MTA-Agent: An Open Recipe for Multimodal Deep Search Agents

Xiangyu Peng[*], **Can Qin**[*], **An Yan**[†], **Xinyi Yang**[†], **Ran Xu**, **Zeyuan Chen**, **Chien-Sheng Wu**
Salesforce Research
becky.peng@salesforce.com
mta-agent.github.io

Abstract

Multimodal large language models (MLLMs) have demonstrated strong capabilities in visual understanding, yet they remain limited in complex, multi-step reasoning that requires deep searching and integrating visual evidence with external knowledge. In this work, we address this challenge by constructing high-quality, verified multi-hop vision-language training data for multimodal deep-search agents. We propose a **M**ulti-hop **T**ool-**A**ugmented **Agent** for Evidence-based QA Synthesis (`MTA-Agent`), which automatically selects tools and their parameters to retrieve and validate evidence from both visual and textual sources and generates structured multi-hop question answer trajectories. Starting from diverse VQA seed datasets, our pipeline produces a large-scale training dataset, `MTA-Vision-DeepSearch`, containing 21K high-quality multi-hop examples. The data is filtered through a multi-stage verification process to ensure factual consistency and answer uniqueness. Using `MTA-Vision-DeepSearch`, a 32B open-source multimodal search agent achieves state-of-the-art performance, reaching an average of 54.63% across six challenging benchmarks, outperforming GPT-5 (51.86%), Gemini-2.5-Pro (50.98%), and Gemini-3-Pro (54.46%) under the same tool settings. We further show that training on our data improves both reasoning depth and tool-use behavior, increasing the average number of steps from 2.27 to 4.28, and leading to more systematic and persistent search strategies. Additionally, we demonstrate that training can be performed without real-time tool calls by replaying cached interactions, significantly reducing training cost. Importantly, we present `MTA-Agent` as a fully open recipe for multimodal deep search: we release the entire dataset, training trajectories, and implementation details to enable reproducibility and future research on open multimodal search agents.

1 Introduction

Multimodal large language models (MLLMs) have achieved strong performance across a wide range of vision tasks (Xue et al., 2024; DeepSeek-AI, 2025; Huang et al., 2025; Qwen-Team, 2025). However, their performance is often limited by the parametric knowledge, making them struggle with complex, fact-intensive visual question answering (VQA) that requires up-to-date or long-tail real-world information (Geng et al., 2025; Chng et al., 2025; Wu et al., 2025). While proprietary systems have made progress by integrating ReAct-style (Yao et al., 2022) reasoning with tool use, multimodal deep search remains largely underexplored, with few existing agents capable of handling high-difficulty vision-language tasks (Xu & Peng, 2025; Hu et al., 2025). We argue that the primary bottleneck is the lack of high-quality training data needed to transfer the strong VQA capabilities of MLLMs to more challenging deep-search scenarios.

[*]Same Contribution
[†]Same Contribution

Current approaches for transferring MLLMs into effective multimodal deep search systems still face three major limitations. First, existing open-source models used as the backbone of deep-search agents are constrained in both reasoning depth and search breadth (Fu et al., 2025; Wu et al., 2025; Narayan et al., 2025). For example, before training, `Qwen3-VL-32B-Instruct` (Qwen-Team, 2025) performs only 1.76 tool-use steps on average on the MMSearch-VL benchmark (Jiang et al., 2024), achieving 68.52% accuracy. In comparison, GPT-5 achieves 77.65% accuracy with an average of 2.60 search steps. The absence of longer and more challenging training data for the underlying LLM backbone further limits the agent's ability to solve questions that require aggregating evidence from multiple sources. Second, the lack of high-quality training data remains a major bottleneck. Existing data synthesis approaches often produce inconsistencies between reasoning processes and final answers, which hinders effective learning (Li et al., 2024; Wu et al., 2025). In contrast to VQA datasets, which have significantly advanced MLLM capabilities, high-quality datasets for multimodal deep search remain scarce. Third, training data for deep search lacks diversity. Most datasets are constructed from single images with web search, resulting in limited coverage of complex and multi-step scenarios (Geng et al., 2025). Such limited and homogeneous data encourages agents to rely on a single search strategy, restricting their ability to generalize to more complex reasoning tasks.

