SUB-PLANNER: A GENERALIST AGENT FRAMEWORK FOR MULTI-DOMAIN REASONING VIA ADAPTIVE SUB-PROBLEM PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Domain-specific downstream tasks often require professional knowledge and multi-step reasoning to derive reliable solutions in specialized scenarios. Multimodal agents offer a promising solution by enabling MLLMs to perform multi-step reasoning and leverage external tools. However, most existing agents are typically built on general models and may lack sufficient domain knowledge. Domain-specific agents often rely on reasoning and tool-use pipelines tailored to specialized domains, limiting their transferability across domains. To address these limitations, we propose a general training-free agent framework that generalizes across diverse specialized domains. Our framework combines a general MLLM for planning and verification with a domain-specific MLLM for answering specialized questions. In the planning stage, the general model adaptively generates sub-problems based on the original task and selects appropriate domain-specific tools to collect supporting evidence. In the answering stage, the domain-specific model then performs reasoning on these sub-problems based on the collected evidence. In the verification stage, the general model further verifies the consistency between tool outputs and sub-problem answers, retaining only validated evidence for final reasoning. Our framework provides a unified training-free solution that requires no domain-specific adaptation and can flexibly integrate various general and domain-specific MLLMs for diverse downstream tasks. Experimental results show that our method achieves strong performance across 17 diverse domains.

1 INTRODUCTION

Domain-specific downstream tasks, including medical image understanding (Pan et al., 2025; Deria et al., 2026), financial reasoning (Gan et al., 2025; Huang et al., 2024), robotic manipulation (Luo et al., 2025; Tan et al., 2026), geospatial analysis (Kuckreja et al., 2024; Danish et al., 2025), pathology image interpretation (Dai et al., 2024; He et al., 2020), and materials characterization (Buehler, 2024; Lai et al., 2025), are widely encountered in real-world scenarios. Solving these tasks typically requires professional knowledge, multi-step reasoning, and tool-use capability. These requirements make it challenging for existing models to handle domain-specific downstream tasks.

Multimodal agents (Fan et al., 2024; Wang et al., 2024; Szot et al., 2025) provide an effective solution for addressing such domain-specific downstream tasks. By invoking external tools, they extend the capabilities of MLLMs beyond end-to-end inference, enabling them to handle tasks that require external knowledge and multi-step reasoning. Existing multimodal agents (Ning et al., 2026; Zhan et al., 2026; Li et al., 2025a) are typically built on general MLLMs, which provide task planning and reasoning capabilities. Benefiting from large-scale and diverse training data, general MLLMs (Liu et al., 2023; Chen et al., 2024; Zhu et al., 2025; Bai et al., 2025b) possess broad knowledge and reasoning abilities. However, these models are mainly trained on open-domain data and may lack the domain-specific knowledge required for specialized tasks, leading to inaccurate intermediate judgments or final predictions even with appropriate reasoning processes.

Domain-specific multimodal agents (Xu et al., 2025; Fallahpour et al., 2025) are developed for specialized domains by incorporating domain-specific MLLMs (Tan et al., 2026; Xu et al., 2026a), or tailored reasoning procedures. These agents can draw on specialized knowledge and relevant

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.

evidence to solve domain-specific tasks. However, existing approaches (Wang et al., 2025b; Xu et al., 2026b; Shao et al., 2026; Lin et al., 2025) are often tightly coupled with predefined reasoning procedures and tool-use pipelines designed for specific domains. Since different domains require distinct types of analysis and evidence, these pipelines are difficult to transfer directly. For example, medical image analysis requires lesion localization and severity assessment, industrial inspection focuses on structural defects and texture variations, and remote sensing involves reasoning about spatial relationships among objects. These differences make it difficult to directly transfer existing domain-specific agent pipelines across domains.

We observe that many complex tasks in specialized domains can often be decomposed into multiple simpler sub-problems. Solving these sub-problems can provide meaningful intermediate evidence and support the final decision-making process. For challenging questions that domain-specific models cannot directly solve, carefully designed sub-problems can guide the reasoning process toward more accurate answers, as shown in Fig. 3 in the experimental section. Motivated by this observation, we propose a general agent framework that generalizes across diverse specialized domains.

Our agent framework consists of two key components: a general model and a domain-specific model. General models (Bai et al., 2025b; Zhu et al., 2025; Li et al., 2024) possess strong reasoning and planning capabilities, allowing them to generate sub-problems based on the characteristics of different tasks, while domain-specific models (Pan et al., 2025; Gu et al., 2024; Kuckreja et al., 2024) provide domain knowledge for accurately solving these sub-problems. In our framework, the general model is responsible for planning and verification by leveraging its reasoning and planning capabilities, while the domain-specific model focuses on answering domain-specific questions using its acquired domain knowledge. For the planning stage, the general model adaptively generates auxiliary sub-problems based on the original task. These sub-problems are designed to provide additional evidence for solving the final task, and the general model invokes appropriate external tools according to the requirements of each sub-problem to collect supporting evidence. For the answering stage, the domain-specific model performs reasoning on these sub-problems based on the collected evidence. For the verification stage, the general model verifies the consistency between tool outputs and sub-problem answers, retaining only validated evidence for final reasoning. Finally, the domain-specific model integrates all validated sub-problem answers to obtain the final prediction.

Compared with existing domain-specific agents (Xu et al., 2025; Fallahpour et al., 2025; Lin et al., 2025), our framework can be directly applied to diverse domain-specific downstream tasks without modifying the reasoning and tool-use pipeline for individual domains. By flexibly integrating different general and domain-specific models, our framework enables effective knowledge transfer across domains. Moreover, the collaboration between general and domain-specific models allows the framework to leverage general reasoning capabilities while incorporating domain-specific knowledge for specialized tasks. In summary, our contributions are as follows:

- **General agent framework for different domain tasks:** We propose a general agent framework that enables adaptive reasoning across diverse domains through a unified pipeline. Unlike existing domain-specific agents, our framework does not require modifications to the existing framework for different tasks.
- **Adaptive sub-problem generation and verification:** We introduce an adaptive sub-problem generation mechanism that decomposes complex domain-specific tasks into multiple relatively simpler sub-problems and leverages domain-specific tools to collect supporting evidence. Furthermore, we design an evidence verification mechanism to filter unreliable information by verifying the consistency between tool outputs and intermediate answers, improving the reliability of multi-stage reasoning.
- **Strong generalization across diverse domains:** Extensive experiments on downstream tasks spanning 17 diverse domains demonstrate that our framework achieves strong performance and generalizes effectively across different specialized scenarios.

2 Related Work

Multimodal Agent. Multimodal agents extend MLLMs with task planning, tool use, and environment interaction, enabling multi-step multimodal reasoning. Existing works explore applications such as video understanding (Fan et al., 2024; Wang et al., 2024), GUI interaction (Li et al., 2025a;

Xu et al., 2025), embodied reasoning, multimodal search, and world-model-based planning (Szot et al., 2025; Ning et al., 2026; Zhan et al., 2026). Domain-specific agents have also been developed for medical reasoning (Fallahpour et al., 2025), autonomous driving (Lin et al., 2025; Wang et al., 2025b), and remote sensing (Shao et al., 2026), typically relying on specialized tools or models. However, these components are often tailored to individual domains and are difficult to transfer directly. In contrast, our framework uses a general model for task planning, sub-problem generation, tool selection, and verification, while leveraging domain-specific models to provide specialized knowledge for individual sub-problems.

Domain-Specific Multimodal Models. Domain-specific multimodal models are developed to complement the limited specialized knowledge of general-purpose models through domain-specific data and supervision. Existing studies cover a wide range of fields, including agriculture (Nawaz et al., 2025; Boudiaf et al., 2026), industrial anomaly understanding (Gu et al., 2024; Li et al., 2026; Zhou et al., 2024), medical image analysis (Deria et al., 2026; Pan et al., 2025; Dai et al., 2024), remote sensing (Kuckreja et al., 2024), chemistry (Li et al., 2025b), finance (Huang et al., 2024), and embodied reasoning (Tan et al., 2026). These models generally provide stronger domain knowledge and fine-grained perception within their target domains. However, they are typically optimized for specific domains or task distributions, with limited emphasis on general task decomposition, cross-domain planning, and coordination across heterogeneous models. In our framework, we therefore use domain-specific models as specialized solvers for generated sub-problems, while a general model handles task planning, tool selection, and intermediate verification.

Self-Verification. Self-verification aims to improve LLM reliability by allowing models to assess and refine their own predictions or reasoning. Since direct self-critique may not reliably correct errors (Stechly et al., 2025), recent studies strengthen verification through reinforcement learning and joint generation-verification training. S^2R (Ma et al., 2025) jointly learns self-verification and self-correction, while SV-LLM (Zhang et al., 2025a) unifies generation and verification within a reinforcement learning framework. CoVerRL (Pan et al., 2026) further improves both through generator-verifier co-evolution. Recent works also explore finer-grained verification of intermediate reasoning, including step-level verification (Pandit et al., 2026) and multimodal verification based on visual grounding (Cai et al., 2026). Our framework performs verification in a multi-domain agent setting, where a general model checks the consistency between domain-specific tool evidence and sub-problem answers before final reasoning.

3 METHOD

In this section, we introduce Sub-Planner, as illustrated in Fig. 1. Let q denote an input query from a specialized domain, which may contain textual and visual inputs, and let $\mathcal{O}$ denote the candidate answer space. For multiple-choice tasks, $\mathcal{O}$ consists of a finite set of candidate options. For open-ended tasks, $\mathcal{O}$ represents the space of valid textual responses. The framework uses a domain-specific model $\mathcal{M}_{spec}$ for answering and a general model $\mathcal{M}_{gen}$ for planning and verification. Given a query, the framework first performs a preliminary assessment that allows easy queries to exit early. For queries requiring additional analysis, Sub-Planner decomposes it into an ordered sequence of sub-problems $S = (s_1, s_2, \ldots, s_K)$ and uses a set of tools $\mathcal{T} = \{t_1, t_2, \ldots, t_M\}$ to collect external evidence v_i for each sub-problem s_i. The framework consists of five stages: preliminary assessment, sub-problem planning, tool invocation, sub-problem solving and verification, and final verification.

