Informatics Foundry

Engineering Informatics and Intelligent Systems 1 (2026)
*journal homepage: http://eiis.informaticsfoundry.org/*

**Research Article**

# Graph Retrieval-Augmented Visual Question Answering for Complex Engineering Drawings

Weiguo Shi[1], Peiling Zhuang[2,*], Aili Pang[2], Zilei Wang[2], Chanjuan Tang[2]

[1]*State Grid Shanghai Economic Research Institute, State Grid Corporation of China Co., Ltd., Shanghai, China*
[2]*State Grid Shanghai Electric Power Design Co. Ltd, Shanghai, China*

* Correspondence: zhuangpl@sh.sgcc.com.cn

**ABSTRACT** Complex engineering drawings, such as piping and instrumentation diagrams (P&ID), substation wiring diagrams, and industrial system topology diagrams, contain dense symbols, text annotations, and multi-hop connectivity relations. Existing MLLMs can handle general visual question answering, but they still face two key limitations in engineering drawings: the high context burden of complete drawings or complete structural files, and the lack of verifiable topological evidence for final answers. This paper proposes Neuro-Symbolic Graph Retrieval-Augmented Generation (NSG-RAG), which reformulates engineering drawing question answering as topology-grounded visual question answering under the setting where PID2Graph provides GraphML topology annotations. NSG-RAG does not retrain an end-to-end drawing parser. Instead, it constructs a retrievable evidence space around GraphML. Given a question, the system performs descriptive node grounding, retrieves relevant $k$-hop subgraphs and multi-hop paths, and verifies whether candidate paths are supported by the underlying graph structure through NetworkX rules. A controlled PID-GraphQA evaluation built from 500 PID2Graph Complete drawings covers symbol recognition, direct connection, negative connection, multi-hop path, and unsupported path claim tasks. Experiments show that, when GraphML is available, graph-structured retrieval and rule verification improve answer accuracy, evidence traceability, and multi-hop topological reasoning. This study is positioned as a proof-of-concept investigation of GraphRAG for complex engineering drawings and provides a reproducible basis for future evaluation with real MLLM backbones and analysis of end-to-end parsing error propagation.

*Keywords:* engineering drawings analysis, blueprint interpretation, retrieval-augmented generation, multi-hop reasoning, traceable question answering

## 1. Introduction

Complex engineering drawings are core information carriers for industrial system design, operation, maintenance, and safety review. Taking piping and instrumentation diagrams (Piping and Instrumentation Diagram, P&ID) as an example, a drawing contains equipment symbols, valves, instruments, connection points, pipelines, and cross-region connectivity relations. Unlike natural images, the semantics of engineering drawings come not only from local visual appearance but also from the topology among symbols. Therefore, question answering for complex engineering drawings must answer not only what appears in the drawing, but also whether two objects are connected, which nodes a connection path passes through, and whether a connection claim is supported by the drawing structure.

MLLMs have shown strong performance in general visual question answering and document understanding. From image-text alignment pretraining in CLIP and frozen visual encoders in BLIP-2 to visual instruction tuning in LLaVA, the MLLM paradigm has made it common to feed general image semantics into LLM reasoning pipelines [1–3]. Models such as Qwen2.5-VL, InternVL3, and GPT-4o also provide strong image-language alignment capabilities [4–6]. However, engineering drawing question answering differs structurally from ordinary VQA. First, symbols are small, lines are dense, and local appearances are often similar, which makes purely visual recognition sensitive to resolution and line width. Second, many questions depend on multi-hop topological relations rather than a single visual target. Third, engineering review requires answers to be traceable and verifiable through evidence chains. Structured VQA benchmarks such as GQA show that final answer accuracy alone is insufficient for evaluating compositional reasoning and grounding [7]. ReasonVQA further shows that multi-hop structural knowledge substantially increases the difficulty of visual question answering [8].

A direct approach is to convert the complete GraphML or JSON graph structure into text and feed it to a large model in one pass. This Static Full-Graph Prompt provides more structural information than purely visual input, but a complete graph contains many nodes, edges, coordinates, and attributes that are irrelevant to the current question, which easily causes context redundancy. Critical evidence can also be diluted in long contexts [9]. Work such as ChatP&ID shows that converting engineering drawings into knowledge graphs and performing GraphRAG queries can reduce the cost of directly processing raw images or smart files and improve answer grounding [10]. PIDQA further shows that P&ID question answering requires

explicit question types, structured queries, and executable answer verification [11]. These studies jointly indicate that the core of engineering drawing question answering is not simply answering from images, but answering with graph structure as evidence.

This paper proposes NSG-RAG, a neuro-symbolic graph retrieval-augmented generation framework for visual question answering over complex engineering drawings. Unlike ChatP&ID, which uses an interactive DEXPI / Neo4j knowledge graph, this paper uses the images, patches, and GraphML annotations provided by PID2Graph [12, 13] and studies topology-grounded QA under the setting where GraphML is available. NSG-RAG treats GraphML as an extracted symbolic and topological evidence space. Given a question, the system performs descriptive node grounding, retrieves relevant $k$-hop subgraphs and multi-hop paths, verifies candidate evidence through NetworkX rules, and finally feeds compact structural evidence into an LLM/MLLM. This paper does not claim to solve end-to-end engineering drawing parsing. Instead, it verifies whether dynamic graph-structured retrieval and rule checking can improve answer accuracy, evidence traceability, and control of unsupported claims in complex drawing question answering. Figure 1 summarizes the overall NSG-RAG workflow.

The contributions of this paper are as follows:

- We propose NSG-RAG, a framework that treats the GraphML topology of complex engineering drawings as a retrievable evidence space and combines $k$-hop subgraphs, multi-hop paths, and neuro-symbolic rule verification to generate traceable answers.
- We construct PID-GraphQA, a controlled evaluation task based on PID2Graph, covering symbol recognition, direct connection, negative connection, multi-hop path, and unsupported path claim. We report stratified results by task type, difficulty, and hop length.
- We establish an evaluation protocol for engineering drawing QA that reports ACC, Evi-F1, Evidence Coverage (Evi-Cov), and MH-Acc in the main results, and separately reports unsupported claims in the error analysis to distinguish answer correctness, verifiable evidence, and invocation risk.

## 2. Related Work

### 2.1. Engineering diagram understanding and graph digitization

Engineering drawing understanding has long studied symbol detection, text recognition, line tracing, and connection relation extraction. Earlier methods usually decomposed the task into symbol detection, line detection, and relation recovery modules, while recent methods increasingly use deep learning models to jointly identify equipment, instruments, valves, and connectivity relations in P&ID drawings [14–17]. PID2Graph further defines P&ID digitization as a graph reconstruction problem and provides image-level and GraphML-level node-edge annotations [12, 13]. This paper does not retrain a drawing parser. Instead, it uses PID2Graph GraphML as available structural evidence and studies how to retrieve and verify this evidence for downstream question answering.

### 2.2. P&ID question answering and engineering GraphRAG

PIDQA converts static P&ID drawings into queryable labeled property graphs and constructs 64,000 question answering samples, covering tasks such as counting, spatial connection, and value-based query [11]. ChatP&ID converts DEXPI smart P&ID files into knowledge graphs and supports natural language interaction through ContextRAG, VectorRAG, PathRAG, and CypherRAG [10]. These studies show that engineering drawing question answering requires structured representations and executable queries rather than relying only on raw image prompting. This paper shares with PIDQA and ChatP&ID the idea of converting engineering drawings into graph structures. The difference is that this paper uses PID2Graph GraphML and NetworkX to compare the evidence quality of dynamic graph-structured retrieval, static full-graph prompting, sparse retrieval, and dense retrieval in a controlled QA benchmark.