To address these limitations, we propose a multi-hop, tool-augmented agent for evidence QA synthesis framework (**`MTA-Agent`**). Unlike existing multimodal deep search methods that often rely on shallow retrieval or limited reasoning patterns, our approach explicitly targets long-horizon, multi-step reasoning grounded in both visual and textual evidence. Our key idea is to transform existing high-quality VQA resources into challenging multi-hop search tasks, enabling richer reasoning scenarios and more diverse tool usage. This allows us to preserve strong visual understanding while significantly increasing the complexity of the reasoning process. We further introduce a tool-augmented agent that performs iterative search and reasoning, leveraging multiple tools and flexible interaction strategies rather than relying on a single retrieval pattern. This design encourages more robust and diverse exploration during evidence collection. To ensure reliability, we incorporate a verification pipeline that filters out inconsistent or shortcut reasoning, resulting in high-quality multi-hop reasoning trajectories. Finally, using the synthesized dataset (**`MTA-Vision-DeepSearch`**), we train a single 32B dense model with RL, achieving state-of-the-art performance across six deep-search benchmarks and surpassing strong commercial search agents. Detailed technical design is provided in Section 3.

The main contributions of this work are as follows:

1. We propose a multi-hop tool-augmented evidence QA synthesis agent (`MTA-Agent`) as a fully open recipe for multimodal deep search (§3), which automatically generates high-quality multi-hop QA data from existing VQA datasets without human efforts.
2. We release a 21K high-quality training dataset (`MTA-Vision-DeepSearch`) for multimodal deep-search agents and human-verified and rewritten test set for evaluation (§ 3.3).
3. We show that our dataset enables a 32B open-source model to achieve state-of-the-art performance, even outperforming leading commercial models (§4). We also release rollout histories, which enable tool-free training and further improve performance compared to training with real tool calls on other existing training dataset (§5.3).
4. We conduct extensive analysis (§5), including the impact of training data choices, strategies to improve data quality, and changes in tool usage behavior during training.

2 Related Works

Deep Search Agents. Recent advancements in LLM agents mark a shift from simple reactive or retrieval-based chatbots (Chen et al., 2024; Yu et al., 2024) to autonomous, goal-directed systems that can plan, self-correct, and execute complex workflows. To support this increased autonomy, modern agents are tightly integrated with tool-use capabilities and secure execution environments (OpenAI, 2025; Wu et al., 2026; Li et al., 2025). Song et al. (2025) propose R1-Searcher, a two-stage outcome-based RL approach to enhance search capabilities. Wu et al. (2026) develop DeepResearch-R1, an open-source training framework

that supports multi-turn web interactions and diverse reward models, including rule-based outcomes and LLM-as-judge feedback. Tao et al. (2025b) introduce a formalization-driven data synthesis framework that enables precise control over reasoning structures through knowledge projectionoperations. Despite these advances, most deep-search agents (Yao et al., 2026) remain primarily text-centric. MMSearch-R1 (Wu et al., 2025) extends search to multimodal settings by enabling dynamic image and text retrieval, while Geng et al. (2025) improve generalization through synthetic trajectories. However, training data creation for these systems often requires substantial human effort, and many works do not release their original training datasets. As a result, existing training data remains limited in scale and diversity, restricting the development of multimodal deep search agents that can learn diverse and robust search strategies.

Multimodal Deep Search Data. Most existing VQA datasets primarily evaluate single-step reasoning or shallow retrieval capabilities. For example, OK-VQA (Marino et al., 2019) requires commonsense reasoning over everyday scenes, while MMT-Bench (Ying et al., 2024) focuses on expert-level knowledge and fine-grained visual understanding. More recent datasets aim to improve and evaluate deep-search capabilities. LiveVQA (Fu et al., 2025) is designed to test reasoning over up-to-date, real-world visual information, and FVQA (Wu et al., 2025) requires the use of external knowledge. However, these tasks are relatively simple and can often be solved with only one or two search steps. Recent efforts (Li et al., 2024; Chen et al., 2025; Huang et al., 2026; Zeng et al., 2026) such as MMSearch (Jiang et al., 2024) and MMSearch-Plus (Tao et al., 2025a) provide more challenging benchmarks for multimodal search. However, they are primarily designed for evaluation and do not offer large-scale training data. To address these limitations, we propose a data generation agent that interacts with four tools to construct single-hop QA pairs, which are then composed into multi-hop questions through a carefully designed verification process. Our approach produces high-quality training data that is scalable and allows explicit control over task difficulty and domain diversity.