3.1 STAGE 1: PRELIMINARY ASSESSMENT

To balance inference efficiency and answer quality for domain-specific queries q, the framework first performs a preliminary assessment with an early-exit mechanism. Specifically, the domain-specific model $\mathcal{M}_{spec}$ is initially prompted to solve the query directly. In this stage, the model $\mathcal{M}_{spec}$ generates both an initial answer a_{init} and its corresponding reasoning rationale r_{init}, formulated as:

$$(a_{init}, r_{init}) = \mathcal{M}_{spec}(q, \mathcal{O}) \tag{1}$$

The generated response is then evaluated by the general model $\mathcal{M}_{gen}$ to determine whether further reasoning is required. Given the query, answer, and rationale, $\mathcal{M}_{gen}$ assesses the logical consistency and sufficiency of the reasoning process. We define a binary verification function $\mathcal{V}$ based on the

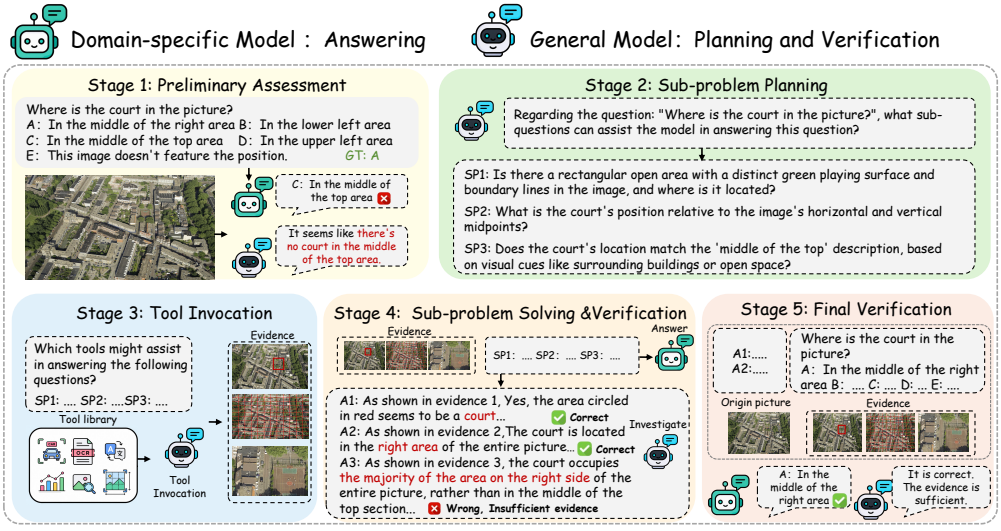


Figure 1: Overview of the proposed Sub-Planner framework. The framework consists of five stages: (a) **Preliminary Assessment**, where the domain-specific model directly answers the query and the general model determines whether further reasoning is required; (b) **Sub-problem Planning**, where the general model decomposes complex queries into multiple relatively simpler sub-problems; (c) **Tool Invocation**, where the general model selects appropriate tools to collect supporting evidence; (d) **Sub-problem Solving and Verification**, where the domain-specific model solves each sub-problem and the general model verifies the intermediate results; (e) **Final Verification**, where the framework integrates validated evidence and determines the final prediction.

general model $\mathcal{M}_{gen}$, which produces a binary verification indicator c:

$$c = \mathcal{V}_{init}^{\mathcal{M}_{gen}}(q, \mathcal{O}, a_{init}, r_{init}) \in \{0, 1\} \tag{2}$$

where $c = 1$ indicates that the initial response passes the check and can be returned without further reasoning, while $c = 0$ indicates that additional planning and tool-assisted analysis are required. When $c = 1$, the framework directly outputs a_{init} as the final answer without invoking subsequent sub-problem planning and tool usage stages. Otherwise, the query is forwarded to the following stages for further analysis.

3.2 Stage 2: Sub-problem Planning

When the preliminary assessment indicates that the initial response is insufficient (i.e., $c = 0$), the framework proceeds to the sub-problem planning stage. The general model $\mathcal{M}_{gen}$ generates an ordered sequence of sub-problems $S = (s_1, s_2, \ldots, s_K)$ based on the original query q. To control the trade-off between reasoning cost and problem coverage, we set the number of sub-problems K within a limited range, using 2 to 5 sub-problems for tasks with different difficulty levels:

$$S = (s_1, s_2, \ldots, s_K) = \mathcal{M}_{gen}(q), \quad \text{s.t.} \quad 2 \leq K \leq 5 \tag{3}$$

The motivation for this planning stage is to address the limitations of directly applying domain-specific models to complex domain-specific queries. In some scenarios, a domain-specific model may focus on dominant patterns while overlooking other relevant aspects of the input. For example, in industrial anomaly detection, the model may primarily rely on global visual characteristics and fail to sufficiently examine local regions associated with potential defects. By decomposing the original query into targeted sub-problems, the general model enables a more systematic analysis process and provides additional guidance for subsequent reasoning stages.

3.3 Stage 3: Tool Invocation

Some sub-problems s_i require additional information from external tools, as the model's internal knowledge may not provide sufficient evidence for reliable reasoning. However, domain-specific

models $\mathcal{M}_{spec}$ are primarily optimized for domain-specific tasks and may have limited ability in tool selection and complex instruction execution.

To address this issue, the general model $\mathcal{M}_{gen}$ is responsible for tool planning. Given the available tool set $\mathcal{T}$, $\mathcal{M}_{gen}$ analyzes the requirements of each sub-problem s_i, selects appropriate tools, and invokes them to obtain the corresponding evidence v_i:

$$v_i = \mathcal{T}_{sel}(s_i), \quad \mathcal{T}_{sel} = \mathcal{M}_{gen}(s_i, \mathcal{T}) \tag{4}$$

where $\mathcal{T}_{sel}$ denotes the selected tool set for solving sub-problem s_i. By assigning tool selection and execution to the general model while preserving domain-specific reasoning in the domain-specific model, the framework integrates external evidence into the subsequent reasoning process. If no external tool is required for a particular sub-problem, the selected tool set is empty, and the corresponding evidence acquisition stage for s_i is skipped.

3.4 Stage 4: Sub-problem Solving and Verification

Given the original visual input I and optional tool-derived evidence v_i, the domain-specific model generates a structured response for each sub-problem s_i:

$$(a_i, r_i, L_i) = \mathcal{M}_{spec}(s_i, I, v_i), \quad i = 1, \ldots, K \tag{5}$$

where a_i is the predicted answer, r_i is the corresponding rationale, and L_i contains the identifiers of the evidence items from v_i used to support the prediction. When $v_i = \emptyset$, the model relies solely on the original visual input I. If the available information is insufficient, the model outputs $a_i = \text{UNK}$ instead of making an unsupported prediction.

The generated sub-answer is then evaluated by the general model $\mathcal{M}_{gen}$ in terms of logical consistency, evidence grounding, and problem-level consistency:

$$(\ell_i, e_i, p_i, \eta_i, u_i) = \mathcal{V}_{sub}^{\mathcal{M}_{gen}}(q, S, s_i, I, v_i, a_i, r_i, L_i) \tag{6}$$

where $\ell_i \in \{0, 1\}$ indicates whether the rationale logically supports the answer, $e_i \in \{0, 1\}$ measures whether the answer is grounded in the visual input and available evidence, and $p_i \in \{0, 1\}$ evaluates its consistency with the original problem and sub-problem. $\eta_i \in \{0, 1, 2\}$ denotes the verification confidence level, and u_i provides a textual explanation.

The final acceptance decision is defined as:

$$f_i = \mathbb{I}(\ell_i = 1) \cdot \mathbb{I}(e_i = 1) \cdot \mathbb{I}(p_i = 1) \cdot \mathbb{I}(\eta_i \geq \tau_{sub}) \cdot \mathbb{I}(a_i \neq \text{UNK}) \tag{7}$$

where $\tau_{sub} = 1$. Accepted sub-answers are collected as:

$$A_{valid}^{(i)} = A_{valid}^{(i-1)} \cup \{(s_i, v_i, a_i, r_i, L_i) \mid f_i = 1\}, \quad A_{valid}^{(0)} = \emptyset \tag{8}$$

Each sub-problem is verified independently, and previously accepted sub-answers are not used as inputs to the verification of subsequent sub-problems. The iterative notation in $A_{valid}^{(i)}$ only represents the collection of accepted results. If $f_i = 0$, the corresponding result is discarded. After all sub-problems are processed, the final validated set is obtained as $A_{valid} = A_{valid}^{(K)}$.

