LEARNING TO UNDERSTAND SYMBOLIC DRAWINGS WITH STRUCTURE-GUIDED ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision language models (VLMs) perform well on natural images and documents but often struggle with complex symbolic drawings, such as mechanical CAD drawings and mathematical geometry diagrams. These drawings encode information through sparse geometry, numerical annotations, multiple views, and formal relations, requiring accurate object identification and annotation binding. We first study structure-assisted reasoning, where a question-independent structured description is provided alongside the image to explicitly represent geometric elements, dimensions, cross-view correspondences, and constraints. Experiments show that such information consistently improves question-answering performance, suggesting that many failures arise from unstable visual-to-structural interpretation. We further propose Structure-Guided On-Policy Distillation (SG-OPD) to transfer this benefit into an image-only model. During training, a structure-assisted teacher provides token-level supervision on trajectories generated by the image-only student. Experiments on mechanical CAD drawings and mathematical geometry diagrams show that SG-OPD consistently improves image-only baselines without requiring structured inputs at inference time. These results demonstrate that explicit structure can facilitate symbolic visual reasoning and be effectively internalized through on-policy distillation.

1 INTRODUCTION

Vision language models (VLMs) have achieved strong performance on a broad range of visual question answering (VQA) tasks, as well as natural-image understanding, document analysis, and general multimodal reasoning Radford et al. (2021); Li et al. (2025); Mathew et al. (2021); Liu et al. (2024); Li et al. (2023). This success, however, does not necessarily extend to visually complex symbolic drawings, such as mathematical diagrams, engineering schematics, and computer-aided design (CAD) drawings Zhang et al. (2024); Lu et al. (2024); Kou et al. (2026). Unlike natural images, these drawings convey information through sparse geometry, dimensions, repeated symbols, multiple views, and formal constraints. Answering a question therefore requires a model to identify the relevant object, bind annotations to that object, align corresponding elements across views, and reason over the resulting structure; even a minor perceptual or binding error can change the answer.

We use mechanical CAD drawings as a representative diagnostic setting to examine this limitation. Across multiple VLMs, we observe frequent failures even on direct-reading questions that require little additional reasoning, a pattern also reported in MechVQA Kou et al. (2026). Typical errors include confusing inner and outer diameters, assigning dimensions to the wrong features, and mixing annotations across views. Analogous difficulties arise in mathematical visual reasoning: models may fail to use diagrammatic evidence consistently or to connect visual quantities with the required symbolic reasoning, as documented by MathVerse and MathVista Zhang et al. (2024); Lu et al. (2024). The lower panel of Figure 1 illustrates a representative example: several models report incorrect inner and outer diameters for a central through-hole when given only the image and question, yet answer correctly when also provided with a structured representation of the same drawing. Taken together, these observations point to two recurring challenges: **entity–attribute and cross-view binding**, namely organizing sparse visual symbols and annotations into the correct objects and relations; and **structure-dependent reasoning**, namely using the recovered visual struc-

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.

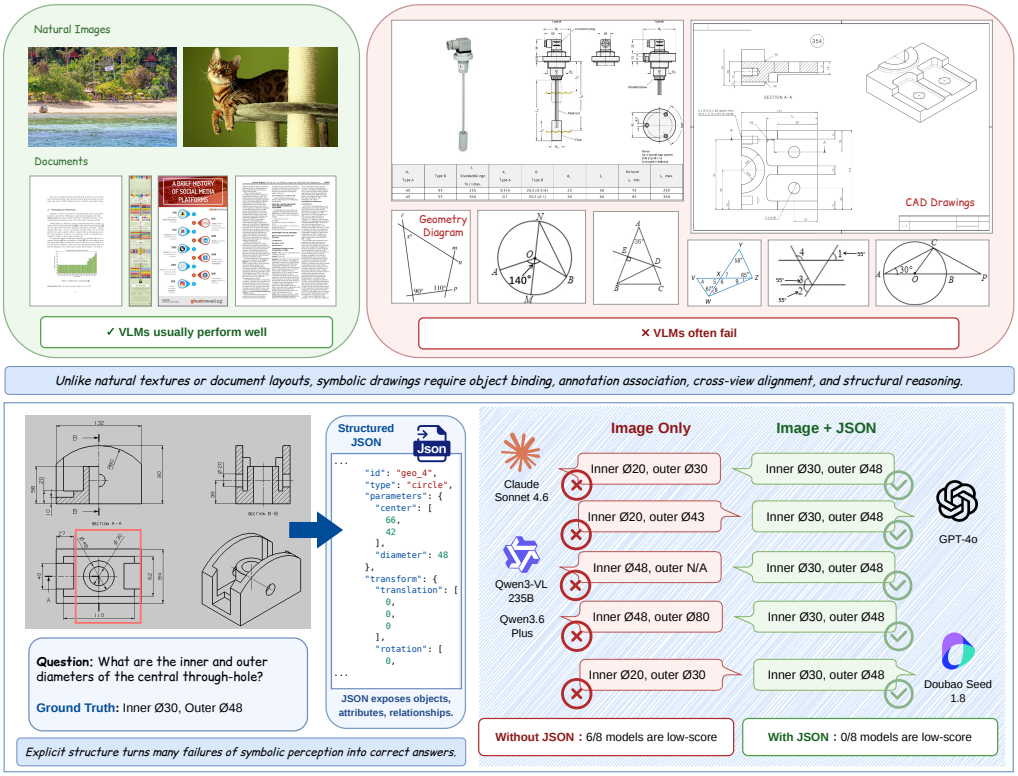

Figure 1: Overview of the domain comparison and the role of structured JSON representations in symbolic-drawing question answering. The lower panel presents a representative bad case from CAD VQA.

ture reliably in the subsequent geometric, mathematical, or engineering reasoning process. These difficulties arise before or alongside complex reasoning, so even a minor perceptual or binding error can change the answer.