3 Data Creation

In this section, we describe how we construct a QA agent to generate training data. We first filter existing VQA datasets to serve as seeds (§3.1). Next, we design a QA agent (§3.1) that selects tools and their parameters to collect feedback and generate single-hop question answer pairs. We further implement a validation system to ensure that the synthetic QA data is factual, unique, and high-quality for deep search. These single-hop questions are connected through entities and merged into multi-hop questions, which are used to train multimodal deep-search agents.

3.1 VQA Seed Selection

We use the training splits of FVQA (Wu et al., 2025), LiveVQA News (Fu et al., 2025), InfoVQA (Mathew et al., 2022), InfoSeek (Chen et al., 2023), and OK-VQA (Marino et al., 2019) as seed datasets. FVQA and LiveVQA contain news images and represent relatively easy deep-search settings. InfoVQA requires OCR over infographic images, InfoSeek involves Wikipedia images and external knowledge retrieval, and OK-VQA requires commonsense reasoning over complex everyday scenes. Ablation studies for the design choices are presented in Section 5.1.

Each VQA sample is processed through a multi-stage filtering pipeline. First, GPT-5-mini verifies that the question requires visual information, discarding those answerable from general knowledge. Second, GPT-5 rewrites the question into a clean, image-grounded free-form format and adds a short disambiguating entity description (e.g., "Birmingham, city in Alabama"). Questions that cannot be converted from multiple-choice are rejected. Third, GPT-5 ensures the answer is a specific proper-noun entity from predefined categories, filtering out ambiguous or generic terms. An *entity* is defined as a uniquely named proper noun (e.g., person, organization, location, product, event, or creative work), rather than a description, number, or multi-part answer (see Appendix A.1). Finally, GPT-5 with

vision and web search verifies factual correctness; samples without a definitive answer are discarded. Full prompts and predefined categories are provided in Appendix A.2.

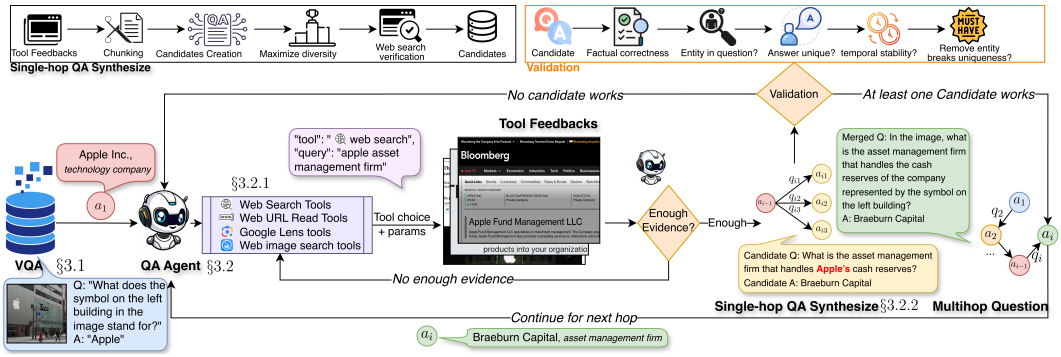

Figure 1: Data creation pipeline. Starting from filtered seed VQA data, the QA agent iteratively selects tools to collect sufficient feedback for generating single-hop QA candidates. These candidates are verified, and the most diverse valid candidate is appended to the multi-hop reasoning chain. If no candidate passes verification, the agent continues to gather additional evidence by selecting tools. Detailed examples are shown in Appendix D.

3.2 QA Agent

We design a QA agent with two types of actions: tool use (§3.2.1) and QA generation (§3.2.2). The agent follows a **ReAct** framework (Yao et al., 2022), where it first generates reasoning and then selects an action from either tool use or QA generation. If it selects tool use, it further chooses a specific tool along with its parameters (e.g., search queries or image URLs). If it selects QA generation, it triggers the process described in § 3.2.2. The agent conditions on the full interaction history, including prior reasoning steps, tool selections, tool feedback, and previously generated questions.