### 2.3. Structural VQA benchmarks

General visual question answering has evolved from image-text matching to compositional reasoning and MLLMs [18–20]. Visual Genome promotes visual relation graphs and scene graph representations through dense object, attribute, and relation annotations [21]. GQA further generates compositional questions from scene graphs and shows that metrics such as consistency, validity, and grounding are essential for structured VQA [7]. ReasonVQA introduces external structural knowledge into VQA and evaluates model degradation as reasoning depth increases through multi-hop questions [8]. These benchmarks suggest that complex drawing question answering should not report only overall answer accuracy. It should also report stratified results by question type, difficulty, and hop length, together with evidence-level metrics. PID-GraphQA therefore explicitly records the evidence path, path hops, difficulty, and unsupported claims.

### 2.4. Retrieval-augmented and neuro-symbolic reasoning

RAG enhances generation models by retrieving external evidence [22, 23], and GraphRAG further extends retrieval units to entities, relations, paths, and subgraphs [24–28]. In engineering drawings, however, retrieved evidence must satisfy topological constraints. A path is reliable evidence only when every edge in the path exists in the underlying graph. Neuro-symbolic methods emphasize combining the perception and language capabilities of neural models with the verifiability of symbolic rules [29, 30]. The symbolic part of NSG-RAG comes from GraphML nodes, edges, and NetworkX verification functions, while the neural part comes from replaceable

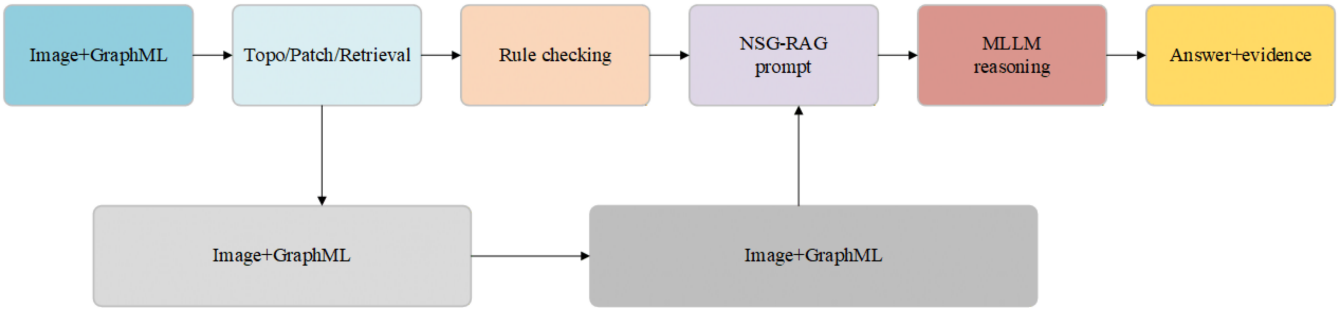



**Fig. 1. NSG-RAG framework.** The pipeline converts GraphML topology into query-conditioned evidence, verifies candidate paths with symbolic rules, and supplies compact grounded context to the LLM/MLLM.

LLM/MLLM backbones. This design moves complex drawing question answering from probabilistic text generation toward evidence-grounded topological reasoning.

## 3. Method

### 3.1. Problem definition

Given an engineering drawing image $I$, the corresponding graph structure $G = (V, E)$, and a natural language question $q$, the goal is to output an answer $\hat{a}$ and its evidence $\hat{e}$. A node $v \in V$ represents a symbol, equipment item, valve, connection point, or instrument in the drawing, and an edge $e \in E$ represents a physical or logical connection. Unlike ordinary VQA, this paper requires the evidence $\hat{e}$ to be verifiable by $G$, such as a node set, edge set, shortest path, or local subgraph.

Formally, the GraphML parser $\Phi(\cdot)$ converts the drawing annotation file $X_{\text{gml}}$ into an attributed topology graph:

$$G = \Phi(X_{\text{gml}}) = (V, E, \mathcal{A}_V, \mathcal{A}_E), \tag{1}$$

where $\mathcal{A}_V$ denotes node attributes such as node class, bounding box, and adjacency description, and $\mathcal{A}_E$ denotes edge labels and connection attributes. Each sample in PID-GraphQA can be represented as $(I, G, q, a^*, p^*)$, where $a^*$ is the gold answer and $p^* = (v_1, \ldots, v_L)$ is the gold evidence path. The model output is defined as:

$$(\hat{a}, \hat{p}) = f(I, q, G), \qquad \hat{p} = (\hat{v}_1, \ldots, \hat{v}_{\hat{L}}), \tag{2}$$

and $\hat{p}$ is required not only to overlap with $p^*$ but also to be supported by the edge set in $G$.

### 3.2. NSG-RAG workflow

NSG-RAG decomposes complex drawing question answering into five stages: drawing structural representation, graph evidence retrieval, rule verification, LLM/MLLM generation, and per-sample validation. First, the system reads PID2Graph Complete images and GraphML files, converts each drawing into a NetworkX graph, and preserves node types, edge labels, bounding boxes, and image paths. Second, source and target descriptions in the question are mapped to candidate node sets, which serve as query seeds for subsequent retrieval. Third, retrieval tools return question-relevant nodes, edges, $k$-hop subgraphs, or multi-hop paths. Fourth, the neuro-symbolic rule module checks whether candidate edges and paths are supported by the underlying GraphML topology. Finally, the system constructs a prompt from the original image, question, structural evidence, and output format constraints, feeds it into an optional LLM/MLLM backbone, and stores the model output and per-sample validation results.

### 3.3. Graph construction from PID2Graph

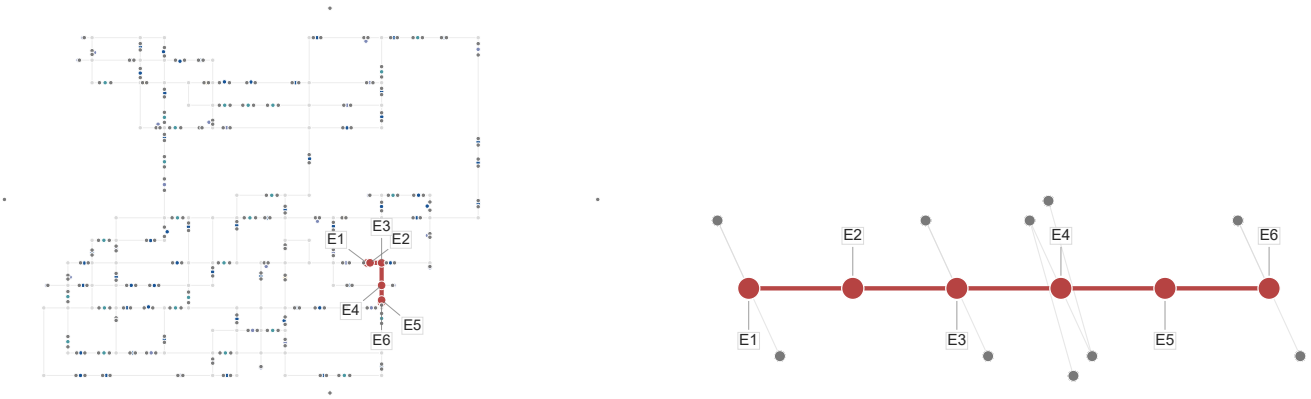
Instead of directly feeding raw engineering images or smart P&ID files into a large model, this paper treats GraphML as already available primitive and topology annotations. Each node $v$ contains an `id`, a `label`, and bounding box coordinates, representing a valve, instrument, connection point, or general symbol. Each edge $e = (u, v)$ represents a traceable topological connection in the drawing. This graph structure is used both to generate gold answers for PID-GraphQA and to retrieve and verify model outputs. To prevent the main results from degenerating into oracle lookup, prediction is not allowed to return the gold shortest path directly from clean GraphML. It must pass through grounding, retrieval evidence, and rule checking.

### 3.4. Graph retrieval tools

Following the practice in engineering GraphRAG work that decomposes graph queries into different tools [10], this paper implements four retrieval tools compatible with PID2Graph. Context-style retrieval returns candidate nodes and their adjacency context. Sparse GraphML retrieval uses BM25 over node, edge, and path documents. Dense GraphML retrieval uses TF-IDF/LSA vector retrieval over the same evidence documents. Path retrieval takes grounded source and target nodes as input and retrieves ordered multi-hop topological chains. The default NSG-RAG configuration combines Path retrieval with $k$-hop subgraph retrieval: the former provides traceable paths, while the latter provides local context and distractor node information. Figure 2 presents a retrieval example on real GraphML.