These observations motivate us to first improve VLM reasoning over complex symbolic drawings by making their underlying structure explicit. We therefore study *structure-assisted reasoning*, where a question-independent structural representation is provided alongside the image to organize geometric objects, annotations, correspondences, and constraints, reducing the burden of recovering these relations directly from pixels. However, generating and supplying such structured representations at inference time introduces additional computational and deployment overhead. This motivates a second goal: transferring the benefits of explicit structure into an image-only model. To this end, we build on on-policy distillation (OPD) Agarwal et al. (2024), using structure as privileged information during training so that the model can internalize structure-assisted reasoning while requiring only the original image and question at deployment.

We therefore propose *Structure-Guided On-Policy Distillation (SG-OPD)*, an extension of OPD that uses explicit structure during training to improve an image-only model. The teacher receives the image, question, and structured representation, whereas the student receives only the image and question. At each student-generated prefix, the teacher uses the additional structure to provide token-level supervision, guiding the student at the states it actually visits. Experiments on both CAD and mathematical geometry tasks show that SG-OPD consistently outperforms image-only baselines, indicating that the benefits of explicit structure can be retained without structured inputs at deployment. Our contributions are summarized as follows:

- We identify the structural bottlenecks that cause current VLMs to struggle with complex symbolic drawings, and propose a structure-assisted reasoning framework that explicitly represents geometric objects, annotations, correspondences, and constraints.

- We propose *Structure-Guided On-Policy Distillation (SG-OPD)*, which transfers the benefits of explicit structure from a structure-assisted teacher to an image-only student.
- Experiments on mechanical CAD drawings and mathematical geometry diagrams show consistent performance gains from both training-free structure-assisted reasoning and SG-OPD.

2 Related Work

2.1 Structured Understanding of Symbolic Drawings

Symbolic drawings arise in a range of domains, such as mechanical CAD drawings, mathematical geometry diagrams, and circuit schematics. MathVista evaluates mathematical reasoning in visual contexts Lu et al. (2024), while MathVerse examines whether models actually use diagrammatic evidence under different combinations of visual and textual information Zhang et al. (2024). MechVQA benchmarks fine-grained mechanical drawing understanding and develops domain-specific models Kou et al. (2026). Beyond evaluation, Inter-GPS parses geometry problems into formal representations for symbolic solving Lu et al. (2021), and FloorplanQA studies spatial reasoning over structured layouts Rodionov et al. (2025). These works advance domain-specific evaluation and structured reasoning, but do not by themselves establish how general-purpose VLMs can reliably recover and use drawing structure. Treating symbolic drawings simply as another category of natural images or documents risks overlooking their specific demands on entity identification, annotation binding, and geometric relations. We investigate these structural demands by providing question-independent structure alongside the same image and question, and study whether its benefits can be transferred to a model that requires no structural input at deployment.

2.2 On-Policy Distillation with Privileged Information

On-policy distillation supervises a student on prefixes sampled from its own policy, rather than only fixed teacher trajectories Agarwal et al. (2024); MiniLLM is a representative reverse-KL language-model distillation method, illustrating that the distributional objective is separate from how prefixes are obtained Gu et al. (2024). Context distillation transfers behavior enabled by additional context to a model without it Snell et al. (2022), and OPCD combines this idea with on-policy learning through a context-conditioned teacher and a context-free student Ye et al. (2026). For VLMs, Vision-OPD similarly distills along student trajectories from a teacher conditioned on local crops to a student with the full image Yuan et al. (2026). SG-OPD follows these training principles but uses question-independent visual structure from the same drawing as privileged teacher context: it organizes objects, annotation bindings, cross-view correspondences, and geometric constraints, then transfers this benefit through forward-KL supervision on student-generated prefixes to a student requiring only the image and question at inference time.

3 Problem Formulation

3.1 Symbolic Drawing Question Answering

Let I be a symbolic drawing image, q its question, and y the target answer. For a dataset $\mathcal{D}$ of drawing–question–answer triples, an image-only VLM represents the conditional answer distribution as

$$p_\theta(y \mid I, q) = \prod_{t=1}^{|y|} p_\theta(y_t \mid I, q, y_{<t}),$$

where y_t is the t-th answer token and $y_{<t}$ is its preceding prefix. Solving the task requires the model to recover entities, attributes, annotations, and relations from I, select the evidence relevant to q, and compose an answer. These operations are coupled in the image-only model: an incorrect object choice, dimension binding, or cross-view correspondence can therefore propagate to later reasoning and alter y.

3.2 Question-Independent Structural Representation

For a drawing domain d, a domain-specific generator maps the complete image to a structured representation z:

$$z = G_d(I), \qquad G_d : \mathcal{I} \to \mathcal{Z}_d, \qquad G_d(I; q, y, \rho) = G_d(I) \text{ for all } q, y, \rho,$$

where ρ denotes a solution rationale. The image includes graphical elements and text rendered inside the drawing. Thus, z may contain CAD dimensions, callouts, view labels, and title-block contents, or diagram labels, measurements, and explicitly marked geometric relations. The external problem statement, information absent from the drawing, question q, answer y, and rationale ρ are unavailable to G_d. The representation is consequently question-independent evidence rather than an answer, a solution trace, or a hint derived from a known answer. It can be generated once and reused for all questions associated with the same drawing.