As shown in Figure 1, the agent operates as follows. (1) It starts from a seed VQA instance consisting of an image, a question q_1, and an entity answer a_1. (2) The agent iteratively selects and uses tools to gather additional evidence (§3.2.1). (3) Once the QA agent determines that sufficient evidence has been collected, it will invokes the QA generation module (§3.2.2) by to produce a new single-hop QA pair (q_k, a_k). If the generated QA pair passes validation as described in Section 3.2.2, the new question is merged with the existing question chain to form an extended multi-hop question $\tilde{q}_k$. If the QA pair fails validation, the failure reason is summarized and used to guide subsequent tool calls, with modified queries or URLs to collect additional evidence. This process repeats iteratively, gradually extending the reasoning chain until a K-hop question is constructed. Examples are shown in Appendix D.

3.2.1 *Tool Use for QA Generation*

The agent is equipped with four tools: (1) **web search**, which uses Tavily [1] to retrieve up to 10 URLs with titles and snippets given a text query; (2) **web reader**, which retrieves the full content of a given URL; (3) **Google Lens**, which searches for visually similar images and returns matched results with titles and snippets given an image; (4) **image search**, which retrieves image URLs given a text query. We do not include OCR tools, as this capability is already covered by the VQA seed data. Additional details are provided in Appendix A.3.

3.2.2 *Single-hop QA Creation Module*

For each hop $k \in \{2, \ldots, K\}$, given the bridge entity a_{k-1} (the answer from the previous hop), we generate a new question q_k and answer a_k through a four-stage procedure. Ablation studies of this procedure are presented in Appendix C.1.

[1] https://www.tavily.com

Candidate Generation. The web content associated with a_{k-1} is segmented into chunks of at most 5000 tokens. Each chunk is scored based on language quality (e.g., sentence completeness; see Appendix A.4) and whether it contains a_{k-1}. We retain the top sources containing relevant information. For each source, GPT-5.1 extracts a candidate triple (q_k, a_k, c_k) (see Appendix A.5.2), where c_k is a supporting excerpt. The prompt enforces that: (1) q_k explicitly includes a_{k-1}; (2) a_k is a unique entity distinct from $\{a_1, \ldots, a_{k-1}\}$; (3) q_k is standalone; (4) removing a_{k-1} makes the answer indeterminate; (5) q_k has a single correct answer; and (6) q_k captures a meaningful relation or attribute. Each candidate is then verified for factual correctness against c_k, answer uniqueness, temporal stability, and entity dependency; invalid candidates are discarded. Further details are in Appendix A.5.

Verification. The selected pair (q_k, a_k) is verified by GPT-5 with live web search against five criteria: (1) factual correctness with respect to the source content, (2) explicit presence of e_{k-1} in q_k, (3) unique answerability, (4) temporal stability of a_k, and (5) entity dependency (removing e_{k-1} breaks uniqueness). The candidate is accepted only if all criteria are satisfied; otherwise it is discarded and generation continues with other chunks. Using a search-augmented verifier (GPT-5 with web search tools) provides an independent factual check beyond the generation source. More details can be found in Appendix A.5.4.

Answer Diversity Selection. When multiple candidates exist, we select the one that produces the most challenging merged question. For each candidate, Qwen3-VL-32B (Qwen-Team, 2025) is asked what search query (r_{weak}) it would use to answer the merged question $\tilde{q}_k$. The weak model's query r_{weak} is compared against the actual retrieval query r_{actual} via token-overlap Jaccard similarity (after stop-word removal):

$$\text{sim}(r_{\text{weak}}, r_{\text{actual}}) = \frac{|\mathcal{T}(r_{\text{weak}}) \cap \mathcal{T}(r_{\text{actual}})|}{|\mathcal{T}(r_{\text{weak}}) \cup \mathcal{T}(r_{\text{actual}})|}$$

We select the candidate with the lowest similarity score; if all candidates exceed a threshold of 0.6, the hop is rejected and a new search is attempted. Answer diversity across hops is enforced separately: the synthesis prompt lists all previous answers $\{a_1, \ldots, a_{k-1}\}$ and instructs the model to produce a distinct answer. More details are in Appendix A.6.

Answer Disambiguation. Because short entity answers (e.g., *Apple*) may be ambiguous for the content search for next hop, we append a short disambiguating phrase using GPT-5.1. The model adds a short phrase from the supporting context c_k that uniquely identify the entity (e.g., *Apple Inc., a technology company*). The disambiguated form $\tilde{a}_k$ is used as the seed for search query for hop $k+1$, while the original a_k remains the ground-truth answer label. See more details in Appendix A.7.