Given a question $q$ and a candidate node set $C_q$, the node grounding module first selects source and target nodes:

$$S_q = \underset{v \in C_q}{\text{TopM}} \; s_{\text{ground}}(q, v), \tag{3}$$




**(a)** Full GraphML topology; E1–E6 mark the query evidence path.

**(b)** Retrieved 2-hop subgraph with the same evidence path retained.

**Fig. 2. Question-conditioned topology retrieval.** Compact E1–E6 labels mark the same evidence path before and after retrieval, avoiding ID occlusion while showing that the retrieved subgraph removes most irrelevant topology.

where $s_{\text{ground}}(q, v)$ is computed from textual overlap among node type, drawing region, node degree, and adjacency labels. Then, $k$-hop subgraph retrieval can be written as:

$$G_q^{(k)} = G\left[\left\{u \in V \mid \min_{s \in S_q} d_G(u, s) \le k\right\}\right], \tag{4}$$

where $d_G(\cdot,\cdot)$ denotes the shortest-path distance in the graph and $G[\cdot]$ denotes the induced subgraph. If the question contains a source node $s$ and a target node $t$, path retrieval returns a candidate path:

$$\hat{p}_{s,t} = \underset{p: s \longrightarrow t}{\operatorname{argmin}} |p|. \tag{5}$$

The final evidence payload is composed of the retrieved subgraph and the path sequence:

$$E_q = \mathcal{T}\left(G_q^{(k)}\right) \cup \mathcal{T}\left(\hat{p}_{s,t}\right), \tag{6}$$

where $\mathcal{T}(\cdot)$ denotes a serialization function that converts a subgraph or path into compact JSON / textual evidence.

Table 1 summarizes the retrieval operators used in the controlled evaluation.

**Table 1**
Graph retrieval operators used in the controlled evaluation.

| Operator | Retrieval unit | Role |
|---|---|---|
| Full-graph context | Truncated whole-graph text | Static context baseline |
| BM25-RAG | Node, edge, and path documents | Sparse lexical retrieval |
| Dense-RAG | TF-IDF/LSA evidence documents | Vector-style retrieval baseline |
| $k$-hop retrieval | Induced neighborhood around grounded nodes | Local topology context |
| Path retrieval | Ordered source–target chain | Multi-hop evidence for NSG-RAG |

### 3.5. Neuro-symbolic rule verification

Before retrieval results enter the generation model, the rule module uses NetworkX to explicitly verify edges and paths. For symbol recognition questions, the system checks whether the target node appears in the retrieved subgraph. For one-hop connection questions, the system checks whether the candidate edge exists. For multi-hop path questions, the system verifies whether each consecutive edge in the path belongs to $E$. This process can be written as:

$$E_q = \text{Retrieve}(q, G), \qquad \tilde{E}_q = \text{Verify}(E_q, \mathcal{R}), \tag{7}$$

where $E_q$ is candidate evidence, $\mathcal{R}$ is a set of topological consistency rules, and $\tilde{E}_q$ is the evidence after rule checking. If a candidate path contains an edge unsupported by the graph structure, the path is not fed into the model as reliable evidence. If the model still outputs a connection or path claim, the sample is marked as an unsupported claim during per-sample validation.

For any candidate path $p = (v_1, \ldots, v_L)$, path validity is defined as:

$$\text{Valid}(p, G) = \prod_{i=1}^{L-1} \mathbf{1}\left[(v_i, v_{i+1}) \in E \vee (v_{i+1}, v_i) \in E\right], \tag{8}$$

where $\mathbf{1}[\cdot]$ is the indicator function. Only when $\text{Valid}(p, G) = 1$ can the path serve as topology evidence after rule verification. Accordingly, the generation input of NSG-RAG can be written as:

$$x_q = \text{Concat}\left(I, q, \tilde{E}_q, \text{Schema}_{\text{JSON}}\right), \tag{9}$$

and the model output is:

$$(\hat{a}, \hat{p}) = \mathcal{M}_\theta(x_q), \tag{10}$$

where $\mathcal{M}_\theta$ denotes a replaceable LLM/MLLM backbone.

### 3.6. LLM/MLLM answer generation and logging

The final prompt contains four types of information: the original image or image path, the natural language question, retrieved structural evidence, and strict JSON output constraints. Structural evidence is provided as compact JSON or a natural language summary, including node IDs, node types, edge labels, path sequences, and local subgraphs. Compared with Static Full-Graph Prompt, NSG-RAG preserves only question-relevant evidence, thereby reducing interference from irrelevant nodes and edges. Real LLM/MLLM evaluation records the prompt, raw generated content, parsed answer, gold answer, generated evidence path, gold evidence path, rule verification status, unsupported claim flag, error type, token usage, latency, and invocation cost. When platform pricing metadata is unavailable, cost-related analysis is excluded from quantitative comparison.

## 4. Dataset

### 4.1. PID2Graph dataset

This paper uses PID2Graph as the basis for the proof-of-concept study. PID2Graph, proposed by Stürmer et al., targets P&ID digitization and provides engineering images together with corresponding GraphML topology structures [12, 13]. The GraphML files contain node IDs, node types, bounding box coordinates, and edge labels, which explicitly represent symbols, connection points, valves, instruments, and connection relations. This paper treats GraphML as extracted primitive and structural information and does not retrain an end-to-end image parsing model.

PID2Graph contains two data types, Complete and Patched. Complete drawings represent full engineering drawings and are suitable for constructing global topology graphs and multi-hop path questions. Patched drawings represent local crops and can support descriptive node grounding and local visual evidence retrieval. The main experiments use Complete drawings to construct PID-GraphQA and incorporate Patched data as local grounding cues in NSG-RAG. The corresponding ablation disables this cue to examine the contribution of local Patch evidence. Figure 3 shows a real Complete drawing, a Patch, GraphML topology, and node type distribution.

### 4.2. PID-GraphQA construction

Based on the node, edge, and path information in PID2Graph, this paper constructs PID-GraphQA as a controlled evaluation benchmark. Following the question type design of PIDQA and the structured benchmark practice of GQA / ReasonVQA, PID-GraphQA records natural language questions, gold answers, evidence nodes, evidence edges, path length, candidate nodes, and distractor nodes.

The formal version no longer exposes node IDs directly as question anchors. Instead, it uses node type, drawing region, node degree, and adjacency labels to generate descriptive grounding. This design better matches the engineering review scenario in which users locate objects through descriptions and verify connections.

Table 2 summarizes the full-data construction setting used in this study.

PID-GraphQA contains five question types. Symbol recognition checks whether a model can locate a descriptively specified node and identify its class. Direct connection verifies a one-hop edge. Direct connection negative introduces nonexistent direct connections to evaluate whether a model over-claims connectivity. Multi-hop path requires an ordered topological chain. Unsupported path claim requires determining whether a given path claim is supported by the underlying GraphML.

Difficulty is stratified by path length and structural distractors: symbol recognition and one-hop questions are easy, 2–3 hop paths and some negative examples are medium, and 4–5 hop paths and unsupported path claims are hard.

Table 3 summarizes these task types and their evidence forms.

## 5. Experiments

### 5.1. Experimental settings

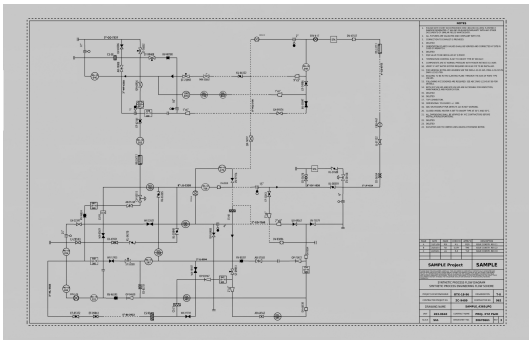
This paper conducts controlled graph-grounded evaluation on the full PID2Graph Complete drawings to assess the effect of structural evidence organization on question answering over complex engineering drawings. The main experiments focus on topology retrieval, evidence compression, and rule verification under the setting where GraphML is available. Real LLM/MLLM backbone evaluation is reported as an independent supplementary protocol.