3.5 Stage 5: Final Verification

After obtaining the validated sub-problem records A_{valid}, the domain-specific model revisits the original query and generates a refined candidate answer based on the validated evidence. It generates a candidate answer a_{prop}, the corresponding rationale r_{prop}, and the identifiers L_{prop} of the validated sub-problems and evidence involved:

$$(a_{prop}, r_{prop}, L_{prop}) = \mathcal{M}_{spec}(q, \mathcal{O}, a_{init}, r_{init}, A_{valid}) \tag{9}$$

If the proposed answer differs from the initial prediction a_{init}, the general model further evaluates whether the revision is sufficiently supported. Specifically, the verification considers three aspects: logical consistency of the proposed reasoning, ℓ_{final}, adequacy of the supporting evidence, e_{final}, and whether the validated evidence provides sufficient justification for changing the initial prediction to the proposed answer, d_{final}:

$$(\ell_{final}, e_{final}, d_{final}, \eta_{final}, u_{final}) = \mathcal{V}_{final}^{\mathcal{M}_{gen}}(q, \mathcal{O}, a_{init}, r_{init}, a_{prop}, r_{prop}, L_{prop}, A_{valid}) \tag{10}$$

where $\ell_{final}, e_{final}, d_{final} \in \{0, 1\}$ indicate whether the proposed reasoning is logically consistent, sufficiently grounded in the provided evidence, and justified for revising the initial prediction, respectively. $\eta_{final} \in \{0, 1, 2\}$ denotes the verifier-assessed confidence category, and u_{final} provides a textual explanation of the verification result. A deterministic acceptance criterion is then applied to determine whether the proposed revision should replace the initial answer:

$$\begin{aligned} g = \mathbb{I}(a_{prop} \in \mathcal{O}) \cdot \mathbb{I}(a_{prop} \neq a_{init}) \cdot \mathbb{I}(\ell_{final} = 1) \cdot \mathbb{I}(e_{final} = 1) \\ \cdot \mathbb{I}(d_{final} = 1) \cdot \mathbb{I}(\eta_{final} \geq \tau_{final}) \cdot \mathbb{I}(L_{prop} \neq \emptyset) \end{aligned} \tag{11}$$

where $\tau_{final} = 1$ denotes the predefined verification threshold. The final prediction is obtained as:

$$a_{final} = \begin{cases} a_{prop}, & \text{if } g = 1, \\ a_{init}, & \text{otherwise.} \end{cases} \tag{12}$$

This verification step allows the framework to incorporate evidence-supported revisions while maintaining the original prediction when the proposed change lacks sufficient support.

4 EXPERIMENTS

Experimental Setups. To demonstrate the generality of our framework, we evaluate it across 17 specialized domains, spanning Finance, Medical, Geometry, Pathology, Industrial Inspection, Geospatial, Remote Sensing, Robotics, GUI, Science, Education, Agriculture, Table, Law, Weather, Code, and Materials. We use Qwen3-VL-8B-Thinking (Bai et al., 2025a) as the generalist model across all domains, together with domain-specialized models for the corresponding settings. For each domain, we organize the evaluation data into different tasks and sample up to 300 examples from each task. If a task contains fewer than 300 examples, we use all available examples. All reported results are averaged over three runs with different random seeds. Detailed descriptions of the benchmarks (§A.1), domain-specialized models (§A.2), and the tool libraries (§A.3) used in our experiments are provided in the Appendix. For the GUI domain, we merge several subtasks and sample 400 examples for evaluation. All compared methods are evaluated under the same tool library configuration to ensure a fair comparison.

Performance across Diverse Domains. As shown in Table 1, we compare Sub-Planner with generalist and specialist base models, as well as representative training-based Gao et al. (2025b); Li et al. (2025c) and training-free agent methods Deng et al. (2026); Lei et al. (2025); Kumar et al. (2024) across 17 domains and 26 tasks. Since using domain-specialized models as agent backbones often limits cross-domain generalization, we use the generalist model as the backbone for all agent-based methods. Domain-specialized models are provided as external tools for all methods to ensure fair comparison. Sub-Planner achieves the highest average accuracy of 61.04% among all methods, improving over MARS Yue et al. (2026) by 6.03% and DART Sivakumaran et al. (2026) by 6.80% on average. Furthermore, Sub-Planner improves over specialist and generalist models by 5.41% and 12.26%, respectively. These results demonstrate that Sub-Planner can effectively leverage domain-specific tools across diverse multimodal tasks without additional training.

Ablation Study. To systematically investigate the contribution of each component in the proposed Sub-Planner framework, we conduct ablation experiments across five representative domains: Geometry, Science, Geospatial, Table, and Weather. As shown in Table 2, we remove sub-question verification, general model tool planning, and sub-question planning from the full framework. Removing sub-question verification results in consistent performance degradation, with a notable drop of 7.00% in the Table domain. Removing general model tool planning causes larger decreases, particularly in Geometry and Science, with drops of 9.66% and 7.34%, respectively. The largest degradation occurs when removing sub-question planning, leading to performance drops of 15.66% in Geometry, 12.34% in Science, and 7.67% in Geospatial. These results demonstrate that all three components make important contributions to the overall effectiveness of Sub-Planner.

Generalization to General MLLMs. To evaluate the generalization of Sub-Planner, we apply the framework to different general-purpose MLLMs. As shown in Table 3, integrating Sub-Planner improves the average performance of all evaluated models across multiple domains. Specifically, LLaVA-OneVision-7B and Qwen3-VL-8B-Thinking obtain average improvements of 9.16% and 14.09%, respectively. These results indicate that the effectiveness of Sub-Planner is not limited to a

Table 1: Comparison of different methods across diverse domains and tasks. Top three results among adapted methods are highlighted as **best**, second, and third. Generalist refers to the general model, while Specialist refers to the corresponding domain-specific model. Red arrows indicate the performance gains of our method over the maximum performance between the Generalist and Specialist models. –.– indicates that the method is not applicable to the corresponding domain.

Domain	Task	Base Models		Training			Training-Free				
		Generalist	*Specialist*	*MAT*	*SPORT*	*MARS*‡	*ToolScope*	*InfantAgent*	*DART*	*MMCT*	***Sub-Planner***
Finance	Fin. Reasoning	77.00	80.00	63.67	60.33	62.33	65.33	59.00	65.00	48.33	**80.33** ↑ 0.33
Medical	Basic Med. Sci.	59.33	64.00	44.00	52.67	**69.00**	61.33	63.67	67.67	47.00	64.33 ↑ 0.33
	Pharmacy	58.67	63.67	44.67	49.00	**71.00**	70.67	60.67	70.00	30.67	69.00 ↑ 5.33
Geometry	Geo. Reasoning	36.67	44.33	33.58	10.95	35.77	37.23	38.69	32.12	2.92	**54.33** ↑ 10.00
Pathology	Diagnosis	23.67	28.33	18.33	23.67	30.67	30.33	13.00	38.33	10.00	**40.00** ↑ 11.67
Industrial Inspection	Defect Cls.	56.33	64.67	29.00	53.00	55.00	48.33	**71.00**	69.67	53.00	65.33 ↑ 0.66
	Defect Loc.	66.00	69.00	42.33	40.33	68.33	52.67	51.33	65.00	31.67	**77.67** ↑ 8.67
	Defect Desc.	71.33	79.33	48.00	56.67	72.00	64.67	72.33	79.33	62.00	**81.67** ↑ 2.34
	Defect Analysis	80.67	89.33	61.33	63.00	85.33	55.67	78.67	78.67	74.67	**91.00** ↑ 1.67
Geospatial	Spa. Reasoning	32.67	37.00	29.00	29.33	37.33	35.33	28.67	35.67	14.33	**42.67** ↑ 5.67
Remote Sensing	Auto. Driving	34.67	40.00	22.67	27.33	**46.67**	20.33	24.33	24.33	14.67	44.33 ↑ 4.33
	RS Underst.	80.33	84.67	41.33	26.00	65.00	25.67	27.00	36.67	12.67	**87.00** ↑ 2.33
	Monitoring	31.00	38.33	26.33	27.33	**49.67**	31.33	32.67	31.67	19.67	46.00 ↑ 7.67
Robotics	Perception	39.00	42.00	28.33	33.33	39.33	36.67	**45.00**	44.33	24.33	42.33 ↑ 0.33
	Affordance	52.00	55.00	33.00	29.67	**56.67**	41.00	55.00	46.33	21.33	56.33 ↑ 1.33
	Error Analysis	18.68	23.73	19.59	26.01	**30.74**	15.20	27.70	27.36	27.36	25.76 ↑ 2.03
GUI	Goal-Task	36.25	40.00	44.00	39.25	45.50	53.25	55.75	**68.00**	43.00	60.00 ↑ 20.00
	Interaction	56.00	70.50	44.75	46.50	55.00	64.00	45.25	68.25	53.75	**77.25** ↑ 6.75
Science	Sci. Reasoning	82.33	92.33	76.00	68.00	89.00	71.00	82.33	89.00	67.33	**99.67** ↑ 7.34
Agriculture	Agri. Analysis	47.00	51.33	32.33	37.00	45.67	46.67	42.00	46.33	28.00	**54.67** ↑ 3.34
Table	Table Underst.	27.67	32.00	22.67	16.33	28.00	32.00	27.33	34.00	24.67	**37.67** ↑ 5.67
Law	Legal Reasoning	44.00	50.67	25.00	28.33	–.–	–.–	51.33	–.–	54.44	**62.67** ↑ 12.00
Weather	Weather Anal.	31.33	41.67	25.67	29.00	36.67	33.67	30.00	38.33	13.67	**43.00** ↑ 1.33
Code	Code Reasoning	29.26	32.10	33.28	34.13	–.–	–.–	43.28	–.–	–.–	**43.72** ↑ 11.62
Education	Edu. Reasoning	52.67	86.00	71.33	68.00	90.00	67.00	80.33	**90.67**	69.00	90.00 ↑ 4.00
Materials	Mat. Reasoning	43.67	46.33	34.33	46.33	**55.67**	46.67	50.00	55.00	32.67	50.33 ↑ 4.00
Average (%)		48.78	55.63	38.25	39.29	55.01	46.08	48.32	54.24	35.25	**61.04** ↑ 5.41

specific backbone. The consistent gains across different models suggest that sub-problem planning, tool invocation, and verification can provide complementary support for multimodal reasoning.

