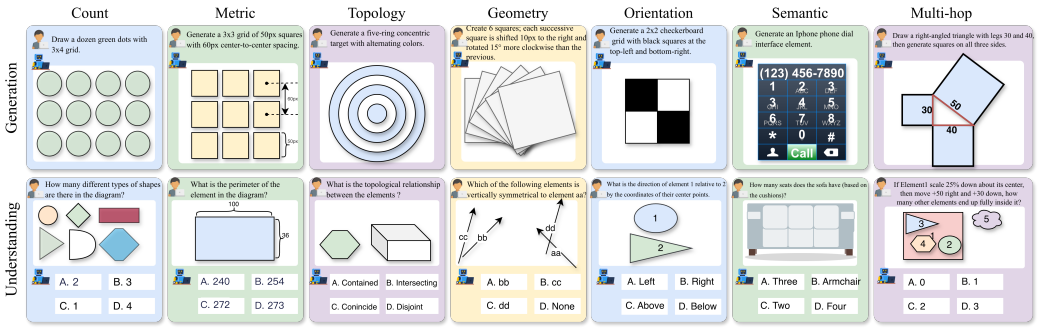

Figure 2: T2 task demos in mxSRBench. **Understanding** and **Generation** examples are shown across six spatial dimensions: Count, Metric, Topology, Geometry Transformation (Geo Trans), Orientation, and Semantic.

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, SVGEdit-Bench (Nishina & Matsui, 2024; 2025) targets editing capabilities, Image-Text Bridging (Cai et al., 2025) probes cross-modal understanding, and SVG-Taxonomy (Xu & Wall, 2024) examines low-level visual tasks. Generation methods include StarVector (Rodriguez et al., 2025), OmniSVG (Yang et al., 2026), SVGBuilder (Chen & 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 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

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.

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.

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

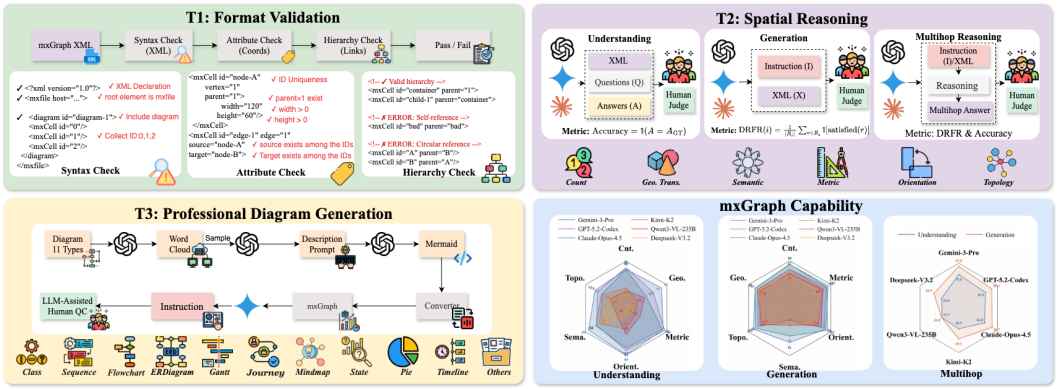

Figure 3: Overview of the mxSRBench evaluation tiers. Our benchmark spans from low-level Format Validation (**T1**) to multi-dimensional Spatial Reasoning (**T2**) and high-level Professional Diagram Synthesis (**T3**). The radar charts illustrate the performance gaps among representative models across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

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 1,792 samples covering 11 professional diagram types (Table 13). Since professional diagrams involve rich, type-specific conventions that are difficult to exhaustively decompose into atomic checklists, we evaluate T3 using a rubric-based visual comparison protocol implemented with a VLM-based judge. Specifically, we render both the ground-truth and model-generated `mxGraph` XML into images and compare them with respect to semantic content, relational structure, spatial organization, readability, and diagram-specific conventions. We refer to the resulting score, reported on a 0–100 scale, as the *Diagram Synthesis Score (DSS)*.

3.2 MXSRBENCH DATASET CONSTRUCTION

mxSRBench contains 3,990 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 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**, the reference diagrams are generated through a structured Mermaid→`mxGraph` XML rendering pipeline, which reduces free-form generation errors but does not completely eliminate instruction–XML mismatches. We therefore perform an independent multimodal alignment screening using multiple judges, each conditioned on the

task instruction, the complete mxGraph XML, and its rendered diagram. Samples receiving a low-alignment decision from multiple judges are flagged for human review rather than being automatically discarded. Confirmed mismatches are repaired when possible and otherwise excluded from the dataset. The screening criteria, per-category statistics, and sample disposition are reported in Appendix C.3.