#### *5.1.1. Benchmark construction*

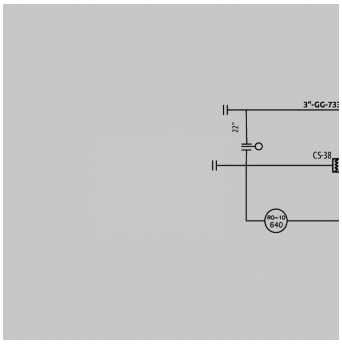
This paper constructs PID-GraphQA from 500 Complete drawings and their GraphML annotations. To avoid the oracle behavior caused by direct lookup from clean GraphML in early versions, the formal evaluation questions no longer expose only node IDs. Instead, they use node type, drawing region, adjacency context, and candidate node sets for descriptive grounding. Question types include symbol recognition, direct connection judgment, negative connection judgment, multi-hop path query, and unsupported path claim inspection. Difficulty is divided into easy, medium, and hard according to path length and structural distractors. Hard samples mainly cover medium-length multi-hop topology of 4–5 hops. A total of 2740 QA samples are generated. After a fixed random-seed split, the Test set contains 2196 samples, including 800 easy, 597 medium, and 799 hard samples.

#### *5.1.2. Baselines and fairness protocol*

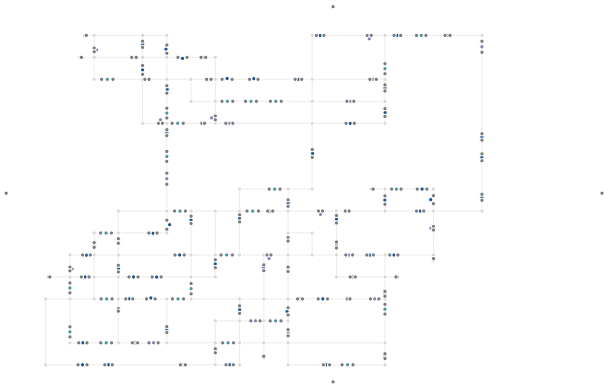
Table 4 summarizes the information accessible to each method. MLLM-only uses only the image path and question text and does not read GraphML. Text-only GraphML uses structural text and descriptive grounding without using the original image, serving as a text-only control under structural input. Static Full-Graph Prompt feeds a truncated summary of the complete GraphML. BM25-RAG and Dense-RAG use sparse retrieval and TF-IDF/LSA dense retrieval, respectively. NSG-RAG uses descriptive grounding, Patch-assisted local localization, $k$-hop/path retrieval, and NetworkX rule verification. To pre-



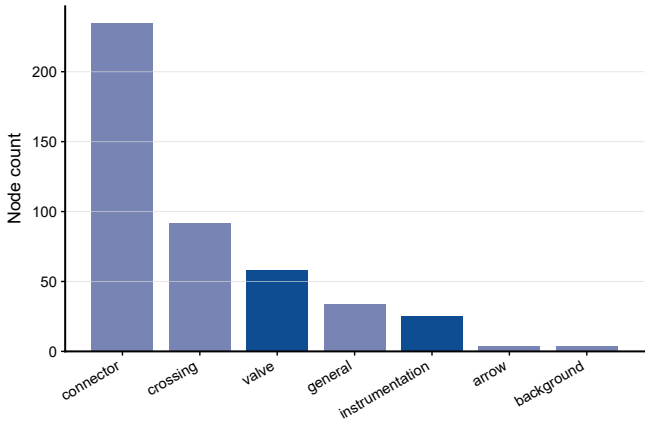
**(a)** Complete drawing.



**(b)** Local patch.



**(c)** GraphML topology.




**(d)** Node type distribution.

**Fig. 3. Representative PID2Graph example.** Complete image, local patch, GraphML topology, and node statistics.

**Table 2**
PID-GraphQA benchmark construction under the full-data evaluation setting.

| Item | Setting |
|---|---|
| Raw data | PID2Graph Complete images, Patched local images, and GraphML topology |
| Task types | Symbol recognition, one-hop connection, multi-hop path, and anomalous structure inspection |
| Difficulty labels | easy / medium / hard |
| Evidence forms | nodes, edges, shortest paths, $k$-hop subgraphs, and Patch metadata |
| Data split | Dev 20%, Test 80%, fixed random split by graph id |
| Evaluation scale | 500 Complete drawings; 2740 derived QA samples; 544 Dev samples; 2196 Test samples |
| Test difficulty distribution | 800 easy samples, 597 medium samples, and 799 hard samples |
| Question distribution | symbol recognition, direct connection, direct negative, multi-hop path, and unsupported path claim |

**Table 3**
PID-GraphQA task taxonomy on the Test split. The benchmark follows the structured-QA tradition of PIDQA and GQA by reporting both task type and evidence form.

| Question type | Goal | Main evidence | Difficulty role | Test N |
|---|---|---|---|---|
| Symbol recognition | Node label identification | node | easy | 400 |
| Direct connection | One-hop edge verification | edge | easy | 400 |
| Negative connection | Unsupported direct edge | empty / rejected edge | medium | 400 |
| Multi-hop path | Ordered topology chain | path | medium / hard | 597 |
| Unsupported path claim | Invalid path inspection | validity decision | hard | 399 |
| Difficulty split | – | – | easy / medium / hard | 800 / 597 / 799 |

vent leakage, all methods in the main results first produce retrieval evidence and then generate answers based on the evidence. They no longer directly call the shortest path from clean GraphML as the predicted answer.

### *5.1.3. LLM/MLLM backbone protocol*

Real backbone evaluation is reported separately from controlled structural evaluation. This paper keeps two OpenAI-

**Table 4**
Baseline settings in the controlled graph-grounded evaluation. "GraphML" denotes access to structural annotations.

| Method | Image | GraphML | Rule checking | Retrieval | Status |
|---|---|---|---|---|---|
| MLLM-only | ✓ | × | × | – | Visual-only baseline |
| Text-only GraphML | × | ✓ | ✓ | Budgeted text | Structural control |
| Static Full-Graph | ✓ | ✓ | ✓ | Truncated graph | Context baseline |
| BM25-RAG | ✓ | ✓ | × | Sparse documents | Retrieval baseline |
| Dense-RAG | ✓ | ✓ | × | TF-IDF/LSA | Retrieval baseline |
| NSG-RAG | ✓ | ✓ | ✓ | Path + subgraph + Patch | Proposed |

compatible Chat Completions backbones: `gpt-5.4` and `qwen3.5-plus`. Both use the same NSG-RAG evidence payload, the same JSON output constraints, and the same evaluation script. Per-sample records include output content, evidence path, rule verification, input tokens, output tokens, latency, and cost. Table 5 gives the backbone settings.

**Table 5**
LLM/MLLM backbone settings for optional real-model evaluation. The same prompt builder, evidence payload, parser, and validator are used for all backbones.

| Backbone | Role in this study |
|---|---|
| `gpt-5.4` | Primary multimodal large language model backbone using image, graph evidence, and question input |
| `qwen3.5-plus` | Additional multimodal large language model backbone using the same input protocol |

If the platform returns token usage, the cost of each call is recorded separately for input and output tokens:

$$\mathrm{Cost}_i = \frac{n_i^{\mathrm{in}}}{1000} c_{\mathrm{in}} + \frac{n_i^{\mathrm{out}}}{1000} c_{\mathrm{out}}, \tag{11}$$

where $n_i^{\mathrm{in}}$ and $n_i^{\mathrm{out}}$ denote the input and output token counts of the $i$-th sample, respectively, and $c_{\mathrm{in}}$ and $c_{\mathrm{out}}$ denote the locally configured prices per thousand tokens. Average latency is recorded as:

$$\overline{\tau} = \frac{1}{N} \sum_{i=1}^{N} \tau_i. \tag{12}$$

When pricing metadata is unavailable, cost columns are excluded from quantitative comparison. Character counts and token usage are still used to analyze the context overhead of different backbones.