Effect of Sub-question Planning. To illustrate the effect of sub-question planning, we present several examples showing how it guides the reasoning of domain-specific models. As shown in Fig. 3, the models fail when directly answering the original queries. After decomposition into targeted sub-questions, they focus on task-relevant evidence, such as morphological characteristics in Pathology,

Table 2: Ablation study of our sub-planner framework across different domains.

Method	Geometry	Science	Geospatial	Table	Weather
Sub-Planner	**54.33%**	**99.67%**	**42.67%**	**37.67%**	**43.00%**
↪ w/o Sub-question Verification	54.00%$^{\downarrow 0.33}$	98.00%$^{\downarrow 1.67}$	40.00%$^{\downarrow 2.67}$	30.67%$^{\downarrow 7.00}$	41.67%$^{\downarrow 1.33}$
↪ w/o General Model Tool Planning	44.67%$^{\downarrow 9.66}$	92.33%$^{\downarrow 7.34}$	37.67%$^{\downarrow 5.00}$	30.00%$^{\downarrow 7.67}$	40.67%$^{\downarrow 2.33}$
↪ w/o Sub-question Planner	38.67%$^{\downarrow 15.66}$	87.33%$^{\downarrow 12.34}$	35.00%$^{\downarrow 7.67}$	27.67%$^{\downarrow 10.00}$	40.33%$^{\downarrow 2.67}$

Table 3: Performance comparison between base MLLMs and their agent-enhanced variants across different domains. The last column reports the average performance over four evaluated domains, with the absolute improvement brought by the agent framework indicated as a superscript.

Model	Geometry	Table	Geospatial	Law	Average
LLaVA-OneVision-7B	33.67%	11.67%	33.00%	42.00%	30.09%
+ Sub-Planner	**46.00%**	**13.67%**	**39.00%**	**58.33%**	**39.25%**$^{\uparrow 9.16}$
InternVL3-8B	21.00%	17.33%	33.00%	38.67%	27.50%
+ Sub-Planner	**46.00%**	**18.33%**	**35.33%**	**56.33%**	**39.00%**$^{\uparrow 11.50}$
Llama-3.2-Vision-11B	15.67%	8.33%	29.67%	43.67%	24.34%
+ Sub-Planner	**37.33%**	**11.67%**	**32.00%**	**61.00%**	**35.50%**$^{\uparrow 11.16}$
Qwen3-VL-8B-Thinking	36.67%	27.67%	32.67%	44.00%	35.25%
+ Sub-Planner	**54.33%**	**37.67%**	**42.67%**	**62.67%**	**49.34%**$^{\uparrow 14.09}$

and pressure patterns in Weather. By resolving these intermediate questions, the models correct their initial predictions and reach the ground-truth answers. These cases show that sub-question planning enables more structured analysis of complex multimodal tasks.

Inference Efficiency. To evaluate the inference efficiency of Sub-Planner, we compare its average inference time with different training-based and training-free methods. As shown in Fig. 2, Sub-Planner achieves an average performance of 61.04% with an inference time of 15.2 seconds per case. Its inference latency is comparable to ToolScope and MAT, while being substantially lower than DART, MMCT, SPORT, and InfantAgent. Meanwhile, Sub-Planner consistently achieves higher average performance than these competing methods. These results indicate that the performance gains of Sub-Planner are achieved without introducing excessive inference overhead. By decomposing complex tasks into sub-problems and selectively invoking specialized tools, Sub-Planner maintains efficient inference while improving cross-domain multimodal reasoning. Overall, the proposed framework provides a favorable balance between task performance and inference efficiency.

Effect of Tool Invocation Based on Sub-questions. To investigate whether using sub-questions can simplify the tool invocation process, we compare the tool selection accuracy based on the original questions and the generated sub-questions. The tool invocation results are first annotated by GPT-5.6, and the resulting annotations are manually verified for correctness. As shown in Fig. 4, selecting tools based on sub-questions achieves higher accuracy across all evaluated domains. Specifically, the accuracy improves from 36.0% to 66.0% in Geospatial, from 22.0% to 28.2% in Pathology, and from 40.0% to 58.1% in Science. These results indicate that sub-question provides more explicit task information for tool selection and improves tool invocation accuracy on cross-domain multimodal tasks.

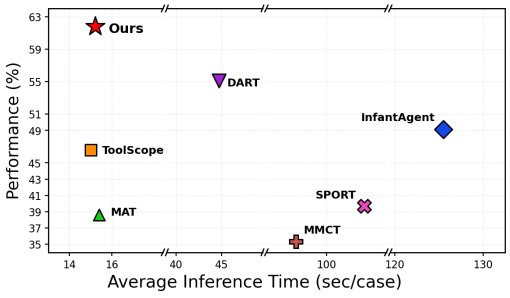


Figure 2: Performance-efficiency comparison across different methods.

Effect of Sub-problem Strategies on Model Accuracy. To evaluate the effect of sub-problem decomposition, we compare three settings: no sub-problems, fixed sub-problems, and dynamic sub-problems. As shown in Fig. 5, both sub-problem strategies improve accuracy over the baseline across all evaluated domains, with dynamic sub-problems achieving the best overall performance.

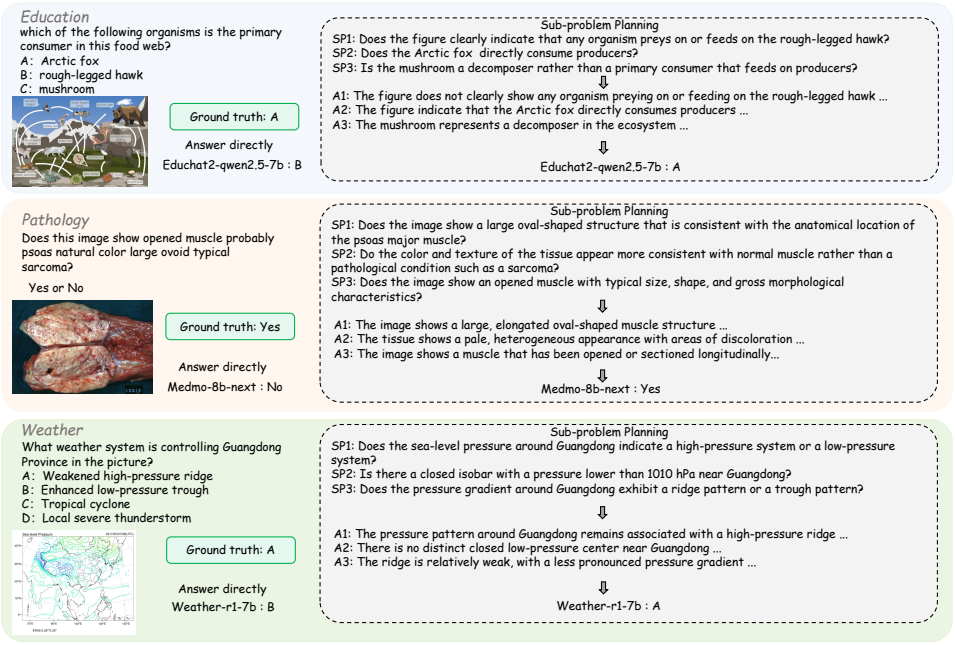


Figure 3: Compared with directly answering the original queries, sub-question planning guides domain-specific models to analyze intermediate evidence and correct their predictions.

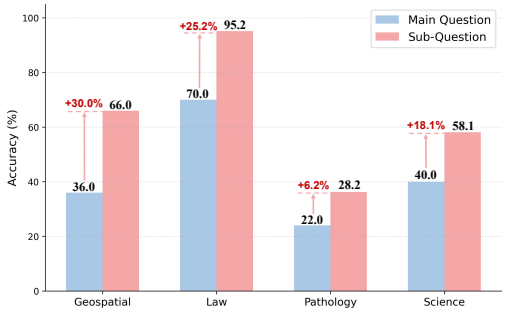


Figure 4: Effect of Tool Invocation Based on Sub-questions. The results demonstrate that sub-questions provide more explicit task information, consistently improving tool selection accuracy across all evaluated domains compared to using the original questions.

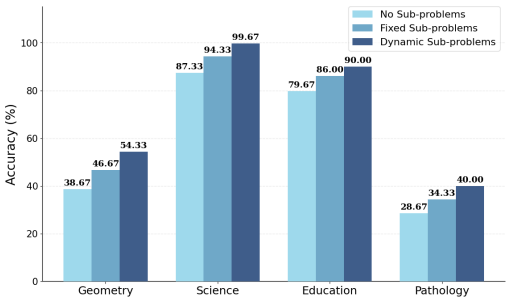


Figure 5: Accuracy comparison of different sub-problem generation strategies across various domains. This figure illustrates the model's performance in four distinct subjects when employing No Sub-problems, Fixed Sub-problems, and Dynamic Sub-problems strategies.

Specifically, accuracy increases from 38.67% to 54.33% in Geometry, from 87.33% to 99.67% in Science, and from 28.67% to 40.00% in Pathology. These results indicate that decomposing complex tasks into sub-problems can improve reasoning accuracy, while dynamic decomposition provides greater flexibility than a fixed strategy across different question types.

5 Conclusion

In this paper, we presented a general agent framework for domain-specific multimodal reasoning that integrates the complementary strengths of general-purpose and domain-specific models. The general model plans reasoning steps, generates task-adaptive sub-problems, selects specialized tools, and verifies intermediate evidence, while domain-specific models solve individual sub-problems. By filtering unreliable evidence before final reasoning, the framework supports diverse domains without manually designed task-specific workflows, highlighting its potential for more reliable and transferable multimodal reasoning. Future work may explore more efficient tool selection, stronger verification, and broader adaptation to open-ended domains and tasks.