Anti-leakage Check. After merging q_k with the existing reasoning chain to form the intermediate multi-hop question, we perform an anti-leakage test using Tavily. For example, consider the question: *"Where is Donald Trump's hometown in the capital of the country shown in the image?"* The country (the United States) can be easily inferred. Such cases are filtered out to prevent information leakage and to ensure that the final questions require genuine multi-hop reasoning and image reading. Specifically, we query Tavily with the merged question and check whether the system can directly retrieve the final answer. If the answer can be obtained, the sample is discarded and generation continues with alternative candidates.

3.3 Data Statistics

For each reasoning chain, we retain all intermediate question–answer pairs, including 2-hop, 3-hop, and 4-hop cases. This allows the model to progressively learn from simpler to more complex reasoning tasks. Ablation studies for this design choice are presented in Section 5.1. Table 1 summarizes the training data statistics. In total, the dataset contains 21K training examples. The ground-truth answers are short and can be reliably verified by a small language model. We also construct a test set of 178 examples, which are manually verified and rewritten by human annotators to evaluate the capability of deep-search agents. The average LLM cost per sample is \$0.28, with an average of 9.3 tool calls per multi-hop QA generation. Detailed examples are shown in Appendix D.

Seed Dataset	Original	2-hop	3-hop	4-hop	Total	Test	Cost	Tools
FVQA	4,750	1,155	1,153	1,113	8,171	33	$ 0.30	9.8
Infoseek	1,942	1,135	1,130	934	5,141	51	$ 0.25	8.4
InfoVQA	217	483	481	400	1,581	38	$ 0.28	9.8
News	918	240	238	236	1,632	13	$ 0.27	8.6
OK-VQA	629	1,418	1,415	1,168	4,630	43	$ 0.31	10.1
Total	8,456	4,431	4,417	3,851	21,155	178	$ 0.28	9.3

Table 1: Number of scenarios per hop in our training and test data. "Cost" and "Tools" indicate the average LLM cost and number of tool calls per sample for generation.

Human verification We conduct human evaluation to verify the quality of our generated training data, based on 444 multi-hop trajectories. Each trajectory is annotated by three human annotators at both the single-hop and merged multi-hop levels. We evaluate the data along three dimensions: (1) **Understandability**: whether the final multi-hop question is clear and understandable to humans. Annotators find that 97.7% of the questions are clearly understandable. (2) **Factuality**: whether each single-hop QA pair is factually correct. 89.4% of the hop-level answers are verified as correct. (3) **Image requirement**: whether the question requires visual information to answer. 76.3% of the questions are verified to require the image, meaning the answer cannot be determined without visual input. Although a subset of questions does not strictly require visual input, these examples are still useful for training, as they help improve the model's general search and reasoning capabilities. See more details about human evaluation in Appendix B.1

4 Experiments

4.1 Experiment Setup

4.1.1 Evaluation Datasets

We evaluate our approach and all baselines on six deep-search-oriented benchmarks, which represent the current state-of-the-art in multimodal search evaluation: MMSearch (Jiang et al., 2024), MMSearch-Plus (Tao et al., 2025a), BrowseComp-VL (BC-VL) (Geng et al., 2025), HR-MMSearch (Chng et al., 2025), the FVQA test split (Wu et al., 2025), and our newly constructed test dataset, `MTA-Vision-DeepSearch`-test (MTA-test).

4.1.2 Search Agent Setup

We adopt a **ReAct** (Reasoning and Acting) framework (Yao et al., 2022) for our multimodal deep search agent. At each step, the agent receives the research question alongside the input image, reasons over its current knowledge state, selects a tool to execute, and incorporates the resulting observation before deciding on the next action (Check Appendix B.2).

ReAct Loop. The agent operates for up to $T = 6$ iterations. At each iteration, the agent produces a structured response containing: (i) a *reasoning* trace over the current state, (ii) an *action* specifying a tool and its parameters, (iii) a binary `should_stop` flag, and (iv) a scalar *confidence* score. The loop terminates early if `should_stop=true` with confidence above 0.7, or if two consecutive tool calls fail. Upon termination, the agent generates a final answer conditioned on the full dialogue history. More details can be found in Appendix B.2.3.