### *5.1.4. Evaluation metrics*

This paper organizes the main metrics from three perspectives: answer correctness, evidence traceability, and multi-hop topological reasoning. Consistent with the evaluation practices of GQA, ReasonVQA, and PIDQA, the main tables report ACC, Evi-F1, Evidence Coverage (Evi-Cov), and MH-Acc. Unsupported Claim Rate (UCR) is not used as a main performance metric. It is instead reported in Error Analysis / Claim Validation Analysis to analyze whether a model produces connection or path claims unsupported by the graph structure.

ACC measures whether the final answer matches the gold answer and applies to symbol type recognition, direct connection judgment, and path questions. Evi-F1 is based on overlap between the predicted evidence path and the gold path at the node-set level, considering both evidence precision and recall. This metric penalizes long evidence that contains many irrelevant nodes. Evidence Coverage (Evi-Cov) measures the proportion of gold evidence nodes covered by the predicted path and focuses on whether the model finds the key supporting nodes. MH-Acc is computed only on `multi_hop_path` samples with path length of at least 2 hops, thereby isolating the ability to perform topological chain reasoning. The denominator of UCR includes only samples with explicit graph claims, namely cases in which the model explicitly claims that a connection or path exists. This metric is used only for error analysis and cannot alone represent overall question answering ability.

Answer accuracy is defined as:

$$\mathrm{ACC} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[\mathrm{norm}(\hat{a}_i) = \mathrm{norm}(a_i^*)\right], \tag{13}$$

where $\mathrm{norm}(\cdot)$ denotes case and whitespace normalization. For the set $\mathcal{D}_e$ with non-empty gold evidence paths, evidence coverage is:

$$\mathrm{EviCov} = \frac{1}{|\mathcal{D}_e|} \sum_{i \in \mathcal{D}_e} \frac{|\hat{P}_i \cap P_i^*|}{|P_i^*|}, \tag{14}$$

where $\hat{P}_i$ and $P_i^*$ denote the node sets of the predicted path and the gold path, respectively. Evidence F1 is further defined as:

$$\mathrm{EviF1} = \frac{1}{|\mathcal{D}_e|} \sum_{i \in \mathcal{D}_e} \frac{2\,\mathrm{Prec}_i\,\mathrm{Rec}_i}{\mathrm{Prec}_i + \mathrm{Rec}_i + \epsilon}, \tag{15}$$

where $\mathrm{Prec}_i = |\hat{P}_i \cap P_i^*|/|\hat{P}_i|$, $\mathrm{Rec}_i = |\hat{P}_i \cap P_i^*|/|P_i^*|$, and $\epsilon$ avoids division by zero. Multi-hop accuracy is computed only on the multi-hop subset $\mathcal{D}_{mh}$:

$$\mathrm{MHAcc} = \frac{1}{|\mathcal{D}_{mh}|} \sum_{i \in \mathcal{D}_{mh}} \mathbf{1}\left[\mathrm{norm}(\hat{a}_i) = \mathrm{norm}(a_i^*)\right]. \tag{16}$$

Unsupported claim rate is defined as:

$$\mathrm{UCR} = \frac{\sum_{i \in C} \mathbf{1}\left[\mathrm{Valid}(\hat{p}_i, G_i) = 0\right]}{|C|}, \tag{17}$$

where $C$ is the set of samples with explicit connection or path claims.

These metrics are complementary. ACC reflects usability at the answer level. Evi-F1 and Evi-Cov reflect traceability at the evidence level. MH-Acc focuses on complex topological reasoning. UCR and claim count (Claim-N) are used only in the subsequent claim validation analysis to measure the risk of unsupported claims, which is sensitive in engineering review scenarios. For real LLM/MLLM backbone evaluation, this paper also records input characters, input tokens, output tokens, response latency, and invocation cost. These metrics are not used to rank methods in controlled structural evaluation, but they can support later analysis of trade-offs between cost and accuracy across different backbones.

## 5.2. Experimental results

### *5.2.1. Main results*

Table 6 and Figure 4 present the results on the full Test split.

The formal evaluation uses descriptive grounding, negative examples, and medium-length multi-hop questions, thereby avoiding oracle behavior from direct lookup in clean GraphML by task design. Under the current controlled evaluation, NSG-RAG achieves 0.902 ACC, 0.632 Evi-F1, and 0.678 MH-Acc, indicating that path retrieval and rule verification improve both answer-level and evidence-level performance.

Different baselines have different error sources. MLLM-only lacks explicit topology input and therefore struggles to answer questions that depend on connectivity, leading to low overall accuracy. Text-only GraphML can recover some answers from structural text, but its evidence organization is affected by context budget and node ordering, and its multi-hop path score is clearly lower than that of NSG-RAG. Static Full-Graph feeds the complete graph structure in one pass and is easily disturbed by irrelevant nodes and edges. BM25-RAG and Dense-RAG can retrieve local evidence, but their retrieval units are mainly based on textual similarity and cannot stably preserve ordered paths. In contrast, NSG-RAG retrieves path-level evidence and performs topological verification before and after generation, producing a more balanced result across answer accuracy, evidence quality, and multi-hop reasoning.

Supplementary real-backbone results are shown in Table 7. This table is reported separately from controlled structural evaluation and presents the invocation results of different LLM/MLLM backbones under the same NSG-RAG evidence input. Because external API evaluation is affected by invocation cost, rate limits, image compression, and output format stability, this paper treats it as a supplementary protocol rather than the basis for the main comparison. A more complete future backbone evaluation can further report token cost, latency, and format-following ability across different models.

### *5.2.2. Question-type analysis*

Table 8 reports answer accuracy by PID-GraphQA question type.

This analysis follows task-category reporting in PIDQA and the practice of observing model behavior by compositional question type in GQA. The results show that different methods have different strengths across tasks. Text-only GraphML and Static Full-Graph perform strongly on negative judgments, indicating that complete structural text helps identify some disconnected samples.

However, both degrade substantially on multi-hop path questions because they lack ordered path retrieval. BM25-RAG and Dense-RAG recover some node-related evidence, but they are unstable across direct connection and multi-hop path tasks. NSG-RAG maintains higher accuracy on direct connection and multi-hop path tasks, showing that path-level evidence is more suitable than ordinary textual similarity retrieval for topological question answering.

### *5.2.3. Difficulty analysis*

Figure 5 presents results stratified by easy, medium, and hard difficulty levels.

Easy samples mainly include symbol recognition and one-hop connection judgment, for which models can obtain partial answers from local node descriptions or short adjacency information. Medium and hard samples introduce longer paths, stronger node ambiguity, and negative connections. In these cases, models must identify source and target nodes and preserve ordered topological chains.

MLLM-only can use visual semantic cues in the question wording for easy symbol questions, but it lacks support for questions that require topological evidence. BM25-RAG and Dense-RAG can retrieve some local evidence, but their retrieval results are often discrete nodes or edge fragments and do not sufficiently preserve path order. NSG-RAG remains relatively stable on hard samples, indicating that path-level retrieval and rule verification support medium-length topological reasoning. This result also shows that difficulty stratification reveals structural reasoning differences beyond overall accuracy.

To further align with the reasoning-depth evaluation of ReasonVQA, Table 9 groups only multi-hop path samples by path length.

The current PID-GraphQA mainly contains 2–3 hop and 4–5 hop paths and does not yet cover paths longer than 6 hops at scale. The results show that NSG-RAG clearly outperforms baselines that do not explicitly model paths on both 2–3 hop and 4–5 hop subsets. BM25-RAG and Dense-RAG can recover some answers on short paths, but they degrade more clearly on 4–5 hop paths. This observation supports the design of path-level retrieval in this paper.

### *5.2.4. Ablation study*

Table 10 and Figure 6 show component ablations.