AI USE STATEMENT

In this work, we used generative AI tools for manuscript writing assistance, including language refinement, sentence rephrasing, grammar checking, and improving readability. We have not used generative AI tools for research ideation, methodology development, experimental design, data analysis, result interpretation, or any other scientific contributions. Additionally, generative AI tools were used solely for language polishing and editing. We have reviewed all AI-assisted content to ensure its correctness, clarity, and consistency. All research ideas, experiments, analyses, and conclusions were developed and conducted by the authors. We take responsibility for the final content of this work, including text refined with the aid of generative AI tools.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b.

Abderrahmene Boudiaf, Irfan Hussain, and Sajid Javed. Agrichat: A multimodal large language model for agriculture image understanding. *arXiv preprint arXiv:2603.16934*, 2026.

Markus J Buehler. Cephalo: multi-modal vision-language models for bio-inspired materials analysis and design. *Advanced Functional Materials*, 2024.

Wei Cai, Jian Zhao, Yuchen Yuan, Tianle Zhang, Ming Zhu, Haichuan Tang, and Xuelong Li. Visual attention reasoning via hierarchical search and self-verification. In *ACL*, 2026.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, 2024.

Dawei Dai, Yuanhui Zhang, Long Xu, Qianlan Yang, Xiaojing Shen, Shuyin Xia, and Guoyin Wang. Pa-llava: A large language-vision assistant for human pathology image understanding. In *2024 IEEE international conference on bioinformatics and biomedicine (BIBM)*, pp. 3138–3143. IEEE, 2024.

Yuhao Dan, Zhikai Lei, Yiyang Gu, Yong Li, Jianghao Yin, Jiaju Lin, Linhao Ye, Zhiyan Tie, Yougen Zhou, Yilei Wang, et al. Educhat: A large-scale language model-based chatbot system for intelligent education. *arXiv preprint arXiv:2308.02773*, 2023.

Muhammad Danish, Muhammad Akhtar Munir, Syed Roshaan Ali Shah, Kartik Kuckreja, Fahad Shahbaz Khan, Paolo Fraccaro, Alexandre Lacoste, and Salman Khan. Geobench-vlm: Benchmarking vision-language models for geospatial tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7132–7142, 2025.

Mengjie Deng, Guanting Dong, and Zhicheng Dou. Toolscope: An agentic framework for vision-guided and long-horizon tool use. In *ACL*, 2026.

Ankan Deria, Komal Kumar, Adinath Madhavrao Dukre, Eran Segal, Salman Khan, and Imran Razzak. Medmo: Grounding and understanding multimodal large language model for medical images. *arXiv preprint arXiv:2602.06965*, 2026.

Adibvafa Fallahpour, Jun Ma, Alif Munim, Hongwei Lyu, and Bo Wang. Medrax: Medical reasoning agent for chest x-ray. *arXiv preprint arXiv:2502.02673*, 2025.

Yue Fan, Xiaojian Ma, Rujie Wu, Yuntao Du, Jiaqi Li, Zhi Gao, and Qing Li. Videoagent: A memory-augmented multimodal agent for video understanding. In *ECCV*, 2024.

Z Fei, X Shen, D Zhu, F Zhou, Z Han, S Zhang, K Chen, Z Shen, and J Lawbench Ge. Benchmarking legal knowledge of large language models. arxiv 2023. *arXiv preprint arXiv:2309.16289*, 2023.

Ziliang Gan, Dong Zhang, Haohan Li, Yang Wu, Xueyuan Lin, Ji Liu, Haipang Wu, Chaoyou Fu, Zenglin Xu, Rongjunchen Zhang, et al. Mme-finance: A multimodal finance benchmark for expert-level understanding and reasoning. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 12867–12874, 2025.

Jiahui Gao, Renjie Pi, Jipeng Zhang, Jiacheng Ye, Wanjun Zhong, Yufei Wang, Lanqing Hong, Jianhua Han, Hang Xu, Zhenguo Li, et al. G-llava: Solving geometric problem with multi-modal large language model. In *ICLR*, 2025a.

Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaojian Ma, Tao Yuan, Yue Fan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and Qing Li. Multi-modal agent tuning: Building a vlm-driven agent for efficient tool usage. In *ICLR*, 2025b.

Zhaopeng Gu, Bingke Zhu, Guibo Zhu, Yingying Chen, Ming Tang, and Jinqiao Wang. Anomalygpt: Detecting industrial anomalies using large vision-language models. In *AAAI*, 2024.

Xuehai He, Yichen Zhang, Luntian Mou, Eric Xing, and Pengtao Xie. Pathvqa: 30000+ questions for medical visual question answering. *arXiv preprint arXiv:2003.10286*, 2020.

Jimin Huang, Mengxi Xiao, Dong Li, Zihao Jiang, Yuzhe Yang, Yifei Zhang, Lingfei Qian, Yan Wang, Xueqing Peng, Yang Ren, et al. Open-finllms: Open multimodal large language models for financial applications. *arXiv preprint arXiv:2408.11878*, 2024.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Xi Jiang, Jian Li, Hanqiu Deng, Yong Liu, Bin-Bin Gao, Yifeng Zhou, Jialin Li, Chengjie Wang, and Feng Zheng. Mmad: A comprehensive benchmark for multimodal large language models in industrial anomaly detection. In *International conference on learning representations*, volume 2025, pp. 87273–87295, 2025.

Hyunju Kang, Woohyun Lee, Jaewon Kim, and Hogun Park. Judo: A juxtaposed domain-oriented multimodal reasoner for industrial anomaly qa. In *ICLR*, 2026.

Kartik Kuckreja, Muhammad Sohail Danish, Muzammal Naseer, Abhijit Das, Salman Khan, and Fahad Shahbaz Khan. Geochat: Grounded large vision-language model for remote sensing. In *CVPR*, 2024.

Somnath Kumar, Yash Gadhia, Tanuja Ganu, and Akshay Nambi. Mmctagent: Multi-modal critical thinking agent framework for complex visual reasoning. *arXiv preprint arXiv:2405.18358*, 2024.

Zhengzhao Lai, Youbin Zheng, Zhenyang Cai, Haonan Lyu, Jinpu Yang, Hongqing Liang, Yan Hu, and Benyou Wang. Can multimodal llms see materials clearly? a multimodal benchmark on materials characterization. *arXiv preprint arXiv:2509.09307*, 2025.

Bin Lei, Weitai Kang, Zijian Zhang, Winson Chen, Xi Xie, Shan Zuo, Mimi Xie, Ali Payani, Mingyi Hong, Yan Yan, et al. Infantagent-next: A multimodal generalist agent for automated computer interaction. *NeurIPS*, 2025.

Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *arXiv preprint arXiv:2407.07895*, 2024.

Hongxin Li, Jingran Su, Jingfan Chen, Zheng Ju, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. Uipro: Unleashing superior interaction capability for gui agents. In *ICCV*, 2025a.

Junxian Li, Di Zhang, Xunzhi Wang, Zeying Hao, Jingdi Lei, Qian Tan, Cai Zhou, Wei Liu, Yaotian Yang, Xinrui Xiong, et al. Chemvlm: Exploring the power of multimodal large language models in chemistry area. In *AAAI*, 2025b.

Pengxiang Li, Zhi Gao, Bofei Zhang, Yapeng Mi, Xiaojian Shawn Ma, Chenrui Shi, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, et al. Iterative tool usage exploration for multimodal agents via step-wise preference tuning. *NeurIPS*, 2025c.

Yanhui Li, Yunkang Cao, Chengliang Liu, Yuan Xiong, Xinghui Dong, and Chao Huang. Iad-r1: Reinforcing consistent reasoning in industrial anomaly detection. In *AAAI*, 2026.

Yunlong Lin, Zixu Lin, Haoyu Chen, Panwang Pan, Chenxin Li, Sixiang Chen, Kairun Wen, Yeying Jin, Wenbo Li, and Xinghao Ding. Jarvisir: Elevating autonomous driving perception with intelligent image restoration. In *CVPR*, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *NeurIPS*, 2023.

Jinqiang Long, Yanqi Dai, Guoxing Yang, Hongpeng Lin, Nanyi Fei, Yizhao Gao, and Zhiwu Lu. Awaker2. 5-vl: stably scaling mllms with parameter-efficient mixture of experts. *arXiv preprint arXiv:2411.10669*, 2024.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *NeurIPS*, 2022.

Yulin Luo, Chun-Kai Fan, Menghang Dong, Jiayu Shi, Xiangju Mi, Mengdi Zhao, Bo-Wen Zhang, Cheng Chi, Jiaming Liu, Gaole Dai, et al. Robobench: A comprehensive evaluation benchmark for multimodal large language models as embodied brain. *arXiv preprint arXiv:2510.17801*, 2025.

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. S2r: Teaching llms to self-verify and self-correct via reinforcement learning. In *ACL*, 2025.

Umair Nawaz, Awais Muhammad, Hanan Gani, Muzammal Naseer, Fahad Shahbaz Khan, Salman Khan, and Rao Anwer. Agriclip: Adapting clip for agriculture and livestock via domain-specialized cross-model alignment. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 9630–9639, 2025.

Xuying Ning, Dongqi Fu, Tianxin Wei, Mengting Ai, Jiaru Zou, Ting-Wei Li, Hanghang Tong, Yada Zhu, Hendrik Hamann, and Jingrui He. Mc-search: Evaluating and enhancing multimodal agentic search with structured long reasoning chains. In *ICLR*, 2026.

Jiazhen Pan, Che Liu, Junde Wu, Fenglin Liu, Jiayuan Zhu, Hongwei Bran Li, Chen Chen, Cheng Ouyang, and Daniel Rueckert. Medvlm-r1: Incentivizing medical reasoning capability of vision-language models (vlms) via reinforcement learning. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2025.