3.3 CAD and Math Structural Representations

We consider two representative symbolic-drawing domains: mechanical CAD drawings and mathematical geometry problems. As illustrated in Figure 2, we design a compact, domain-specific structural representation for each domain.

For a mechanical CAD drawing, the representation uses a multi-view feature schema that organizes title-block metadata, drawing views, salient geometric features, dimensions, engineering constraints, and high-level part semantics. Shared geometry IDs link instances of the same feature across different views, making cross-view correspondence and dimension-to-feature binding explicit.

For a mathematical geometry problem, the representation uses a typed geometric graph containing labeled points, necessary synthetic points, geometric objects, typed relations, and measurements bound to their corresponding objects.

These representations expose the identities and bindings that are often difficult to recover reliably from pixels while retaining only evidence visible or explicitly annotated in I. In particular, persistent feature identifiers make repeated CAD features correspond across views, and typed graph relations bind geometric measurements to their intended objects. This shared formulation lets us compare direct structure-assisted inference with an image-only model under the same image, question, decoding procedure, and answer format.

3.4 Structure-Assisted Inference and Privileged Training Information

Given $z = G_d(I)$, the structure-assisted model augments the image and question with z and predicts

$$p_{\mathrm{str}}(y \mid I, q, z) = \prod_{t=1}^{|y|} p_{\mathrm{str}}(y_t \mid I, q, z, y_{<t}),$$

whereas the image-only model uses $p_{\mathrm{img}}(y_t \mid I, q, y_{<t})$. The two distributions differ only in access to z; both retain I so that fine visual evidence and typography omitted by the compact representation remain available. Direct comparison measures the benefit of recovering explicit structure and defines the performance available when the drawing can be parsed reliably.

At deployment, however, generating and storing z may be undesirable. We therefore treat z as privileged information available to a teacher only during training. For a student-generated prefix $\hat{y}_{<t}$, the teacher provides $p_T(\cdot \mid I, q, z, \hat{y}_{<t})$, while the student provides $p_\theta(\cdot \mid I, q, \hat{y}_{<t})$. The training objective transfers the teacher's structural advantage at the visited prefixes, and the resulting student requires only (I, q) at inference. This formulation separates the diagnostic value of explicit structure from the deployable goal of internalizing that value in an image-only model.

4 Method

We design the method around a simple separation of concerns. A drawing first contains a latent set of entities and relations, while the question selects which part of that set must be used to produce an answer. Current VLMs often attempt both operations in one pass, so an early mistake in identifying a

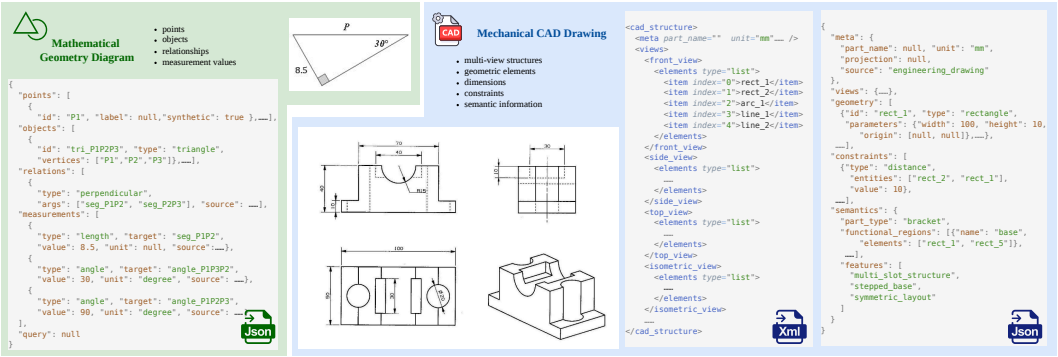

Figure 2: Examples of JSON/XML-style structural representations for two representative symbolic-drawing domains: a mechanical CAD drawing and a mathematical geometry problem.

feature or binding a dimension can contaminate all subsequent reasoning. Our framework therefore has two stages: (i) expose a question-independent representation of the drawing to measure the value of recovering this latent structure, and (ii) use that representation as privileged information during training so that an image-only model can learn the corresponding behavior. The first stage is deliberately training-free and serves as a controlled diagnostic; the second stage is the deployable model.

4.1 Framework Overview and Design Motivation

For each drawing-question pair (I, q), a domain-specific generator produces $z = G_d(I)$ before the question is considered. Keeping z independent of q is important for two reasons. It prevents the representation from encoding the answer or a question-specific hint, and it allows the same parsed drawing to be reused for multiple questions. We include the image together with z rather than replacing the image: the image preserves fine visual evidence and typography that a compact parser may omit, whereas z provides stable identities and relations that are difficult to recover from pixels. This combination directly targets the observed failure mode of entity–attribute and cross-view binding.

The resulting structure-assisted model is used as a teacher, and an image-only model with the same answer interface is used as a student. The teacher has access to (I, q, z) only during training of the student; at deployment the student receives (I, q) and no generated file or parser output. This privileged-information setup isolates the scientific question of whether explicit structure can be internalized, while preserving the standard VLM inference contract.