After rule verification is removed, NSG-RAG w/o Rules can still retrieve some question-relevant nodes, so Evi-F1 does not completely collapse. However, because edge-level legality checking is absent, this variant is more likely to produce unsupported path claims, showing that path verification is a key component for suppressing unsupported claims. After multi-

**Table 6**
Main results on the full PID-GraphQA Test split. ACC, Evi-F1, Evi-Cov, and MH-Acc evaluate answer correctness, evidence quality, and multi-hop topology reasoning.

| Method | ACC | Evi-F1 | Evi-Cov | MH-Acc | N |
|---|---|---|---|---|---|
| MLLM-only | 0.182 | 0.000 | 0.000 | 0.000 | 2196 |
| Text-only GraphML | 0.698 | 0.303 | 0.304 | 0.032 | 2196 |
| Static Full-Graph | 0.604 | 0.185 | 0.185 | 0.000 | 2196 |
| BM25-RAG | 0.484 | 0.220 | 0.220 | 0.077 | 2196 |
| Dense-RAG | 0.378 | 0.188 | 0.188 | 0.114 | 2196 |
| NSG-RAG | **0.902** | **0.632** | **0.649** | **0.678** | 2196 |

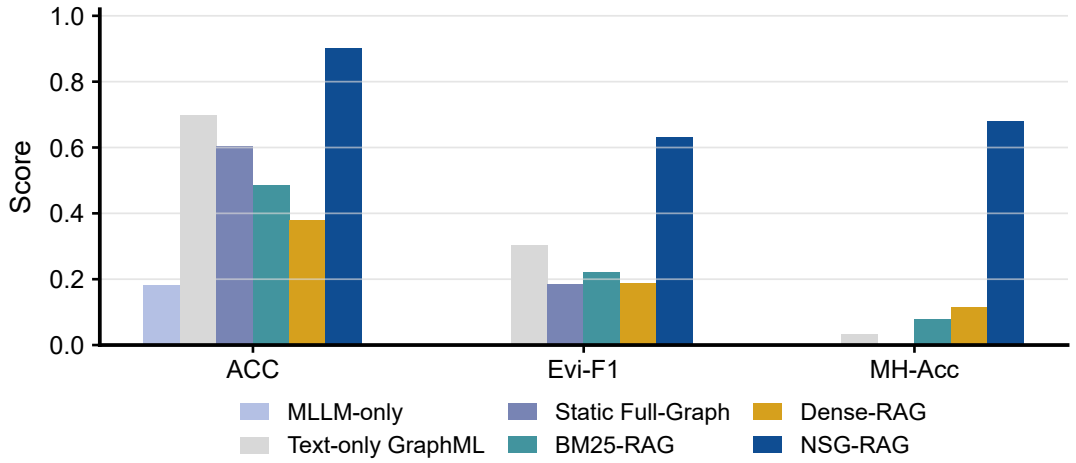



**Fig. 4. Controlled evaluation on the full Test split.** NSG-RAG improves answer accuracy, evidence quality, and multi-hop topology reasoning under the same PID-GraphQA protocol.

**Table 7**
Backbone evaluation results and recorded generation-efficiency indicators under the same NSG-RAG evidence payload.

| Backbone | ACC | Evi-F1 | Evi-Cov | MH-Acc | UCR | In-Tok | Out-Tok | Lat. (s) | N |
|---|---|---|---|---|---|---|---|---|---|
| `gpt-5-nano` | 0.520 | 0.730 | 0.731 | 0.978 | 0.000 | 2459 | 620 | 12.8 | 500 |
| `qwen3.5-plus` | 0.840 | 1.000 | 1.000 | 1.000 | 0.000 | 2861 | 1536 | 118.0 | 500 |

**Table 8**
ACC by PID-GraphQA question type. This fine-grained view follows compositional VQA benchmarks where aggregate accuracy alone is insufficient.

| Question type | N | MLLM | Text | FullGraph | BM25 | Dense | NSG-RAG |
|---|---|---|---|---|---|---|---|
| Symbol recognition | 400 | 1.000 | 1.000 | 0.993 | 0.990 | 0.728 | **1.000** |
| Direct connection | 400 | 0.000 | 0.787 | 0.357 | 0.140 | 0.060 | **0.940** |
| Negative connection | 400 | 0.000 | 0.998 | 0.968 | 0.412 | 0.117 | **1.000** |
| Multi-hop path | 597 | 0.000 | 0.032 | 0.000 | 0.077 | 0.114 | **0.678** |
| Unsupported path claim | 399 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 | **1.000** |

**Table 9**
Multi-hop path accuracy by hop length. Hop-stratified analysis follows ReasonVQA-style evaluation of structural reasoning depth.

| Hop length | N | MLLM | Text | FullGraph | BM25 | Dense | NSG-RAG |
|---|---|---|---|---|---|---|---|
| 2–3 | 197 | 0.000 | 0.005 | 0.000 | 0.234 | 0.310 | **0.690** |
| 4–5 | 400 | 0.000 | 0.045 | 0.000 | 0.000 | 0.018 | **0.672** |
| ≥6 | 0 | – | – | – | – | – | – |

hop path retrieval is removed, the model can still answer some questions using local $k$-hop subgraphs, but MH-Acc drops to 0, indicating that local neighborhoods alone are insufficient for stable ordered multi-hop question answering.

**Table 10**
Ablation results on the full PID-GraphQA Test split. The variants isolate rule checking, path retrieval, graph structure, and Patch evidence.

| Variant | ACC | Evi-F1 | Evi-Cov | MH-Acc | Interpretation |
|---|---|---|---|---|---|
| NSG-RAG | 0.902 | 0.632 | 0.649 | 0.678 | Path retrieval with rule checking |
| NSG-RAG w/o Rules | 0.297 | 0.546 | 0.553 | 0.002 | Rule checking disabled |
| w/o Multi-hop Retrieval | 0.717 | 0.324 | 0.324 | 0.000 | Ordered path retrieval disabled |
| w/o Edge/Path Evidence | 0.432 | 0.186 | 0.186 | 0.000 | Node-only evidence |
| w/o Graph Structure | 0.182 | 0.000 | 0.000 | 0.000 | No explicit topology |
| w/o Patch Evidence | 0.718 | 0.171 | 0.191 | 0.201 | Patch evidence disabled |

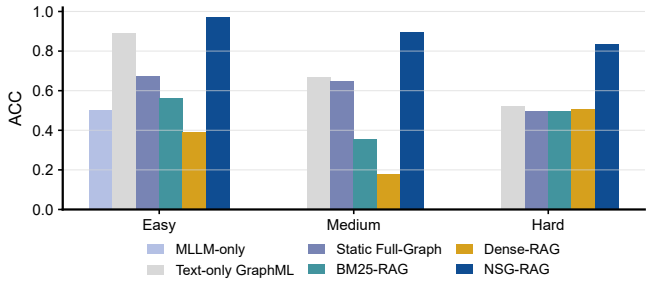



**Fig. 5. Difficulty-stratified accuracy.** NSG-RAG remains more stable as questions require longer paths and stronger structural grounding.

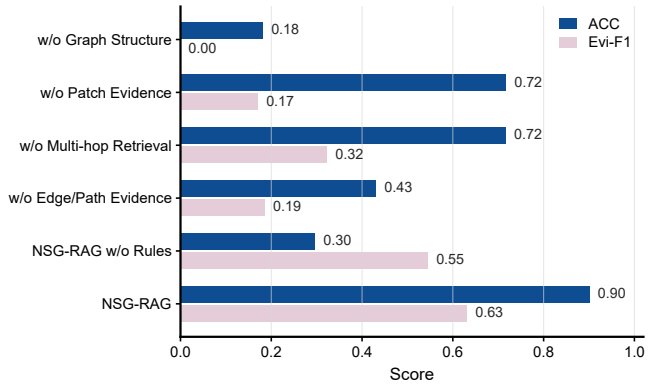
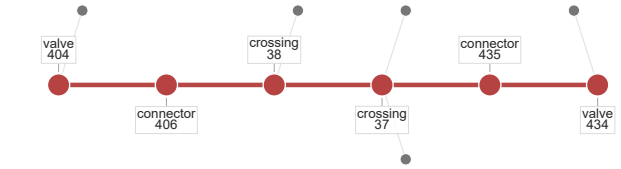



**Fig. 6. Ablation comparison.** Removing path retrieval, graph structure, Patch evidence, or rule checking changes answer and evidence-aware scores.