Teng Pan, Yuchen Yan, Zixuan Wang, Ruiqing Zhang, Guiyang Hou, Wenqi Zhang, Weiming Lu, Jun Xiao, and Yongliang Shen. Coverrl: Breaking the consensus trap in label-free reasoning via generator-verifier co-evolution. In *ACL*, 2026.

Shrey Pandit, Austin Xu, Xuan-Phi Nguyen, Yifei Ming, Caiming Xiong, and Shafiq Joty. Hard2verify: A step-level verification benchmark for open-ended frontier math. In *ACL*, 2026.

Monoshi Kumar Roy, Simin Chen, Benjamin Steenhoek, Jinjun Peng, Gail Kaiser, Baishakhi Ray, and Wei Le. Codesense: A real-world benchmark and dataset for code semantic reasoning. In *ICLR*, 2026.

Run Shao, Ziyu Li, Zhaoyang Zhang, Linrui Xu, Xinran He, Hongyuan Yuan, Bolei He, Yongxing Dai, Yiming Yan, Yijun Chen, et al. Asking like socrates: Socrates helps vlms understand remote sensing images. In *CVPR*, 2026.

Chenrui Shi, Zedong Yu, Zhi Gao, Ruining Feng, Enqi Liu, Yuwei Wu, Yunde Jia, Liuyu Xiang, Zhaofeng He, and Qing Li. Gui knowledge bench: Revealing the knowledge gap of vlms in gui tasks. *arXiv preprint arXiv:2510.26098*, 2025.

Nithin Sivakumaran, Justin Chen, David Wan, Yue Zhang, Jaehong Yoon, Elias Stengel-Eskin, and Mohit Bansal. Dart: Leveraging multi-agent disagreement for tool recruitment in multimodal reasoning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics*, 2026.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. In *ICLR*, 2025.

Andrew Szot, Bogdan Mazoure, Omar Attia, Aleksei Timofeev, Harsh Agrawal, Devon Hjelm, Zhe Gan, Zsolt Kira, and Alexander Toshev. From multimodal llms to generalist embodied agents: Methods and lessons. In *CVPR*, 2025.

Huajie Tan, Enshen Zhou, Zhiyu Li, Yijie Xu, Yuheng Ji, Xiansheng Chen, Cheng Chi, Pengwei Wang, Huizhu Jia, Yulong Ao, et al. Robobrain 2.5: Depth in sight, time in mind. *arXiv preprint arXiv:2601.14352*, 2026.

Haotian Wang, Yi Guan, Fanshu Meng, Chao Zhao, Lian Yan, Yang Yang, and Jingchi Jiang. Agri-cm3: A chinese massive multi-modal, multi-level benchmark for agricultural understanding and reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11729–11754, 2025a.

Shihao Wang, Zhiding Yu, Xiaohui Jiang, Shiyi Lan, Min Shi, Nadine Chang, Jan Kautz, Ying Li, and Jose M Alvarez. Omnidrive: A holistic vision-language dataset for autonomous driving with counterfactual reasoning. In *CVPR*, 2025b.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. In *ECCV*, 2024.

Kaiyu Wu, Pucheng Han, Hualong Zhang, Naigeng Wu, and Keze Wang. Weather-r1: Logically consistent reinforcement fine-tuning for multimodal reasoning in meteorology. In *ICASSP*, 2026.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, et al. Tablebench: A comprehensive and complex benchmark for table question answering. In *AAAI*, 2025.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, et al. Mobile-agent-v3. 5: Multi-platform fundamental gui agents. *arXiv preprint arXiv:2602.16855*, 2026a.

Peiran Xu, Jiaqi Zheng, and Yadong Mu. Roboagent: Chaining basic capabilities for embodied task planning. In *CVPR*, 2026b.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction. In *ICML*, 2025.

Shengbin Yue, Wei Chen, Siyuan Wang, Bingxuan Li, Chenchen Shen, Shujun Liu, Yuxuan Zhou, Yao Xiao, Song Yun, Xuanjing Huang, et al. Disc-lawllm: Fine-tuning large language models for intelligent legal services. *arXiv preprint arXiv:2309.11325*, 2023.

Tan Yue, Qiong Wu, and Dongyan Zhao. Mars: Multimodal adaptive reasoning model for avoiding overthinking. In *AAAI*, 2026.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *CVPR*, 2024.

Yu-Wei Zhan, Xin Wang, Pengzhe Mao, Tongtong Feng, Ren Wang, and Wenwu Zhu. Modularagent: A task-aware modular framework for joint optimization of multimodal large language models and world models. In *CVPR*, 2026.

Fuxiang Zhang, Jiacheng Xu, Chaojie Wang, Ce Cui, Yang Liu, and Bo An. Incentivizing llms to self-verify their answers. *NeurIPS*, 2025a.

YiFan Zhang, Huanyu Zhang, Haochen Tian, Chaoyou Fu, Shuangqing Zhang, Junfei Wu, Feng Li, Kun Wang, Qingsong Wen, Zhang Zhang, et al. Mme-realworld: Could your multimodal llm challenge high-resolution real-world scenarios that are difficult for humans? In *International Conference on Learning Representations*, volume 2025, pp. 89655–89701, 2025b.

Mingyu Zheng, Xinwei Feng, Qingyi Si, Qiaoqiao She, Zheng Lin, Wenbin Jiang, and Weiping Wang. Multimodal table understanding. In *ACL*, 2024.

Qihang Zhou, Jiangtao Yan, Shibo He, Wenchao Meng, and Jiming Chen. Pointad: Comprehending 3d anomalies from points and pixels for zero-shot 3d anomaly detection. *NeurIPS*, 2024.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

A Appendix

A.1 Benchmark Details

MME-Finance (Gan et al., 2025) is a multimodal benchmark designed to evaluate model capabilities in financial scenarios. It contains finance-related visual and textual information and requires models to understand domain-specific concepts and perform financial reasoning. We use MME-Finance to evaluate the Finance domain.

MMMU (Yue et al., 2024) is a multidisciplinary multimodal benchmark containing questions that require both domain knowledge and visual reasoning. For the Medical domain, we use medical-related problems from MMMU, which require models to interpret visual information and apply specialized medical knowledge.

Geo170K (Gao et al., 2025a) is a large-scale geometry dataset consisting of diagram-based geometry problems. Solving these problems requires models to understand geometric structures from visual inputs and combine them with mathematical reasoning. We use Geo170K to evaluate the Geometry domain.

PathVQA (He et al., 2020) is a visual question answering dataset constructed from pathology images. Each example contains a pathology image together with a natural-language question, requiring models to combine visual interpretation with domain-specific medical knowledge. We use PathVQA to evaluate the Pathology domain.

MMAD (Jiang et al., 2025) is a multimodal benchmark for industrial anomaly understanding. It contains industrial inspection images involving different types of products and anomalies, requiring models to identify and reason about abnormal visual patterns. We use MMAD to evaluate the Industrial Inspection domain.

GEOBench-VLM (Danish et al., 2025) is a benchmark designed to evaluate vision-language models on geospatial understanding and reasoning tasks. The benchmark requires models to interpret geospatial visual information and answer domain-specific questions. We use GEOBench-VLM to evaluate the Geospatial domain.

MME-RealWorld (Zhang et al., 2025b) is a benchmark designed to evaluate multimodal models in complex real-world scenarios. We use its remote-sensing-related data to evaluate the Remote Sensing domain, where models need to understand large-scale visual scenes and reason about geographic information.

RoboBench (Luo et al., 2025) is a benchmark for evaluating multimodal reasoning in robotics scenarios. It requires models to understand visual observations and reason about objects, spatial relationships, and task-related information. We use RoboBench to evaluate the Robotics domain.

GUI Knowledge Bench (Shi et al., 2025) evaluates the understanding of graphical user interfaces. Its tasks require models to recognize interface elements, understand their functions, and reason about interactions within GUI environments. We use this benchmark to evaluate the GUI domain.

ScienceQA (Lu et al., 2022) is a multimodal question answering benchmark involving scientific and educational knowledge. Questions may contain textual context and visual information and require models to combine knowledge with multi-step reasoning. Based on our task-level partition, we use ScienceQA to construct separate evaluation settings for the Science and Education domains.

Agri-CM3 (Wang et al., 2025a) is a multimodal benchmark focusing on agricultural scenarios. It requires models to interpret agriculture-related visual information and answer questions using domain-specific knowledge. We use Agri-CM3 to evaluate the Agriculture domain.

TableBench (Wu et al., 2025) is a benchmark designed to evaluate table understanding and reasoning. Models are required to interpret structured tabular information, identify relevant entries, and perform reasoning based on relationships among different cells. We use TableBench to evaluate the Table domain.

LawBench (Fei et al., 2023) is a benchmark for evaluating legal knowledge and reasoning. It covers diverse law-related tasks that require models to understand legal concepts and apply them to specific questions or scenarios. We use LawBench to evaluate the Law domain.

WeatherQA (Wu et al., 2026) is a question answering benchmark focusing on weather-related understanding and reasoning. It requires models to interpret weather information and apply meteorological knowledge to answer domain-specific questions. We use WeatherQA to evaluate the Weather domain.

CodeSense (Roy et al., 2026) is a benchmark designed to evaluate code understanding and reasoning capabilities. Its tasks require models to understand program semantics and reason about the behavior or functionality of code. We use CodeSense to evaluate the Code domain.

MatCha (Lai et al., 2025) is a benchmark focusing on materials-related understanding and reasoning. It requires models to apply specialized materials-science knowledge to solve domain-specific problems. We use MatCha to evaluate the Materials domain.

A.2 Domain-specific Model Details

G-LLaVA-7B (Gao et al., 2025a) is a domain-specialized multimodal large language model designed for geometric reasoning tasks. It is trained to understand visual geometric structures and perform multi-step reasoning over diagram-based problems. We use G-LLaVA-7B as the specialist model for the Geometry domain.