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 $(x + \frac{w}{2},\ y + \frac{h}{2})$.

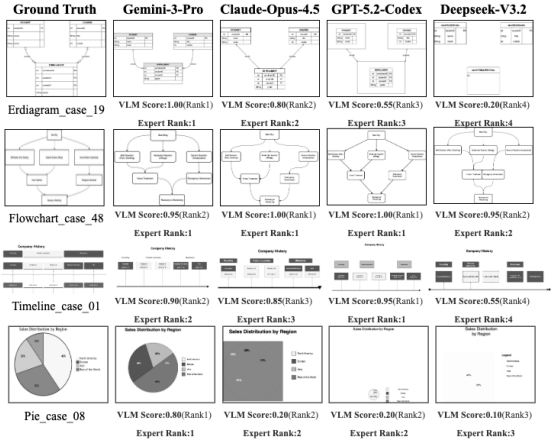

Figure 4: T3 generation examples with DSS values and expert ranks. The strong rank correlation validates our evaluation protocol.

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; DSS) 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 similarity/quality score (%, reported as DSS), 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 and Gemini-3-Flash (Google DeepMind, 2025), Claude-Opus-4.5 and Claude-Sonnet-4.5 (Anthropic, 2025), GPT-5.2 and GPT-5.2-Codex (OpenAI, 2025)) and open-source (DeepSeek-V3.2 (DeepSeek-AI, 2025), GLM-4.6, Qwen3-VL-235B-A22B-Instruct and Qwen3-Coder-480B-A35B-Instruct (Qwen Team, 2025), Kimi-K2-Instruct-0905 (Moonshot AI, 2025), MiniMax-M2 (MiniMax-AI, 2025)).

4.2 MAIN RESULTS

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).

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 6). 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 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

Cross-benchmark Correlation Table 4 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.

Table 4: 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 & Math-AI, 2024)	0.105	0.061	0.746

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.

Table 5: LLM performance on professional diagram generation (**T3**). Diagram Synthesis Scores (DSS, %) 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	92.66	**92.49**	94.18	**68.81**	58.45	**84.57**	**75.00**	48.57	**64.92**	96.25	**88.63**	**78.59**
	Gemini-3-Flash	**95.98**	81.75	**95.34**	67.74	57.12	83.08	73.23	**54.68**	57.82	**97.06**	87.38	77.38
	Claude-Opus-4.5	96.17	74.40	93.73	68.76	**61.22**	84.38	73.27	45.74	62.61	96.73	88.08	76.83
	GPT-5.2-Codex	83.19	72.73	92.89	61.93	57.20	75.80	62.95	18.50	39.95	89.20	85.29	67.24
	GPT-5.2	85.87	60.37	92.61	63.92	59.78	67.53	60.68	17.96	16.98	86.59	80.58	62.99
Open	Kimi-K2	**87.47**	**61.80**	**92.34**	49.10	40.82	65.82	54.75	**21.89**	**37.67**	89.63	76.92	61.65
	GLM-4.6	82.49	60.26	91.97	**62.41**	**53.61**	**78.82**	58.32	15.69	31.86	**91.05**	**80.33**	**64.26**
	DeepSeek-V3.2	85.19	58.69	91.72	48.47	36.26	57.16	**61.61**	16.08	25.87	83.21	71.75	57.82
	Qwen3-VL-235B	52.24	13.96	29.26	5.62	0.44	28.86	24.15	5.19	20.62	37.00	12.94	20.93

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

Professional Diagram Generation Model performance across diagram types reveals three levels of complexity. Table 5 presents detailed DSS values 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 (16.9%), 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 5 visualizes this hierarchy across models and diagram types. Figure 4 shows representative T3 generation demos with their DSS values and expert ranks. To assess the robustness of our findings to the choice of evaluator, we conduct a swap-judge analysis using GPT-5.4 and Claude Opus 4.5 as alternative judges. The primary evaluation uses Gemini-3-Flash, and the image-based evaluation protocol is kept unchanged. At the model level, the Spearman correlations between the alternative-judge rankings and the primary ranking are 1.000 for GPT-5.4 and 0.939 for Claude Opus 4.5, respectively. These results indicate that the overall T3 model ranking is robust to evaluator choice, suggesting that the main conclusions are not substantially affected by the selection of a single evaluator. Detailed per-category correlations are reported in Appendix B.3.