Tools. The agent has access to seven tools: *Web search* retrieves ranked web snippets, titles, and URLs. *Web image search* returns text descriptions of image results from the web. *Web URL reader* extracts the full text content of a given webpage. *Reverse image search* leverages Google Lens to identify objects and landmarks and retrieve visually similar images. *OCR* extracts all visible text from the image. *Python execution* runs code in a persistent IPython session for data analysis and computation. All tool calls are dispatched to a local observation server and the returned text observation is appended to the dialogue history as a user turn and fed back to the agent in the next iteration. Details are in Appendix B.2.2. Note that tools not used in the training QA generation pipeline (Section 3) are included to support

downstream question answering and out-of-distribution questions, but are not required during QA construction.

4.1.3 *Baselines*

We compare our fine-tuned model with three commercial large language models: GPT-5 (Singh et al., 2025), Gemini-2.5-Pro and Gemini-3-Pro (DeepMind, 2025). For each model, we evaluate two settings: (1) direct answering without tools, and (2) the same agent framework with the identical tool setup used in our approach. We also compare against two open-source models, Qwen3-VL-8B and Qwen3-VL-32B, evaluated both with and without the same agent tool configuration. In addition, we report benchmark results from prior work. However, different papers adopt different tool setups, we directly list the numbers they reported in their original papers. Specifically, we compare with SenseNova-MARS (Chng et al., 2025), WebWatcher (Geng et al., 2025), MM-DeepResearch 32B (Yao et al., 2026) and MMSearch-R1-7B (Wu et al., 2025).

4.1.4 *Training Setup*

We fine-tune two models, Qwen3-VL-32B-Instruct and Qwen3-VL-8B-Instruct (Qwen-Team, 2025), using the training data summarized in Table 1, resulting in two models: `MTA-DeepSearch-32B` and `MTA-DeepSearch-8B`. Training is performed with DAPO (Yu et al., 2025), using a batch size of 64, 8 rollout samples per prompt, a learning rate of 2×10^{-6}, and 100 training steps. Each rollout uses a ReAct-style agent that interacts with tools for up to 6 steps. Observation tokens are masked from the policy gradient loss. Rewards are computed asynchronously using an LLM-based judge (see Appendix B.3.). The 32B and 8B models are trained on 32 and 8 H100 GPUs, respectively, using the `verl-tool` framework[2].

4.2 Main Results

Potential Dataset Contamination in Model Training. Some datasets may have been inadvertently included in the training data of certain models. For example, MM-Search appears to be present in the training corpus of Gemini-3-Pro: while Gemini-2.5-Pro achieves 39.8% on this benchmark, Gemini-3-Pro reaches 65.88% even without the use of external search tools. This discrepancy suggests that performance gains may partially stem from prior exposure rather than improved reasoning or retrieval capabilities. Consequently, the reported results of such commercial LLMs on these benchmarks may not accurately reflect their true deep search abilities.

Training Improves Agent Performance and Scales Effectively. Table 2 demonstrates that training on `MTA-Vision-DeepSearch` significantly improves the performance of the base models. Compared to models without training but equipped with same tools (§4.1.2), `MTA-DeepSearch-8B` achieves an average improvement of 12.08%, while `MTA-DeepSearch-32B` achieves a 13.78% gain in benchmark accuracy. Performance scales consistently with model size: the 32B model outperforms the 8B variant both before and after training, while RL training yields complementary gains across scales, demonstrating the scalability of our approach.

Competitive Performance Against State-of-the-Art Search Agents. Compared to specialized multimodal deep search agents (see the Multimodal Deep Search Agents rows in Table 2), our approach achieves competitive or superior results without requiring task-specific architectures, highlighting the generality of our training paradigm. Furthermore, when compared to state-of-the-art commercial LLMs under the same tool setting, our fine-tuned model `MTA-DeepSearch-32B` also achieves comparable or even superior performance. Specifically, it outperforms GPT-5 by 2.78% and Gemini-2.5-pro by 3.65%. Even when compared to Gemini-3-pro, which may benefit from potential training data contamination, our model achieves comparable performance (54.63% vs. 54.46%).