4.2 Question-Independent Structural Representation

Let $z = G_d(I)$ denote the representation of drawing I in domain d. We use a compact schema whose fields correspond to the relations needed for downstream answering. In CAD, z records metadata, view decomposition, salient geometric features, dimensions, engineering constraints, and part-level semantics; shared feature identifiers link the same hole, slot, or profile across views. In geometry diagrams, z is a typed graph of points, lines, circles, polygons, measurements, and relations such as incidence, parallelism, perpendicularity, and equal lengths. We record a relation only when it is visible or explicitly annotated, and use null or omission for uncertain values. This conservative policy is motivated by error propagation: a fabricated relation can make a confident but invalid proof, whereas an omitted relation can still be recovered from the image.

Formally, the generator maps directly from the complete drawing image to a domain-specific space, $G_d : \mathcal{I} \to \mathcal{Z}_d$, and satisfies

$$G_d(I; q, y, \rho) = G_d(I) \qquad \forall q, y, \rho,$$

where y and ρ are the answer and reasoning. Thus z contains no target-dependent supervision. We bind each annotation to its most plausible geometric target and assign persistent identifiers to

repeated features; these choices are motivated by the fact that most errors in our setting concern which object a number belongs to, rather than arithmetic after the correct object has been selected. The representation is serialized as text and supplied as an auxiliary context, so it can be used with existing VLMs without changing their visual encoder.

4.3 Structure-Assisted Reasoning

Let y_t be the t-th answer token. An image-only model predicts

$$p_{\text{img}}(y_t \mid I, q, y_{<t}),$$

whereas the structure-assisted model predicts

$$p_{\text{str}}(y_t \mid I, q, z, y_{<t}).$$

Both models receive exactly the same image, question, and answer format. The only additional input is z. This controlled comparison is motivated by the need to distinguish gains from better structural grounding from gains caused by extra textual context or a different prompt. By externalizing object identities, annotation bindings, and cross-view relations, z lets the model allocate its capacity to selecting relevant evidence and carrying out the required geometric or mathematical operation. We therefore treat the structure-assisted result as an upper bound on what can be achieved when the drawing has been parsed reliably, while retaining the image to avoid testing a parser-only system.

4.4 Structure-Guided On-Policy Distillation

Direct structure access is useful but requires running and storing a domain-specific generator at inference. SG-OPD treats the clean representation as privileged information available to a teacher only during training. Given (I, q, z), the teacher predicts

$$p_T(v \mid I, q, z, \hat{y}_{<t}),$$

while the student sees only (I, q) and predicts

$$p_\theta(v \mid I, q, \hat{y}_{<t}),$$

for vocabulary token v. Evaluating both distributions on the same prefix ensures that the teacher's structural advantage is expressed as local next-token guidance rather than as a separate answer or rationale target.

On-policy prefixes and token-level objective. We sample prefixes from the current student policy. For each $(I, q, y) \sim \mathcal{D}$, the student rolls out

$$\hat{\tau} = (\hat{y}_1, \ldots, \hat{y}_T) \sim p_\theta(\cdot \mid I, q).$$

At every visited prefix $\hat{y}_{<t}$, the teacher supplies a distribution over the next token. On-policy sampling is motivated by exposure bias: fixed reference prefixes rarely contain the wrong object choices and malformed intermediate answers that the deployed student will actually encounter. Asking the teacher to correct these states trains the student to recover after early binding errors and reduces the train–test mismatch.

We optimize the forward-KL objective

$$\mathcal{L}_{\text{SG-OPD}} = \mathbb{E}_{(I,q)\sim\mathcal{D},\, \hat{\tau}\sim p_\theta} \left[\frac{1}{|\hat{\tau}|} \sum_{t=1}^{|\hat{\tau}|} D_{\text{KL}}(p_T(\cdot \mid I, q, z, \hat{y}_{<t}) \,\|\, p_\theta(\cdot \mid I, q, \hat{y}_{<t})) \right].$$

Forward KL is chosen because it makes the student cover the teacher's structurally informed support, including alternative tokens that remain plausible under ambiguous visual evidence, instead of collapsing only to a single sampled answer. We stop gradients through the teacher and update only θ. After training, both teacher and structure input are discarded and deployment uses $\hat{y} \sim p_\theta(\cdot \mid I, q)$, so the computational cost and interface match the original image-only baseline.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We evaluate symbolic-drawing understanding in two domains: mechanical CAD drawings and mathematical geometry problems. The CAD setting tests whether a model can ground numerical annotations and engineering symbols in the correct geometric features, often across multiple orthographic or sectional views. The mathematical setting tests whether it can extract and use diagrammatic evidence while performing geometric or algebraic reasoning.

Evaluation benchmarks. For CAD, we evaluate on MechVQA Kou et al. (2026). For mathematics, all evaluations are conducted on the test-mini splits of MathVista Lu et al. (2024) and MathVerse Zhang et al. (2024). We use the *geometry reasoning*, *algebraic reasoning*, and *geometry problem solving* subsets of MathVista test-mini, together with all subsets of MathVerse test-mini.

Training data. For the CAD domain, we construct a training set of 6,097 question–answer pairs from mechanical CAD drawings. This set is strictly disjoint from MechVQA. It contains two types of questions: basic drawing-reading questions, which require directly identifying dimensions, hole diameters, hole spacing, sectional views, and symmetry; and advanced engineering-reasoning questions, which cover process planning, tolerance and assembly analysis, error calculation, and descriptions of operating principles. The two categories account for approximately 80% and 20% of the CAD training data, respectively.