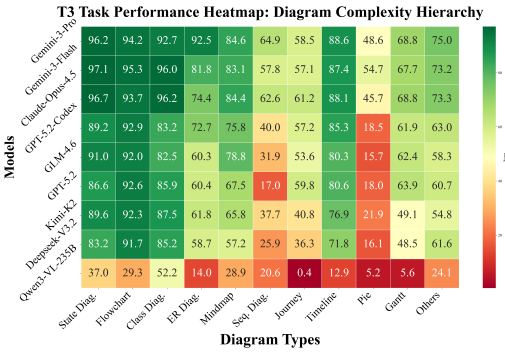

Figure 5: T3 performance heatmap across diagram types.

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 10), 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 9). 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 3,990 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.

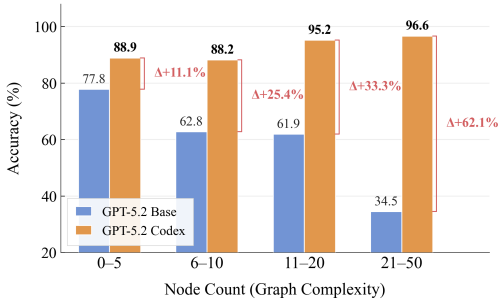

Figure 6: Performance comparison between GPT-5.2 and GPT-5.2-Codex across varying graph complexities (node counts).

AI use statement

In this work, we used generative AI tools for generating the synthetic benchmark data (diagrams, QA pairs, and task instructions in `mxGraph` XML, Appendix C) and for parts of the evaluation methodology, including an LLM-based judge used to score the reported results. We did not use generative AI tools for proposing or refining hypotheses, formulating or proving mathematical claims, or developing the conceptual framework, and these tasks are not applicable to this work. We additionally used LLMs for drafting parts of the manuscript and language polishing. We have reviewed AI-assisted work through manual verification of generated samples and evaluation scores, and we take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Reproducibility Statement

We describe the construction of the benchmark, including the rule-engine and LLM-assisted synthesis pipelines, quality-control protocols, and final sample disposition, in Section 3 and Appendix C. Task definitions for T1–T3 and the full evaluation protocols (Accuracy, DRFR, and the DSS visual-judge procedure, including human-annotation agreement) are specified in Section 3 and Appendix C. Experimental settings, including model versions, decoding configuration (temperature 0), and prompts used for generation and evaluation, are provided in Section 4 and Appendix E. An anonymized copy of the source code, benchmark data, and evaluation toolkit is available at the anonymous link provided in the footnote of the Conclusion (Section 5).

[2] Dataset is available at [link]

REFERENCES

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

Anthropic. Claude 4.5 system card. Technical report, Anthropic, July 2025. URL https://assets.anthropic.com/m/12f214efcc2f457a/original/Claude-Sonnet-4-5-System-Card.pdf. 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. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pp. 2425–2433, 2015.

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

Mu Cai, Zeyi Huang, Yuheng Li, Utkarsh Ojha, Haohan Wang, and Yong Jae Lee. 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)*, pp. 5377–5386. IEEE, 2025.

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

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

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

Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric Xing, and Liang Lin. Geoqa: A geometric question answering benchmark towards multimodal numerical reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 513–523, 2021a.

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

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

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

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

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

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

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

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, December 2025. URL https://arxiv.org/abs/2512.02556.

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

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

Google DeepMind. A new era of intelligence with gemini 3. Technical report, Google DeepMind, 2025. URL https://blog.google/products-and-platforms/products/gemini/gemini-3/. 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, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

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

Kaiyi Huang, Chengqi Duan, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. 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, 2025.

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

Amita Kamath, Jack Hessel, and Kai-Wei Chang. 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*, pp. 9161–9175, 2023.

Fangjun Li, David C Hogg, and Anthony G Cohn. 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, pp. 18500–18507, 2024.

Jinke Li, Jiarui Yu, Chenxing Wei, Hande Dong, Qiang Lin, Liangjing Yang, Zhicai Wang, and Yanbin Hao. 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*, pp. 13156–13163, 2025.

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

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

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. 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)*, pp. 6774–6786, 2021.