ScienceQA-LLaVA-13B (Lu et al., 2022) is a science-oriented multimodal model adapted from LLaVA for scientific question answering. It combines visual understanding with scientific knowledge to solve complex science problems involving textual and visual information. We use ScienceQA-LLaVA-13B as the specialist model for the Science domain.

EduChat2-Qwen2.5-7B (Dan et al., 2023) is an educational-domain language model designed for educational question answering and reasoning. It incorporates instructional knowledge and domain-specific reasoning abilities for solving educational tasks. We use EduChat2-Qwen2.5-7B as the specialist model for the Education domain.

GeoChat-7B (Kuckreja et al., 2024) is a geospatial vision-language model specialized in understanding remote sensing images and geographical information. It is designed to perform spatial reasoning by combining visual perception with geographic knowledge. We use GeoChat-7B as the specialist model for the Geospatial domain.

Table-LLaVA-7B (Zheng et al., 2024) is a multimodal model specialized in table understanding and reasoning. It is trained to interpret structured table information from visual inputs and answer questions requiring reasoning over tabular data. We use Table-LLaVA-7B as the specialist model for the Table domain.

DISC-LawLLM-Int8 (Yue et al., 2023) is a legal-domain large language model designed for legal understanding and reasoning tasks. It incorporates legal knowledge and instruction tuning to handle domain-specific legal queries. We use DISC-LawLLM-Int8 as the specialist model for the Law domain.

MedMO-8B-Next (Deria et al., 2026) is a medical-domain multimodal model designed to integrate medical knowledge with visual understanding. It supports specialized reasoning over medical images and clinical-related questions. We use MedMO-8B-Next as the specialist model for the Medical and Pathology domains.

Weather-R1-7B (Wu et al., 2026) is a weather-domain reasoning model designed for meteorological understanding and analysis. It incorporates weather-related knowledge to interpret visual and textual information in meteorological scenarios. We use Weather-R1-7B as the specialist model for the Weather domain.

FinLLaVA (Huang et al., 2024) is a financial vision-language model specialized in financial understanding and reasoning. It is trained on finance-related multimodal data to handle financial analysis tasks involving both textual and visual information. We use FinLLaVA as the specialist model for the Finance domain.

AgriChat (Boudiaf et al., 2026) is an agricultural-domain multimodal model designed to understand agricultural images and related textual descriptions. It incorporates agriculture-specific knowledge for solving domain-related reasoning tasks. We use AgriChat as the specialist model for the Agriculture domain.

JUDO (Kang et al., 2026) is an industrial vision-language model designed for manufacturing inspection tasks. It focuses on understanding industrial images and performing defect-related reasoning, including classification, localization, and analysis. We use JUDO as the specialist model for the Industrial Inspection domain.

Awaker2.5-VL (Long et al., 2024) is a general-purpose vision-language model with strong capabilities in real-world visual understanding tasks. It provides multimodal reasoning ability for complex scenarios such as autonomous driving and remote sensing. We use Awaker2.5-VL as the specialist model for the Remote Sensing domain.

RoboBrain2.5-8B-NV (Tan et al., 2026) is a robotics-oriented multimodal model designed for embodied intelligence tasks. It focuses on robot perception, affordance reasoning, and environment understanding. We use RoboBrain2.5-8B-NV as the specialist model for the Robotics domain.

GUI-Owl-1.5-8B-Instruct (Xu et al., 2026a) is a multimodal model specialized in graphical user interface understanding and interaction reasoning. It is designed to interpret GUI layouts and perform task-oriented reasoning over interface elements. We use GUI-Owl-1.5-8B-Instruct as the specialist model for the GUI domain.

Qwen2.5-Coder-7B-Instruct (Hui et al., 2024) is a code-specialized language model designed for programming understanding and code reasoning. It provides strong capabilities in program analysis, execution reasoning, and code generation. We use Qwen2.5-Coder-7B-Instruct as the specialist model for the Code domain.

Cephalo-11B (Buehler, 2024) is a multimodal vision-language model developed for materials science and bio-inspired materials research. It is designed to understand scientific images together with materials-related textual information, supporting visual reasoning over material structures, microstructures, characterization figures, and bio-inspired material systems. We use Cephalo-11B as the specialist model for the Materials domain.

A.3 Cross-Domain Toolkit

We construct a unified toolkit that covers all evaluation domains and contains more than 60 tools for acquiring task-relevant evidence. We summarize the function, interface, and output of each tool below.

`crop_image` extracts and enlarges a specified region from an image for focused visual inspection. It takes an image path, a bounding box specified by normalized coordinates $[x_1, y_1, x_2, y_2]$ within $[0, 1]$, and an optional output directory. It returns the normalized bounding box actually used, the original crop size, the final output size, and the filename of the generated cropped image.

`tile_grid` divides an image into a regular grid to support global inspection and spatial localization. It takes an image path, the numbers of rows and columns, and an optional output directory. It returns the grid dimensions, the cell-indexing convention, and the generated grid image.

`measure_color` measures color statistics for an image or a specified image region. It takes an image path and an optional bounding box. It returns the mean RGB and HSV values, the mean color of the remaining image, the color difference between the selected region and the remainder, and a confidence estimate.

`compare_colors` compares the color properties of two regions within the same image. It takes an image path and two bounding boxes. It returns the color statistics of both regions, their color difference, relative lightness and saturation information, and a confidence estimate.

`intensity_summary` summarizes the grayscale intensity distribution of an image or image region. It takes an image path and an optional bounding box. It returns the mean and standard deviation of the intensity values, the 5th, 50th, and 95th percentiles, and a confidence estimate.

`edge_density` measures local edge and gradient statistics to characterize texture and structural variation. It takes an image path and an optional bounding box. It returns the edge density, mean gradient, adaptive threshold, and a confidence estimate.

`component_count` estimates the number of connected components within an image region. It takes an image path, an optional bounding box, and an optional output directory. It returns the estimated component count, the segmentation threshold, a confidence estimate, and an optional visualization file.

`ocr_image` extracts visible text from an image or a specified image region. It takes an image path and an optional bounding box. It returns the recognized text, line-level recognition results, information about the OCR model, and a confidence estimate.

`detect_objects` performs open-vocabulary object detection based on textual labels. It takes an image path, a list of object labels, and an optional output directory. It returns the detected objects, their localization results, the number of detections, the requested labels, information about the detection model, and an optional visualization file.

`segment_objects` segments an object or interface element specified by a textual target. It takes an image path, a target description, and an optional output directory. It returns the segmentation result, fallback bounding boxes when available, the mask area fraction, the number of polygons, model information, and an optional visualization file.

`compare_images` measures visual differences between two images, frames, or selected regions. It takes a primary image path, an optional second image path, optional bounding boxes for the two inputs, and an optional output directory. It returns the changed region, the fraction of changed pixels, the maximum difference, the mean absolute difference, and an optional visualization file.

`track_objects` summarizes visual changes across an ordered sequence of frames. It takes a list of ordered image paths and an optional output directory. It returns the number of processed frames, detected transitions, the tracking mode, and a confidence estimate.

`reference_compare` compares a query image with a set of reference images using conventional visual features. It takes a query image path, a list of reference image paths, and an optional output directory. It returns per-reference comparison results, aggregate color and edge differences, a reference deviation score, and an optional visualization file.

`semantic_reference_compare` compares a query image with reference images using semantic visual representations. It takes a query image path and a list of reference image paths. It returns semantic comparison scores between the query image and individual reference images.

`calculator` evaluates basic arithmetic expressions and supported mathematical functions. It takes an arithmetic expression and returns the normalized expression together with the computed value.

`python_repl` executes a restricted Python snippet for lightweight numerical computation. It takes Python code and returns the standard output, the computed result, and the resulting local variables.

`symbolic_math` performs symbolic simplification and equation-oriented computation using SymPy. It takes a mathematical expression together with optional equations and variable names. It returns the simplified symbolic expression and, when available, a numerical result.

`unit_convert` converts values between supported physical units. It takes a numerical value, a source unit, and a target unit. It returns the converted value together with the source unit, the target unit, and the detected physical dimension.

`text_evidence_extract` extracts lexical evidence from textual problem content. It takes input text together with an optional query and optional answer choices. It returns detected numbers, dates, negation terms, relevant sentences, and coverage statistics for the candidate options.

`document_parse` parses text-dense document images using OCR. It takes an image path and an optional bounding box. It returns the extracted text, line-level OCR results, OCR metadata, and information about the OCR model.

`table_extract` converts a table image into structured row candidates. It takes an image path and an optional bounding box. It returns the OCR result, extracted rows, and the number of detected rows.

`table_math` performs deterministic numerical operations over explicitly provided table values. It takes an operation and may additionally receive values, rows, a target column, a numerator, a denominator, or weights. It returns the executed operation, the numerical result, and the number of values used.

`structured_table_parse` parses a textual table representation into structured records. It takes table text and returns the detected columns, parsed records, and the number of rows.

`table_query` performs structured queries over textual table data. It takes table text, a query operation, a target column, and an optional filter condition. It returns the matched records or values, the number of matched rows, and the query result.

`geometry_diagram_parse` extracts pixel-level geometric structures from diagram images. It takes an image path and an optional bounding box. It returns detected line segments and circles, their counts, the image size, and the fraction of edge pixels.

`spatial_grid_stats` summarizes the spatial distribution of visual statistics over a regular grid. It takes an image path and the numbers of grid rows and columns. It returns per-cell measurements together with the resulting grid dimensions.

`legal_text_analyze` extracts structured information from legal text. It takes legal text and returns detected article references, monetary expressions, legal terms, dates, and quoted spans.