For mathematics, we use the training sets of UniGeo Chen et al. (2022) and PGPS9K Zhang et al. (2023), totaling 18,195 examples. The set includes 11,520 multiple-choice questions: 8,021 from PGPS9K and 3,499 UniGeo calculation problems. The remaining 6,675 examples are non-multiple-choice or proof problems, all drawn from UniGeo proving. This mixture supplies both answer-selection supervision and open-ended geometric reasoning trajectories.

Training configuration. All trained models use Qwen3-VL-8B-Instruct Team (2025). Unless specified otherwise, SG-OPD runs for one epoch with batch size 32. For each question, the student generates $N = 8$ on-policy rollouts, whose prefixes receive token-level supervision from the structure-assisted teacher (Section 4.4). We use this configuration in both domains to isolate the effects of domain and available structure.

5.2 STRUCTURE-ASSISTED REASONING RESULTS

We first quantify the benefit of providing the question-independent representation at inference time, before studying SG-OPD. We compare six settings:

- **Image only** uses the drawing and question without an external structural representation.
- **Image+JSON** adds the Gemini-3-Flash-generated structure serialized as JSON.
- **Image+XML** replaces the JSON serialization with an XML representation obtained by rule-based transformation of the same JSON structure.
- **JSON only** removes the image and answers from the generated JSON alone.
- **JSON native** has the evaluated model generate its own JSON from the image before answering.
- **JSON perturbation** supplies the image and generated JSON after geometric-object IDs are replaced with consistent meaningless random names across ID fields, values, references, lists, and matching keys; dimensions, geometry, and semantic labels are unchanged.

All scores below are exact-answer accuracy; the paired differences are computed against the image-only condition for the same model and split.

Mechanical CAD. Table 1 shows that augmenting the image with a structured representation generally improves MechVQA performance, demonstrating that the benefit is not restricted to a single serialization. Clean JSON improves every evaluated model over its image-only baseline. XML also

Model	Image only	Image+JSON	Image+XML	JSON only	JSON native	JSON pert.
Qwen3-VL-235B-A22B	82.30	**85.00**	78.98	37.38	79.49	79.75
Claude Sonnet-4-thinking	70.50	76.80	**77.20**	47.08	69.87	75.11
Doubao-Seed-2.0-Pro	89.70	**91.00**	86.30	50.54	90.80	89.70
Claude Sonnet-4.6	60.17	83.40	83.80	55.02	82.78	**83.88**
GPT-4o Hurst et al. (2024)	70.00	73.24	73.50	43.29	67.17	**75.11**
Doubao-Seed-1.8	88.10	88.35	85.23	47.51	87.93	**89.45**
Qwen3.6-Plus	89.20	89.45	89.11	48.86	**91.81**	89.36
GPT-5.4	83.70	86.91	**86.92**	54.00	84.47	85.99

Table 1: MechVQA accuracy (%). Clean structure is supplied in the Image+JSON/XML conditions; the remaining columns are diagnostic ablations.

produces substantial gains for several models, but is less consistent and can slightly reduce accuracy for others. One plausible explanation is that XML is more verbose and relies on nested tag boundaries, making long-range references and object–attribute associations harder to recover; differences in models' pretraining exposure to and ability to parse XML may further amplify this serialization sensitivity. Thus, the results support the utility of structure itself while showing that the way the structure is encoded still affects how reliably a model can use it.

The JSON-only condition remains capable of answering a non-trivial fraction of the questions, confirming that the representation itself contains task-relevant geometry, dimensions, and relations. Nevertheless, it is consistently much worse than Image+JSON, showing that the structured text is not sufficient and that the gains cannot be explained by simply converting all visual evidence into text. Instead, the model benefits from jointly grounding the explicit relations in the JSON against the original drawing. Native JSON generation improves some models but not others, which is consistent with differences in their intrinsic ability to extract a faithful structure before reasoning: self-generated structure helps only when the model can construct it reliably.

Finally, consistently renaming geometric-object IDs with meaningless identifiers leaves performance strong for most models and often preserves much of the clean-JSON gain. Because this perturbation changes the object names while preserving reference consistency, geometry, and relational connectivity, the result indicates that models do not primarily align JSON entities with visual objects through the literal identifier strings. They instead appear to rely mainly on the pattern of relations among objects—together with the image and retained geometric attributes—to establish the correspondence needed for reasoning.

Mathematical geometry. The mathematics benchmarks show larger and more uniform gains. As shown in Table 2, Image+JSON improves all MathVista skills by roughly 16–17 points; for example, geometry problem solving increases from 69.71% to 87.02%. The consistency across skills indicates that the representation supports both visual extraction and subsequent symbolic computation, rather than benefiting only one question format.

MathVerse test-mini provides a controlled test of visual dependence. Direct access to the structure (Image+JSON) improves all five splits by 4.82–6.85 points, whereas the image-only SG-OPD student shows smaller and uneven gains of 0.12–2.79 points. The direct-structure improvements are especially strong in spatial and symbolic tasks such as solid or analytic geometry, volume, and expression manipulation. Overall, these results support the CAD diagnosis: making points, objects, and measurements explicit lets the model focus more reliably on the required proof or calculation.

Taken together, the two domains exhibit the same qualitative ordering—image plus clean structure is strongest and image-only is the appropriate deployment baseline—but differ in where the structure helps most. CAD benefits are concentrated in grounding dimensions and view correspondences, while mathematics benefits are more pronounced for multi-step geometric operations. These complementary gains motivate SG-OPD: the next experiments test whether the structure-assisted behavior can be internalized during training and retained when the structure is removed at inference.