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

MiniMax-AI. Minimax-m2: Efficient agentic intelligence, 2025. URL https://huggingface.co/MiniMaxAI/MiniMax-M2. Open Source Release for Agentic Coding Workflows.

Moonshot AI. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025. URL https://arxiv.org/abs/2507.20534.

Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. 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, pp. 4296–4304, 2024.

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

Kunato Nishina and Yusuke Matsui. 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*, pp. 8142–8147, 2024.

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

OpenAI. Introducing gpt-5.2: Specialist-level reasoning and reliability, December 2025. URL https://openai.com/index/introducing-gpt-5-2/. 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, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

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

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

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

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

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

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

Md Imbesat Rizvi, Xiaodan Zhu, and Iryna Gurevych. 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)*, pp. 4750–4767, 2024.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Yuxiang Zhou, Jiazheng Li, Yanzheng Xiang, Hanqi Yan, Lin Gui, and Yulan He. 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*, pp. 14365–14378, 2024.

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

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

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.

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%

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%

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 9 provides detailed agreement statistics across all spatial dimensions.

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: 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**

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).

B.3 Judge Bias Analysis

We provide a per-category analysis of evaluator sensitivity for T3. For each alternative judge, we sample 20 cases per model and diagram type. For each diagram type, we first average each model's scores over the available matched cases and then compute Spearman's rank correlation between the

Table 10: Per-category model-ranking Spearman correlations between the primary Gemini-3-Flash evaluation and the alternative T3 judges. Values are rounded to three decimal places.

Diagram Type	GPT-5.4	Claude Opus 4.5
Class Diagram	0.933	0.936
ER Diagram	0.933	0.952
Flowchart	0.750	0.818
Gantt	0.717	0.952
Journey	0.745	0.815
Mindmap	0.967	1.000
Others	0.867	0.997
Pie	0.983	0.976
Sequence Diagram	1.000	0.988
State Diagram	0.900	0.879
Timeline	0.979	0.985

model rankings produced by the primary Gemini-3-Flash judge and each alternative judge. The resulting per-category correlations are reported in Table 10.

B.4 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 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

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 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 12).

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

Table 12: **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.

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.

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 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)

Table 13: **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

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 QUALITY CONTROL.

Because the T3 reference diagrams are produced through a structured Mermaid→`mxGraph` XML→PNG pipeline, the main remaining quality risk is a mismatch between the regenerated natural-language instruction and the corresponding diagram. We therefore perform an independent multimodal alignment screening. Each judge is provided with the task instruction, the complete `mxGraph` XML, and the rendered diagram, and is asked to determine whether the diagram satisfies the instruction with respect to semantic content, relations, diagram-type conventions, and structural consistency.

Cases receiving a low-alignment decision from multiple judges are treated as candidate quality issues and are sent for human review. The review is conducted by three annotators, including one undergraduate student and two PhD students, with backgrounds in computer science, artificial intelligence, visual reasoning, and structured-diagram construction. To keep the manual stage consistent with the automatic screening, annotators apply the same four-dimension rubric and decision categories defined in the screening prompt (Appendix E.7) and classify each case as aligned, correctable, or to be excluded. Correctable cases are locally repaired and re-verified before being retained; cases whose instruction–XML mismatch cannot be reliably resolved are excluded. After this joint LLM–human screening, 91 samples were removed from the initial 1,883-case pool, yielding the final T3 evaluation set of 1,792 samples used throughout the paper.

C.3.4 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 DSS in %), which we aggregate across categories and also report per-category performance.

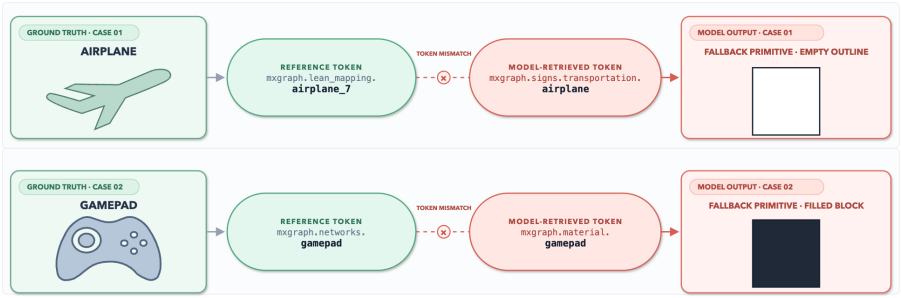

Figure 7: Example of vocabulary grounding errors. The model fails to retrieve correct mxgraph shapes for semantic entities (airplane and gamepad), resulting in fallback to default geometric primitives.