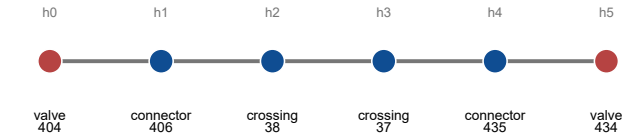
When Patch evidence is disabled, descriptive grounding is more likely to drift under visual ambiguity among candidate nodes, causing answer accuracy and evidence metrics to decrease together. Keeping only node evidence or removing graph structure further weakens topological reasoning ability, showing that complex engineering drawing question answering requires not only node type recognition but also the correspondence among edges, paths, and local visual cues. Overall, the ablation study supports the core hypothesis of this paper: the gains of NSG-RAG come from the combination of structural retrieval, path evidence, and rule verification rather than a single module.

### *5.2.5. Case study*

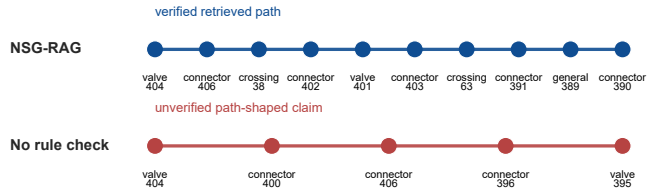
Figure 7 shows a real multi-hop evidence case derived from PID2Graph GraphML annotations.




**(a)** Local topology around the evidence path.




**(b)** Ordered gold evidence chain.

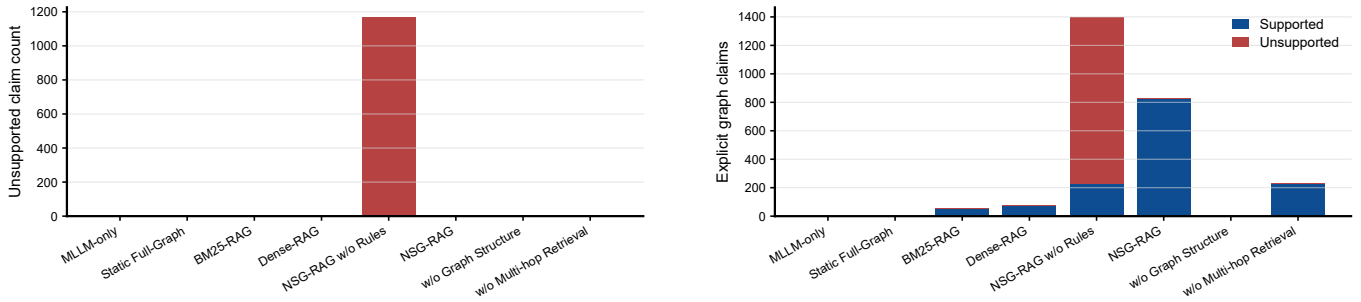



**(c)** Retrieved NSG-RAG evidence and unverified path-shaped claim.

**Fig. 7. Representative multi-hop evidence case.** Local topology, gold path, and model evidence comparison.

The highlighted nodes and edges correspond to the topological chain used to answer the question. The side panel gives the gold path, the path retrieved by NSG-RAG, and a path-shaped claim that has not been confirmed by rules. This case shows that generating a plausible node sequence is not sufficient for reliable evidence. Every consecutive edge in the path must be verifiable in the underlying GraphML. This figure is a representative case analysis and does not represent the overall distribution.

Table 11 summarizes the representative case study and the role of graph-grounded evidence validation.




**(a)** Unsupported claim counts.

**(b)** Explicit graph claims.

**Fig. 8. Unsupported evidence claim analysis.** The panels separate unsupported claim counts from the support status of explicit graph claims.

**Table 11**
Representative case study with graph-grounded evidence. The exact node sequence depends on the selected full-data sample and is visualized in Figure 7.

| Item | Description |
|---|---|
| Question type | Multi-hop path / unsupported evidence inspection |
| Gold evidence | Ordered path from clean GraphML annotation |
| Unverified claim | Path-shaped output before topology validation |
| NSG-RAG | Accepts only evidence that passes topology validation on the inference graph |
| Use in paper | Illustrates evidence traceability rather than aggregate performance |

### *5.2.6. Error analysis and limitations*

Figure 8 summarizes unsupported claims.

Errors mainly come from eight sources: node grounding failure, full graph truncation failure, path retrieval failure, unsupported path claim, evidence-answer mismatch, GraphML noise, output format error, and visual grounding absent. Node grounding failure means that the question description is mapped to an incorrect node. Full graph truncation failure means that complete-graph prompting loses critical evidence after truncation or ordering. Path retrieval failure means that the retrieved nodes are related to the question but cannot form the correct path. Unsupported path claim means that the model outputs a connection relation that does not exist in the graph structure.

The current results still have limitations. First, GraphML is treated as accessible structural input, and end-to-end drawing parsing errors are not yet covered. Second, Patch evidence is currently used mainly as an auxiliary cue for local grounding and has not yet been extended into full visual reasoning based on local image crops read by a real MLLM. Third, BM25 and TF-IDF/LSA Dense-RAG are reproducible experimental controls and cannot replace industrial embedding models or closed-source MLLM evaluation. Fourth, large-scale API evaluation of real backbones such as `gpt-5.4` and `qwen3.5-plus` should be reported as a separate follow-up experiment. Despite these limitations, the current controlled results show that using graph structure as a retrievable evidence space provides a clearer verification boundary for question answering over complex engineering drawings.

## 6. Conclusion

This paper proposes NSG-RAG as a proof-of-concept framework for visual question answering over complex engineering drawings without assuming that end-to-end drawing parsing has already been fully solved. The method treats the GraphML topology of PID2Graph as a retrievable evidence space and provides traceable context to an LLM/MLLM through local subgraphs, multi-hop paths, and rule verification. Compared with visual-only input, static full-graph prompting, and general text retrieval, NSG-RAG achieves better answer accuracy, evidence consistency, and multi-hop reasoning ability in full controlled evaluation.

The main value of this paper is to move engineering drawing question answering from answer generation alone toward evidence-driven structural verification. Joint evaluation with ACC, Evi-F1, Evi-Cov, and MH-Acc shows that topology retrieval affects not only final answers but also the completeness of evidence chains. UCR and claim count (Claim-N) serve as error analysis metrics for checking whether path claims are supported by the underlying graph structure. The real-backbone evaluation protocol further records token usage, latency, cost, and per-sample rule verification, supporting future comparisons among commercial APIs and smaller models. Overall, NSG-RAG provides an extensible research direction for review-oriented recognition, topology retrieval, and knowledge-driven question answering over complex engineering drawings.

## Funding

This research was funded by State Grid Shanghai Economic Research Institute (SGTYHT/23-JS-004).

## Author Contributions

**Weiguo Shi**: Conceptualization, Supervision, Resources, Project administration, Writing – review & editing. **Peiling Zhuang**: Conceptualization, Methodology, Supervision, Formal analysis, Project administration, Writing – review & editing, Correspondence. **Aili Pang**: Methodology, Validation, Investigation, Data curation, Writing – review & editing. **Zilei Wang**: Software, Validation, Formal analysis, Visualization, Writing – review & editing. **Chanjuan Tang**: Data cura-

tion, Investigation, Resources, Validation, Writing – review & editing.

## Data Availability Statement

Data will be made available on request.

## Conflict of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References


[1] Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning Transferable Visual Models from Natural Language Supervision. In Proceedings of the International Conference on Machine Learning (ICML), 2021, Vol. 139, pp. 8748–8763. https://doi.org/10.48550/arXiv.2103.00020.