5.3 Structure-Guided On-Policy Distillation Results

We evaluate SG-OPD as an image-only student: it is trained with supervision from the structure-assisted teacher but receives no JSON at test time. Gains in this subsection are measured against

MathVista skill	Image only	Image+JSON	SG-OPD	Δ Image+JSON	Δ SG-OPD
Geometry reasoning	69.46	85.77	78.24	+16.31	+8.78
Algebraic reasoning	68.33	84.70	77.94	+16.37	+9.61
Geometry problem solving	69.71	87.02	81.25	+17.31	+11.54

Table 2: MathVista test-mini accuracy (%) for image-only, structure-assisted, and SG-OPD inference. Both gains are measured against the image-only baseline.

MathVerse split	Image only	Image+JSON	SG-OPD	Δ Image+JSON	Δ SG-OPD
Text Lite	45.43	51.27	45.56	+5.84	+0.13
Vision Only	37.82	42.64	37.94	+4.82	+0.12
Vision Dominant	44.42	51.27	44.62	+6.85	+0.20
Vision Intensive	43.15	48.98	43.40	+5.83	+0.25
Text Dominant	49.49	56.22	52.28	+6.73	+2.79

Table 3: MathVerse test-mini Overall accuracy (%) across visual-dependence splits. Only the Overall score is shown; both gains are measured against the Image-only baseline.

the untrained Qwen3-VL-8B image-only baseline; the Image+JSON result serves as the teacher reference.

Mechanical CAD. SG-OPD improves MechVQA accuracy from 72.80% to 73.40%, recovering part of the teacher's 2.80-point advantage (Table 4). The 0.60-point gain provides initial evidence that drawing structure can be partially internalized without an explicit representation at inference.

Benchmark	Image only	Image+JSON	SG-OPD
MechVQA	72.80	75.60	73.40

Table 4: MechVQA accuracy (%) for the image-only baseline, structure-assisted teacher, and image-only SG-OPD student.

MathVista. Transfer is substantially stronger on MathVista: SG-OPD gains 8.78–11.54 points over the image-only baseline and recovers approximately 54–67% of the improvement provided by direct JSON access (Table 2). The largest transfer occurs on geometry problem solving, where accuracy rises from 69.71% to 81.25%.

MathVerse. In contrast, transfer on MathVerse is limited and uneven: SG-OPD gains 0.12–2.79 points across the five splits, despite the larger gains obtained with direct JSON access (Table 3). The largest improvement occurs on Text Dominant, while the other four splits remain close to the image-only baseline (within 0.25 points). This gap suggests that fine-grained diagram structure is not yet transferred effectively on most MathVerse splits.

Overall, SG-OPD transfers a substantial fraction of the structural advantage on MathVista, a smaller fraction on MechVQA, and little on MathVerse. Direct access to the structured representation therefore remains a useful upper bound and highlights where more effective structural transfer is needed.

6 Conclusion

We studied how explicit structure can improve VLM understanding of symbolic drawings. Across mechanical CAD and mathematical geometry tasks, structure-assisted reasoning consistently improved performance by making objects, annotations, and relations more explicit. Our proposed Structure-Guided On-Policy Distillation method further transferred part of this benefit to an image-only student, yielding gains without requiring structured input at inference time. These results highlight structured representations as a useful tool for symbolic visual reasoning and motivate future work on more accurate structure extraction and more effective distillation.

AI USE STATEMENT

We used large language models (LLMs) solely to polish and improve the clarity, grammar, and readability of the manuscript. The LLMs were not used to generate the research ideas, design the methodology, conduct the experiments, analyze the results, or draw the scientific conclusions. All AI-assisted text was reviewed and verified by the authors, who take full responsibility for the final content of this work.

ETHICS STATEMENT

This work does not involve human participants, personal data, sensitive information, or other ethically problematic activities. We are not aware of any ethical issues associated with this research.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide a structured description of the generation prompts in the appendix, together with the OPD training settings and the composition of the mathematics training dataset. The CAD-domain training dataset was constructed by the authors; due to copyright restrictions, it cannot currently be released publicly. Nevertheless, the methodology, training procedures, prompt structures, and dataset composition are described in sufficient detail to enable reproduction of our approach.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

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.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Qian Kou, Xiaofeng Shi, Yulin Li, Xiaosong Qiu, Xinyang Wang, Hua Zhou, and Cao Dongxing. Mechvqa: Benchmarking and enhancing multimodal llms on comprehensive mechanical drawing understanding. *arXiv preprint arXiv:2605.30794*, 2026.

Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023.

Siqi Li, Yufan Shen, Xiangnan Chen, Jiayi Chen, Hengwei Ju, Haodong Duan, Song Mao, Hongbin Zhou, Bo Zhang, Bin Fu, et al. Gdi-bench: a benchmark for general document intelligence with vision and reasoning decoupling. *arXiv preprint arXiv:2505.00063*, 2025.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pp. 216–233. Springer, 2024.

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.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, volume 2024, pp. 23439–23554, 2024.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 2199–2208. IEEE, 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Fedor Rodionov, Abdelrahman Eldesokey, Michael Birsak, John Femiani, Bernard Ghanem, and Peter Wonka. Floorplanqa: A benchmark for spatial reasoning in llms using structured representations. *arXiv preprint arXiv:2507.07644*, 2025.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

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

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026.