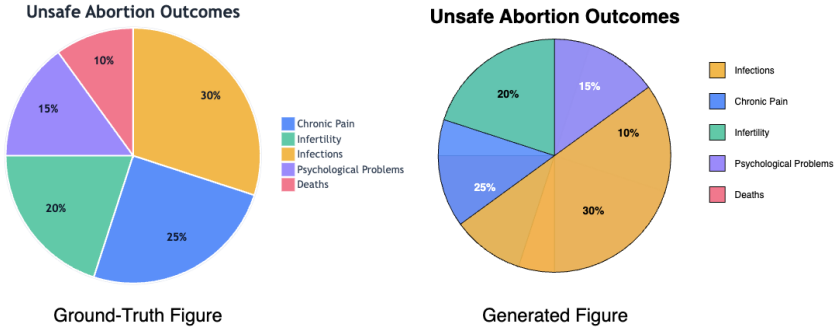

Figure 8: Example of integrity grounding errors. The model omits fine-grained parameters (e.g., `startAngle`) in pie chart generation, disrupting the intended visual layout despite correct high-level intent.

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. 10, Fig. 9).

E Prompts

E.1 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) for automated evaluation.
Rules:

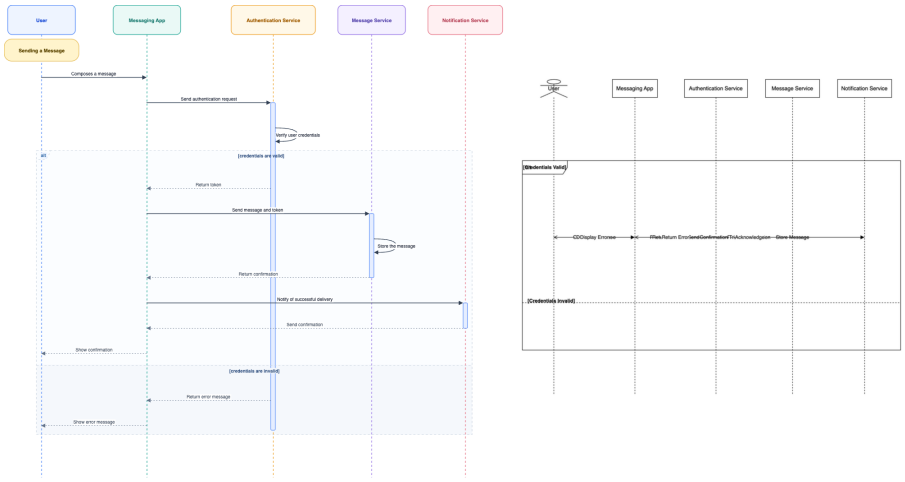

Figure 9: Example of constraint propagation errors in sequence diagram generation. The model exhibits message collisions, loss of temporal ordering, and systematic omission of critical UML elements (activation bars, return messages).

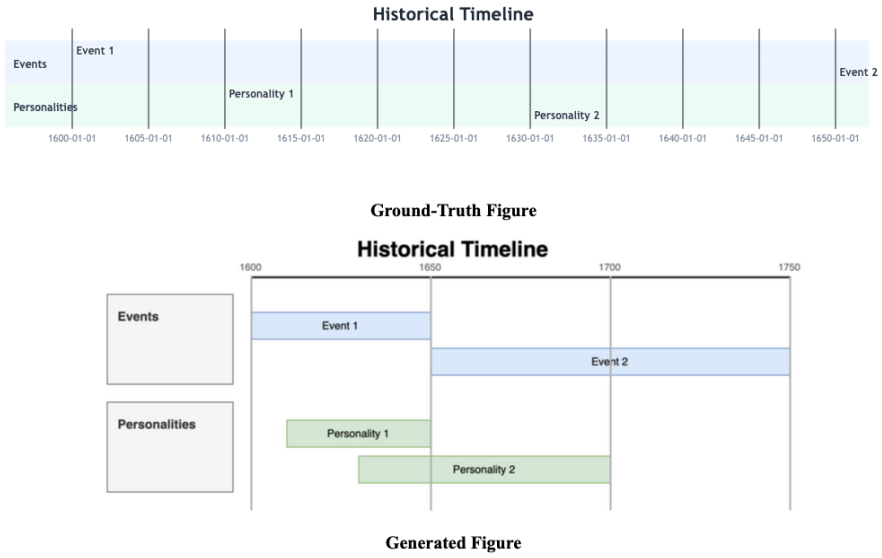

Figure 10: Example of constraint propagation errors in Gantt chart generation. The model produces incorrect time axes and converts milestones into duration bars, violating scheduling semantics.