[2] Li, J.; Li, D.; Savarese, S.; Hoi, S.C.H. BLIP-2: Bootstrapping Language-Image Pre-Training with Frozen Image Encoders and Large Language Models. In Proceedings of the International Conference on Machine Learning (ICML), 2023, Vol. 202, pp. 19730–19742. https://doi.org/10.48550/arXiv.2301.12597.

[3] Liu, H.; Li, C.; Wu, Q.; Lee, Y.J. Visual Instruction Tuning. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS), 2023, Vol. 36, pp. 34892–34916. https://doi.org/10.48550/arXiv.2304.08485.

[4] Bai, S.; Chen, K.; Liu, X.; Wang, J.; Ge, W.; Song, S.; Dang, K.; Wang, P.; Wang, S.; Tang, J.; et al. Qwen2.5-VL Technical Report. arXiv preprint arXiv:2502.13923, 2025. https://doi.org/10.48550/arXiv.2502.13923.

[5] Zhu, J.; Wang, W.; Chen, Z.; Liu, Z.; Ye, S.; Gu, L.; Tian, H.; Duan, Y.; Su, W.; Shao, J.; et al. InternVL3: Exploring Advanced Training and Test-Time Recipes for Open-Source Multimodal Models. arXiv preprint arXiv:2504.10479, 2025. https://doi.org/10.48550/arXiv.2504.10479.

[6] OpenAI. GPT-4o System Card. arXiv preprint arXiv:2410.21276, 2024. https://doi.org/10.48550/arXiv.2410.21276.

[7] Hudson, D.A.; Manning, C.D. GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 6700–6709. https://doi.org/10.1109/CVPR.2019.00686.

[8] Tran, D.T.; Tran, T.K.; Hauswirth, M.; Phuoc, D.L. ReasonVQA: A Multi-Hop Reasoning Benchmark with Structural Knowledge for Visual Question Answering. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2025, pp. 18793–18803. https://doi.org/10.1109/ICCV51701.2025.01746.

[9] Liu, N.F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; Liang, P. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics* **2024**, *12*, 157–173. https://doi.org/10.1162/tacl_a_00638.

[10] Alimin, A.A.; Schweidtmann, A.M. GraphRAG for Engineering Diagrams: ChatP&ID Enables LLM Interaction with P&IDs. arXiv preprint arXiv:2603.22528, 2026. https://doi.org/10.48550/arXiv.2603.22528.

[11] Gupta, M.; Wei, C.; Czerniawski, T.; Eiris, R. PIDQA—Question Answering on Piping and Instrumentation Diagrams. *Machine Learning and Knowledge Extraction* **2025**, *7*, 39. https://doi.org/10.3390/make7020039.

[12] Stürmer, J.M.; Graumann, M.; Koch, T. From Engineering Diagrams to Graphs: Digitizing P&IDs with Transformers. In Proceedings of the 2025 IEEE 12th International Conference on Data Science and Advanced Analytics (DSAA), 2025, pp. 1–11. https://doi.org/10.1109/DSAA65442.2025.11248012.

[13] Stürmer, J.M.; Graumann, M.; Koch, T. PID2Graph. Zenodo, 2025. https://doi.org/10.5281/zenodo.14803338.

[14] Yu, E.S.; Cha, J.M.; Lee, T.; Kim, J.; Mun, D. Features Recognition from Piping and Instrumentation Diagrams in Image Format Using a Deep Learning Network. *Energies* **2019**, *12*, 4425. https://doi.org/10.3390/en12234425.

[15] Kim, H.; Lee, W.; Kim, M.; Moon, Y.; Lee, T.; Cho, M.; Mun, D. Deep-Learning-Based Recognition of Symbols and Texts at an Industrially Applicable Level from Images of High-Density Piping and Instrumentation Diagrams. *Expert Systems with Applications* **2021**, *183*, 115337. https://doi.org/10.1016/j.eswa.2021.115337.

[16] Moon, Y.; Lee, J.; Mun, D.; Lim, S. Deep Learning-Based Method to Recognize Line Objects and Flow Arrows from Image-Format Piping and Instrumentation Diagrams for Digitization. *Applied Sciences* **2021**, *11*, 10054. https://doi.org/10.3390/app112110054.

[17] Qu, N.; Xue, J.; Gao, S. Automatic Recognition System for P&ID Diagrams in Distributed Control Systems. In Proceedings of the 2025 IEEE International Conference on Computational Intelligence and Smart Application Technology (CISAT), 2025, pp. 674–678. https://doi.org/10.1109/CISAT66811.2025.11181755.

[18] Antol, S.; Agrawal, A.; Lu, J.; Mitchell, M.; Batra, D.; Zitnick, C.L.; Parikh, D. VQA: Visual Question Answering. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2425–2433. https://doi.org/10.1109/ICCV.2015.279.

[19] Goyal, Y.; Khot, T.; Agrawal, A.; Summers-Stay, D.; Batra, D.; Parikh, D. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. *International Journal of Computer Vision* **2019**, *127*, 398–414. https://doi.org/10.1007/s11263-018-1116-0.

[20] Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Zitnick, C.L.; Girshick, R. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1988–1997. https://doi.org/10.1109/CVPR.2017.215.

[21] Krishna, R.; Zhu, Y.; Groth, O.; Johnson, J.; Hata, K.; Kravitz, J.; Chen, S.; Kalantidis, Y.; Li, L.J.; Shamma, D.A.; et al. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *International Journal of Computer Vision* **2017**, *123*, 32–73. https://doi.org/10.1007/s11263-016-0981-7.

[22] Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; tau Yih, W.; Rocktäschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS), 2020, Vol. 33, pp. 9459–9474. https://doi.org/10.48550/arXiv.2005.11401.

[23] Karpukhin, V.; Oguz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; tau Yih, W. Dense Passage Retrieval for Open-Domain Question Answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550.

[24] Edge, D.; Trinh, H.; Cheng, N.; Bradley, J.; Chao, A.; Mody, A.; Truitt, S.; Metropolitansky, J.; Ness, R.O.; Larson, J. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv preprint arXiv:2404.16130, 2024. https://doi.org/10.48550/arXiv.2404.16130.

[25] Guo, Z.; Xia, L.; Yu, Y.; Ao, T.; Huang, C. LightRAG: Simple and Fast Retrieval-Augmented Generation. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2025, 2025, pp. 10746–10761. https://doi.org/10.18653/v1/2025.findings-emnlp.568.

[26] Yu, J.; Liu, Y.; Gu, J.; Torr, P.; Zhou, D. Can Knowledge-Graph-Based Retrieval Augmented Generation Really Retrieve What You Need? In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS), 2025. https://doi.org/10.48550/arXiv.2510.16582.

[27] Hsiao, C.H.; Wang, Y.C.; Lin, T.S.; Yeh, Y.R.; Chen, C.S. MegaRAG: Multimodal Knowledge Graph-Based Retrieval Augmented Generation. arXiv preprint arXiv:2512.20626, 2025. https://doi.org/10.48550/arXiv.2512.20626.

[28] Yuan, X.; Ning, L.; Ye, Q.; Fan, W.; Li, Q. mKG-RAG: Leveraging Multimodal Knowledge Graphs in Retrieval-Augmented Generation for Knowledge-Intensive VQA. In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 2026. https://doi.org/10.1145/3805712.3809680.

[29] Chen, S.; Yuan, Z.; Zhang, Q.; Hua, W.; Cao, J.; Huang, X. NeuSymEA: Neuro-Symbolic Entity Alignment via Variational Inference. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS), 2025, Vol. 38. https://doi.org/10.48550/arXiv.2410.04153.

[30] Schmidt, W.J.; Rincon-Yanez, D.; Kharlamov, E.; Paschke, A. LLMs on the Rise: Neuro-Symbolic AI for Knowledge Graph Construction in Manufacturing: Systematic Literature Review. *IEEE Access* **2026**, *14*, 28383–28410. https://doi.org/10.1109/ACCESS.2026.3665999.