Qianhao Yuan, Jie Lou, Xing Yu, Hongyu Lin, Le Sun, Xianpei Han, and Yaojie Lu. Vision-opd: Learning to see fine details for multimodal llms via on-policy self-distillation. *arXiv preprint arXiv:2605.18740*, 2026.

Ming-Liang Zhang, Fei Yin, and Cheng-Lin Liu. A multi-modal neural geometric solver with textual clauses parsed from diagram. In *IJCAI*, 2023.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pp. 169–186. Springer, 2024.

A Appendix

A.1 Structured Representation Generation Prompts

The following prompts generate the question-independent structured representations described in Section 3.

Math structured representation generation prompt

You are an expert in interpreting mathematical geometry diagrams. Convert the provided geometry image and visible problem text into a typed geometric graph.
Required output schema. Output exactly one valid JSON object with the following top-level structure:

```
{
  "points": [],
  "objects": [],
  "relations": [],
  "measurements": [],
  "query": null
}
```

- Output JSON only; do not output explanations or reasoning.
- Do not solve the problem.
- Only record information visible in the diagram or explicitly stated in the text.

1. Points

- Represent a visible labeled point as follows:

```
{
  "id": "A",
  "label": "A",
  "synthetic": false
}
```

- Important unlabeled points may use synthetic IDs: `I1`, `I2`, ... for intersections; `T1`, `T2`, ... for tangent points; and `P1`, `P2`, ... for other points. For example:

```
{
  "id": "I1",
  "label": null,
  "synthetic": true
}
```

- Do not treat every letter as a point. A letter beside an entire line may be a line label.

2. Objects

- Allowed object types are `segment`, `line`, `ray`, `circle`, `arc`, `angle`, `triangle`, `quadrilateral`, and `polygon`.
- Use the following formats for the basic object types:

```
{
  "id": "seg_AB",
  "type": "segment",
  "points": ["A", "B"]
}
{
  "id": "line_a",
  "type": "line",
  "label": "a",
  "points": []
}
{
  "id": "ray_AB",
  "type": "ray",
  "start": "A",
  "through": "B"
}
{
  "id": "circle_O",
  "type": "circle",
  "center": "O"
}
{
  "id": "arc_AB",
  "type": "arc",
  "circle": "circle_O",
  "endpoints": ["A", "B"],
  "kind": "unknown"
}
{
  "id": "angle_BAC",
  "type": "angle",
  "vertex": "A",
  "points": ["B", "A", "C"],
  "label": null
}
```

- Represent a numbered angle as follows:

```
{
  "id": "angle_1",
  "type": "angle",
  "vertex": "I1",
  "between": ["line_a", "line_c"],
  "region": "upper_right",
  "label": "1"
}
```

  Allowed angle regions are `upper_left`, `upper_right`, `lower_left`, `lower_right`, `left`, `right`, `above`, `below`, and `unknown`.

- Use visible cyclic order for polygon vertices. For example:

```
{
  "id": "tri_ABC",
  "type": "triangle",
  "vertices": ["A", "B", "C"]
}
```

3. Relations

- Each relation has the following form, where the allowed sources are `diagram` and `text`:

```
{
  "type": "relation_name",
  "args": [],
  "source": "diagram"
}
```

- Common relations include the following:

```
{
  "type": "on",
  "args": ["A", "circle_O"],
  "source": "diagram"
}
{
  "type": "intersect",
  "args": ["seg_AD", "seg_BC", "I1"],
  "source": "diagram"
}
{
  "type": "collinear",
  "args": ["A", "B", "C"],
  "source": "diagram"
}
{
  "type": "parallel",
  "args": ["line_a", "line_b"],
  "source": "diagram"
}
{
  "type": "perpendicular",
  "args": ["seg_AB", "seg_CD"],
  "source": "diagram"
}
{
  "type": "equal_length",
  "args": ["seg_AB", "seg_CD"],
  "source": "diagram"
}
{
  "type": "equal_angle",
  "args": ["angle_1", "angle_2"],
```

```
    "source": "diagram"
  }
  {
    "type": "midpoint",
    "args": ["M", "seg_AB"],
    "source": "diagram"
  }
```

- Circle relations include `center_of`, `radius`, `diameter`, `chord`, `tangent`, `secant`, and `concyclic`. For example:

```
{
  "type": "radius",
  "args": ["seg_OB", "circle_O"],
  "source": "diagram"
}
{
  "type": "chord",
  "args": ["seg_AB", "circle_O"],
  "source": "diagram"
}
{
  "type": "tangent",
  "args": ["line_a", "circle_O", "T1"],
  "source": "diagram"
}
{
  "type": "concyclic",
  "args": ["A", "B", "C", "D"],
  "source": "diagram"
}
```

- Do not infer parallel, perpendicular, equal, midpoint, or tangent relations only from approximate appearance.

4. Measurements

- Bind every visible numerical or symbolic annotation to an object whenever possible. Examples include:

```
{
  "type": "length",
  "target": "seg_AB",
  "value": 5,
  "unit": null,
  "source": "diagram"
}
{
  "type": "angle",
  "target": "angle_BAC",
  "value": 50,
  "unit": "degree",
  "source": "diagram"
}
{
  "type": "angle",
  "target": "angle_1",
  "value": 55,
  "unit": "degree",
  "source": "diagram"
}
```

- Other measurement types are `radius`, `diameter`, `arc_measure`, `area`, and `perimeter`.
- Use strings for symbolic values, such as `"x"`, `"2x+3"`, `"sqrt(3)"`, and `"60-x"`.