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)

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 `<`, `>` as `>`, `"` as `"`, and `&` as `&`.

– For simple text (newlines), use `
`. For complex formatting, the `style` must include `html=1`, but HTML tags inside `value` must be escaped (e.g., `Text`).
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)

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-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.5 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.6 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"
}
```

E.7 T3 Data Quality Control: Alignment Screening Judge Prompt

```
You are an expert auditor for the quality of a professional diagram
generation benchmark.

Your task is to audit whether a natural-language diagram specification is
faithfully implemented by the provided Draw.io mxGraph XML and whether the
resulting rendered PNG correctly exposes the required information.

This is a GOLD-DATA QUALITY AUDIT. You are not evaluating a model prediction
against a reference image, and you are not judging artistic quality. The
question is whether the instruction--XML pair is reliable enough to serve as
a benchmark item.

You will receive:

1. The diagram type.
2. A natural-language task description.
3. The complete Draw.io mxGraph XML.
4. A PNG rendered from the XML.

The natural-language task description is the specification.
The mxGraph XML is the proposed implementation.
The PNG is the final rendered evidence.

Do not treat metadata/category inconsistency as an instruction--XML
alignment failure. Report it separately as `metadata_consistency`.

Use the XML as the primary evidence for:
- required entities and labels;
- attributes, methods, values, dates, and percentages;
- edges, directions, dependencies, containment, and hierarchy;
- diagram-specific notation;
- coordinates, dimensions, and style parameters.