`pathology_stain_stats` measures descriptive staining and tissue statistics in pathology images. It takes an image path, an optional bounding box, and an optional minimum component area. It returns tissue coverage, mean color, hematoxylin and eosin statistics when available, and statistics for nucleus-like connected components.

`weather_map_stats` measures color and spatial patterns in weather maps. It takes an image path together with an optional bounding box and an optional number of color clusters. It returns the dominant color palette, spatial grid statistics, and image dimensions.

`cad_program_check` performs static checks on CadQuery, OpenSCAD, or CAD DSL programs. It takes program code and an optional final variable name. It returns syntax and safety status, detected CAD operations and primitives, assigned variables, rule violations, and information about the final variable.

`cad_parameter_extract` extracts explicit design parameters from CAD descriptions or code. It takes CAD-related text and returns detected dimensions, shape terms, and CAD operations.

`pubchem_lookup` retrieves compound information from PubChem. It takes a query, a query type such as name, SMILES, CID, or molecular formula, and an optional maximum number of results. It returns matched compound identifiers and records together with the number of retrieved results.

`name2smiles` converts a compound name into molecular identifiers. It takes a compound name and returns the PubChem CID, canonical and isomeric SMILES, the IUPAC name, and known synonyms.

`smiles2name` converts a SMILES representation into compound names and identifiers. It takes a SMILES string and returns the PubChem CID, canonical SMILES, a preferred name, and alternative names.

`formula_parser` parses a chemical formula and derives basic composition statistics. It takes a molecular formula and returns element counts, total and heavy atom counts, molecular weight, the Hill formula, and any unsupported elements.

`molecule_descriptors` retrieves compound properties and basic molecular descriptors. It takes a compound query and its query type. It returns a structured compound record containing the available molecular properties and descriptors.

`region_measure` segments and measures connected regions in an image. It takes an image path together with optional region, threshold, polarity, and minimum-area settings. It returns the number and properties of detected regions, their coverage fraction, the applied threshold and polarity, and an optional visualization file.

`particle_stats` summarizes particle-level statistics in two-dimensional microscopy images. It takes an image path and an optional bounding box. It returns the particle count, coverage fraction, area statistics, equivalent-diameter statistics, detected regions, and an optional visualization file.

`spectrum_peaks` detects peaks from a one-dimensional projection of a spectrum image. It takes an image path together with optional region, orientation, maximum peak count, and prominence settings. It returns the detected peaks, the peak count, axis information, calibration status, and a confidence estimate.

`xrd_integrate` converts a two-dimensional diffraction image into a radial intensity profile. It takes an image path together with optional region, center, bin count, and inversion settings. It returns the estimated center, radial profile, detected peaks, the integration method, and output units.

`parse_code` extracts structural information from source code. It takes source code, a programming language, and a target identifier or expression. It returns syntax validity, matched target lines, definitions, names, syntax nodes, assignments, conditions, and functions.

`static_analyzer` performs lightweight static analysis of source code. It takes source code, a programming language, and a target. It returns syntax validity, assignment and branch counts, called functions, return counts, target lines, and additional structural information.

`compile_code` checks source code syntax or compilation status. It takes source code, a programming language, and a timeout. It returns the compilation status, syntax status, return code, standard output, and standard error.

`execute_code` executes source code in a controlled subprocess. It takes source code, a programming language, standard input, and a timeout. It returns the process return code, standard output, standard error, timeout status, and available execution or syntax diagnostics.

`trace_execution` traces the execution of a small program. It takes source code, a programming language, standard input, and a timeout. It returns the execution status, standard output, standard error, timeout status, and a line-level execution trace.

`symbolic_execute` extracts symbolic execution cues from source code without performing complete symbolic execution. It takes source code and a programming language. It returns syntax information, assignments, conditions, functions, aliases, path constraints, and related structural information.

`shell_command` executes a restricted shell command in a controlled working directory. It takes a command and a timeout. It returns the working directory, process return code, standard output, standard error, and timeout status.

`read_file` reads a local file or a specified range of lines. It takes a file path, optional starting and ending line numbers, and a maximum character limit. It returns the requested content, the actual line range, the total line count, and file availability information.

`write_file` writes content to an allowed workspace or temporary directory. It takes a sandbox-relative path, the content to be written, and a write mode. It returns the target path, the applied mode, and the write status.

search_text searches for textual evidence in a local corpus or repository. It takes a query, a search root, a file pattern, and a maximum number of results. It returns the matched passages, the number of matches, the resolved search root, and backend information.

run_tests runs a test command in a controlled working directory. It takes a test command and a timeout. It returns the working directory, process return code, standard output, standard error, and timeout status.

screen_parse parses graphical user interfaces by combining OCR, object detection, and layout grouping. It takes an image path, a maximum number of elements, an optional bounding box, and an optional output directory. It returns detected elements, element and label counts, OCR results, detection results, and a structured screen summary.

semantic_match compares a query with visible interface content and candidate labels. It takes an image path, a textual query, optional candidate labels, and an optional bounding box. It returns the best match, similarity scores, candidate information, supporting text, and a confidence estimate.

icon_caption converts a small interface control or icon into a textual description. It takes an image path and an optional bounding box. It returns a generated caption, a summary of the visual evidence used, and a confidence estimate.

verifier checks whether a visual interface supports a candidate claim. It takes an image path, a textual claim, and an optional bounding box. It returns a verdict, a support score, textual evidence, and a confidence estimate.

structural_grounder organizes interface elements into a lightweight spatial layout. It takes an image path together with an optional bounding box and output directory. It returns grouped rows, object and text counts, and the normalized region used for grounding.

grounded_sam2 locates or segments open-vocabulary targets across ordered visual frames. It takes a list of image paths, a textual target prompt, and an operating mode. It returns frame-level observations, entity-organized results, detected-frame counts, visual evidence, optional generated artifacts, and execution status.

object_inventory identifies visible entities in a single frame or a frame sequence. It takes a list of image paths and a scope setting. It returns the detected entities, their count, the number of processed frames, and execution status.

spatial_geometry derives spatial relations from detected object boxes. It takes a list of object boxes and an optional image size. It returns object centers, pairwise spatial relations, the numbers of objects and relations, and execution status.

trajectory_tracking tracks a specified entity across an ordered frame sequence. It takes a list of image paths, an entity name, and a tracking mode. It returns the estimated trajectory, a motion summary, the numbers of tracked and total frames, and execution status.

scene_understanding extracts entities and relations from an ordered sequence of visual observations. It takes a list of image paths, an operating mode, and an optional set of target entities. It returns detected entities and relations, their counts, the number of processed frames, and execution status.

scene_graph_generation constructs a temporal scene graph from ordered visual frames. It takes a list of image paths together with optional target entities and relation types. It returns the scene entities, relations, query scope, relation statistics, and execution status.

temporal_reasoning identifies temporal evidence and event windows in ordered frame sequences. It takes a list of image paths, a question, an operating mode, and an optional maximum number of frames. It returns evidence frames, the temporal type and window, visual evidence, the total number of frames, optional generated artifacts, and execution status.

interaction_detection detects interactions among specified entities by combining object, spatial, and temporal evidence. It takes a list of target entities. It returns detected interaction events, fused evidence, missing evidence when present, the number of detected events, and execution status.

knowledge_retrieval retrieves agricultural knowledge about pathogens, transmission, environmental conditions, symptoms, and control measures. It takes a query, optional candidate entities,

Table 4: Reliability evaluation of the verification modules in Sub-Planner.

Sub-question Verification		
Verifier Decision	**Human Correct**	**Human Incorrect**
General Model Accept	209	12
General Model Reject	41	238
Final Verification		
Metric	**Cases**	**Rate**
Incorrect Revision Accepted	23 / 500	**4.6%**
Correct Revision Rejected	34 / 500	**6.8%**

optional requested fields, and a retrieval depth. It returns retrieved evidence records, the number of results, cache information, and the corpus identifier.

`ground_and_segment` locates and isolates crop organs, lesions, or infected regions. It takes an image path, a textual query, and a maximum number of returned regions. It returns detected bounding boxes, confidence scores, area ratios, cropped regions, the number of masks, and the number of detections.

`visual_semantic_match` ranks textual visual hypotheses according to their similarity to an image. It takes an image path, a set of candidate texts, a retrieval depth, and an optional output directory. It returns the ranked candidates and information about the visual matching backend.

`visual_evidence_match` matches visible agricultural symptoms against external visual evidence. It takes an image path and a retrieval depth. It returns the matched evidence records and information about the visual matching backend.

A.4 Verification Reliability Analysis

To evaluate the reliability of the general model as a verifier, we first explain the motivation for using a general model for verification. Specialized models are optimized for specific domains but may produce incorrect answers due to domain-specific reasoning errors. In such cases, using the same specialized model for verification may make it difficult to identify these errors. In contrast, a general model can often recognize inconsistencies or logical errors without requiring detailed domain-specific knowledge. Therefore, we use the general model as an independent verifier to assess the outputs of specialized models.

We conduct a manual annotation study on both sub-question verification and final verification stages. For sub-question verification, we randomly sample 500 sub-question verification cases and construct a balanced evaluation set containing 250 correct and 250 incorrect cases annotated by humans. As shown in Table 4, the verifier successfully rejects 238 out of 250 incorrect intermediate results, while only 12 incorrect results are mistakenly accepted. Meanwhile, 209 out of 250 correct intermediate results are retained, demonstrating that the verifier effectively filters unreliable reasoning while preserving most valid intermediate results. For final verification, we independently sample another 500 answer revision cases and manually inspect whether the verifier correctly identifies revision outcomes. We find that only 23 incorrect revisions are mistakenly accepted, corresponding to an incorrect revision acceptance rate of 4.6%. These results demonstrate that the general model verifier provides reliable verification for both intermediate reasoning and final answer revision stages.