- Use this binding priority:
  1. annotation arrow;
  2. angle arc or dimension mark;
  3. nearest compatible object;
  4. spatial position.
- A bare number inside an angle is usually an angle label. A value containing a degree symbol is an angle or arc measurement.

5. Query

- Use the applicable query format:

```
{
  "type": "find_length",
  "target": "seg_AB"
}
{
  "type": "find_angle",
  "target": "angle_1"
}
{
  "type": "find_radius",
  "target": "circle_O"
}
{
  "type": "find_area",
  "target": "tri_ABC"
}
```

- Represent a proof query as follows:

```
{
  "type": "prove",
  "target": {
    "type": "parallel",
    "args": ["seg_AB", "seg_CD"]
  }
}
```

- If no question is visible, use `"query": null`.

Critical rules

1. Inspect the entire image before producing JSON.
2. Extract visible points, lines, segments, rays, circles, arcs, angles, and polygons.
3. Distinguish point labels, line labels, angle labels, and measurement values.
4. Create synthetic IDs for important unlabeled intersections or tangent points.
5. Bind all visible numerical annotations whenever possible.
6. Do not assume the diagram is drawn to scale.
7. Do not invent invisible segments, arcs, or mathematical relations.
8. Only create arcs that are marked, measured, highlighted, or referenced.
9. Every ID used in a relation or measurement must exist in `"points"` or `"objects"`.
10. Avoid duplicate objects, relations, and measurements.
11. Omit genuinely uncertain information rather than guessing.
12. Return valid JSON even when some arrays are empty.

Now analyze the provided geometry image and output only the JSON object.

CAD structured representation generation prompt

You are an expert in interpreting mechanical engineering drawings. Convert the given engineering drawing image into a structured JSON representation. Follow the rules strictly: do not omit fields, do not invent values, and use `null` whenever a value is unclear.

Required output schema. Output a valid JSON object with exactly the following top-level structure:

```
{
  "meta": {},
  "views": {},
  "geometry": [],
  "constraints": [],
  "semantics": {}
}
```

1. Meta (basic information)

- `"part_name"`: a string, for example from the title text in the drawing;
- `"unit"`: use `"mm"` by default unless another unit is specified;
- `"projection"`: use `"first_angle"` or `"third_angle"` if identifiable; otherwise use `null`;
- `"source"`: use `"engineering_drawing"`.

2. Views (view decomposition)

- Identify the different views, such as `top_view` and `front_view`.
- Each view must contain:

```
{
  "elements": [list of geometry IDs visible in this view]
}
```

- Do not redefine geometry in the view descriptions.

3. Geometry (core geometric elements)

- Extract only geometry that is dimensioned, semantically salient, or needed for downstream reasoning.
- Do not enumerate every visible construction line, border line, hatch edge, or repeated tiny segment. If the drawing contains many unnamed low-level lines, summarize them at the feature level instead.
- Each geometry item must include:

```
{
  "id": unique string,
  "type": one of [
    "circle", "hole", "slot", "rectangle",
    "arc", "line", "fillet"
  ],
  "parameters": { ... },
  "transform": {
    "translation": [x, y, z],
    "rotation": [rx, ry, rz]
  }
}
```

- Follow these geometry rules:
  - All elements must have unique IDs.
  - Use explicit numeric values rather than symbolic values Φ.
  - Keep units consistent in millimeters.
  - If a center is unknown, use `null`.
  - Prefer high-value elements: holes, slots, circles, main outlines, key rectangles, arcs, and fillets.

  - Avoid outputs such as `line_1`, ..., `line_500` unless a particular line is explicitly dimensioned and necessary.
  - Keep the geometry list compact enough to remain valid JSON within the model output limit.
- Use the following type-specific parameters:

```
Circle:
{ "center": [x, y], "diameter": number }

Hole:
{ "center": [x, y], "diameter": number }

Slot:
{ "center": [x, y] or null, "angle": degree,
  "width": number, "depth": number or null }

Rectangle:
{ "width": number, "height": number, "origin": [x, y] }

Fillet:
{ "radius": number, "position": string or null }
```

4. Constraints (geometric relationships)

- Explicitly extract relationships. Each constraint has the following form:

```
{
  "type": one of [
    "concentric", "parallel", "perpendicular",
    "symmetric", "distance", "angle"
  ],
  "entities": [list of geometry IDs],
  "value": number or null
}
```

- Do not assume implicit relationships; include a relationship only if it is visually indicated.
- Express angles in degrees and distances as numeric values.
- Only include constraints tied to explicit dimensions, callouts, centerlines, or clear geometric annotations.

5. Semantics (high-level understanding)

- Use the following structure:

```
{
  "part_type": string (e.g., "connector", "bracket") or null,
  "functional_regions": [
    {
      "name": string,
      "elements": [geometry IDs]
    }
  ],
  "features": [list of high-level features]
}
```

- Examples of features are `"rotational_symmetry"`, `"multi_slot_structure"`, and `"stepped_base"`.

Strict constraints

- Output must be valid JSON, with no comments or trailing commas.
- Do not include explanations or hallucinate missing dimensions.
- Use `null` if uncertain.
- All numeric values must be numbers, not strings.

- Prioritize completeness of important dimensions and features over exhaustive enumeration of minor line segments.

Goal. The output should support reconstruction of geometry, multi-hop reasoning, and visual question answering.
Now analyze the image and produce the JSON.