Use the PNG as complementary evidence for:
- whether required text is visible;
- label clipping, overlap, and occlusion;
- whether nodes and edges are actually rendered;
- readability and visual usability;
- discrepancies between XML and the final rendered output.
```

Do not compare the diagram against an assumed canonical layout.
Do not penalize a valid alternative layout unless the task description
explicitly requires a particular position, ordering, alignment, grouping,
proportion, size, date, or style.

Do not penalize an XML style property merely because it differs from a
default Draw.io style. For example, do not treat `curved=1` as an error when
the instruction does not prohibit curved connectors and the rendered diagram
still satisfies the required relation.

−−
Step 1: Check specification completeness
−−

Before scoring the XML, determine whether the task description is sufficiently
complete and internally consistent for an alignment audit.

Treat the specification as incomplete or ambiguous if:
− the text is visibly truncated;
− a numbered requirement starts but its details are missing;
− the instruction refers to entities or constraints that are never defined;
− the instruction contains contradictory positions, relations, values, or
  diagram types;
− essential requirements cannot be determined from the available text.

Do not infer missing requirements from the XML or PNG.
Do not give credit for content that appears in the XML but is not specified in
the task description.

If the specification is incomplete, truncated, or internally contradictory,
record an `ambiguous_instruction` issue with severity `major` and set the final
decision to `needs_review`, regardless of the numerical average.

−−
Step 2: Extract explicit requirements
−−

Construct a concise internal checklist of the requirements explicitly stated
or unambiguously implied by the task description.

The checklist should cover, when applicable:
− required nodes, labels, attributes, methods, values, dates, and percentages;
− required edges, directions, labels, ordering, dependencies, and hierarchy;
− required diagram type and diagram−specific notation;
− explicit spatial, temporal, geometric, numerical, or layout constraints;
− explicit visibility and rendering requirements.

Each requirement must be classified internally as:
− satisfied;
− violated;
− uncertain;
− not verifiable because the specification is incomplete.

Do not silently convert an uncertain requirement into a satisfied one.

Step 3: Evaluate the four dimensions

Score each dimension from 0 to 100.

1. content_coverage

Evaluate whether every explicitly required entity, node, label, attribute, method, value, date, percentage, and other content is present and correct.

Scoring guidance:
– 100: all explicit content requirements are satisfied;
– 90−−99: only harmless, non−semantic formatting differences;
– 75−−89: one or more non−central content issues or uncertain details;
– 60−−74: an important required item is missing, altered, or unverifiable;
– 0−−59: core content is missing or substantially incorrect.

A required missing label or value is not a cosmetic issue.

2. relation_logic

Evaluate whether all required edges and semantic relations are correct, including direction, source and target, ordering, dependencies, containment, hierarchy, cardinality, and sequence.

Scoring guidance:
– 100: all required relations are present and semantically correct;
– 90−−99: only harmless routing or connector−style differences;
– 75−−89: a localized non−central relation issue;
– 60−−74: an important relation is uncertain or incorrectly represented;
– 0−−59: a core relation, direction, dependency, or hierarchy is wrong.

A wrong direction, missing core edge, wrong cardinality, or wrong parent−child relation is a substantive issue, not a minor style issue.

3. diagram_convention

Evaluate only diagram conventions explicitly required by the task description or unambiguously implied by the stated diagram type.

Examples include:
– UML class compartments and generalization symbols;
– ER cardinality notation;
– sequence lifelines and message order;
– state transition notation;
– Gantt dependencies and milestones;
– mindmap or timeline hierarchy.

Do not penalize valid alternative representations when the instruction does not specify a particular notation.

A required but incorrect symbol or diagram convention should normally be treated as a major issue if it changes the intended semantics.

4. constraint_layout

Evaluate explicit spatial, temporal, numerical, geometric, proportional, grouping, and readability constraints.

If the task description does not specify a particular layout constraint, do not invent one and do not penalize a valid alternative arrangement.

A layout issue is substantive when it causes:
– a required node to be in the wrong relative position;
– a required ordering or hierarchy to be unclear;
– an explicitly required proportion, date, or numerical relation to be wrong;
– required labels or relations to be hidden, clipped, or unreadable;
– severe overlap or rendering failure.

Small differences in spacing, centering, line curvature, or box width should not lower the score when they do not affect the stated requirements.

––
Step 4: Apply hard decision rules
––

The arithmetic mean of the four dimension scores is the descriptive `alignment_score`, but it must not override critical issues.

Apply these rules:

1. If the instruction is truncated, incomplete, or contradictory:
   – include an `ambiguous_instruction` issue with severity `major`;
   – set `decision` to `needs_review`;
   – do not set `decision` to `aligned`.

2. If a required core node, relation, direction, value, date, cardinality, hierarchy, or diagram convention is missing or wrong:
   – include a major issue;
   – set `decision` to `needs_review` or `misaligned`;
   – do not classify the item as fully aligned.

3. If a required item is present in XML but not visible or readable in PNG:
   – include a `rendering` issue;
   – use severity `major` when the missing visibility affects semantics;
   – set `decision` to `needs_review` or `misaligned`.

4. A sample may be `aligned` only when:
   – the specification is complete and internally consistent;
   – all explicit semantic requirements are satisfied;
   – no major issue is present;
   – any remaining differences are purely cosmetic or layout–equivalent.

5. Use the following score interpretation:
   – 95––100: fully aligned;
   – 85––94: mostly aligned;
   – 70––84: needs review;
   – 0––69: misaligned.

```
The final decision should be:

– `aligned`: all explicit requirements are satisfied and no substantive issue
  remains;
– `mostly_aligned`: only minor non–semantic deviations remain;
– `needs_review`: the specification is incomplete or a potential semantic or
  rendering problem requires human verification;
– `misaligned`: one or more core requirements are not implemented.

————————————————————————————————————————————————
Output format
————————————————————————————————————————————————

Return exactly one JSON object and no Markdown.

{
  "sample_id": "string",
  "diagram_type": "string",
  "content_coverage": 0,
  "relation_logic": 0,
  "diagram_convention": 0,
  "constraint_layout": 0,
  "alignment_score": 0,
  "decision": "aligned|mostly_aligned|needs_review|misaligned",
  "issues": [
    {
      "type": "missing_content | wrong_relation | wrong_order | wrong_hierarchy | wrong_value |
            wrong_convention | layout_constraint | rendering | ambiguous_instruction | other",
      "severity": "minor|major",
      "evidence": "A concise, concrete explanation grounded in the task description, XML, or PNG"
    }
  ],
  "brief_reason": "One concise sentence explaining the overall judgment"
}

Every score below 100 must be supported by concrete evidence.
Do not report a minor issue when it affects the semantic validity of the
benchmark item.
Do not report an issue merely because the XML uses a different but valid
layout or style.
Do not add keys.
Do not wrap the JSON in Markdown.
```