[2] https://github.com/TIGER-AI-Lab/verl-tool

Model	MMSrch-Plus	HR-MMSrch	BC-VL	MMSrch	FVQA	MTA-test	Avg
Direct Answer							
GPT-5	13.18	16.39	41.60	36.47	50.83	21.35	29.97
Gemini-2.5-pro	12.22	15.41	39.85	41.76	54.50	21.91	30.94
Gemini-3-pro	26.37♠	23.61	41.35	65.88♠	58.94	26.97	40.52
Qwen3-VL-8B-Inst.	1.29	4.59	20.05	12.35	26.94	6.18	11.90
Qwen3-VL-32B-Inst.	2.25	4.59	24.81	17.65	24.81	11.24	14.22
Agent Workflow							
GPT-5	31.61	52.13	51.63	77.65	72.28	25.84	51.86
Gemini-2.5 Pro	30.65	48.20	49.50	77.65	72.33	27.53	50.98
Gemini-3 Pro	**33.51**	53.20	<u>51.78</u>	**82.94**	**76.67**	<u>28.65</u>	<u>54.46</u>
Qwen3-VL-8B-Inst.	18.38	32.13	35.89	57.06	64.22	11.80	36.58
Qwen3-VL-32B-Inst.	14.84	38.69	38.69	68.52	66.94	17.42	40.85
Multimodal Deep Search Agents							
MMSearch-R1-7B	-	20.33†	-	53.80†	58.40†	-	-
SenseNova-MARS-8B	-	41.64†	-	67.84†	67.11†	-	-
SenseNova-MARS-32B	-	**54.43**†	-	74.27†	72.61†	-	-
Webwatcher-7B	-	-	20.30†	49.10†	-	-	-
Webwatcher-32B	-	-	26.70†	55.30†	-	-	-
MM-DeepResearch 32B	-	-	43.00†	69.00†	70.10†	-	-
Ours							
`MTA-DeepSearch-8B`	26.77	47.54	44.36	79.41	73.06	20.79	48.66 (+12.08)
`MTA-DeepSearch-32B`	<u>31.93</u>	<u>53.95</u>	**53.77**	<u>82.35</u>	<u>76.00</u>	**29.78**	**54.63** (+13.78)

Table 2: Accuracy (%) on deep-search benchmarks. Best results are in **bold**, second-best are <u>underlined</u>, † denotes results reported in the original papers, and ♠ indicates potential dataset contamination.

Data Choice	MMSrch-Plus	BC-VL	MMSrch	HR-MMSrch	Avg.
Qwen3-VL-8B-Inst.	18.38	35.89	57.06	32.13	35.87
w/ F+L	20.96	43.22	73.46	43.28	45.23
w/ F + L + 4-hops (F+L)	20.32	43.35	74.12	43.60	45.35
w/ F + L + 2/3/4 hops (F + L)	20.96	44.11	75.29	44.61	46.24
w/ F + L + 2/3/4 hops (Full)	**26.77**	**44.36**	**79.41**	**47.54**	**49.52**

Table 3: Impact of different training data choices for Qwen3-VL-8B-Inst. as the search agent backbone, using the same settings (100 steps, batch size = 64, rollout = 8).

5 Analysis

5.1 Effect of the Data choice

We first train the search agent backbone using the FVQA and LiveVQA-News datasets (F+L). Table 3 shows that this setting improves performance over the Qwen3-VL-8B-Inst. (Qwen-Team, 2025) baseline without training, increasing the average score from 35.87% to 45.23% across all benchmarks. However, these datasets mainly contain news-related images, which are relatively simple and lack diversity. To increase task difficulty, we extend the training data with generated 4-hop questions (F+L+4-hops (F+L)). This leads to only limited improvement, with the average score increasing slightly from 45.23% to 45.35%, suggesting that these 4-hop questions are too challenging when used alone. To address this, we further include intermediate 2-hop and 3-hop questions together with 4-hop questions (F+L+2/3/4 hops). As shown in the table, this results in consistent performance gains, improving the average score to 46.24%. Finally, the full data mixture (F+L+2/3/4 hops (Full)), which incorporates InfoVQA, Infoseek, and OK-VQA, achieves the best results across all benchmarks, reaching an average of 49.52%. This highlights the importance of both diverse seed VQA data and a balanced range of question difficulties for training an effective search agent.

Model	MMSrch-Plus	HR-MMSrch	BC-VL	MMSrch	FVQA	MTA-test	Avg
Qwen3-VL-8B-Inst.	18.38	32.13	35.89	57.06	64.22	11.80	36.58
+ FVQA & LiveVQA	20.96	43.28	43.22	73.46	71.06	12.36	44.06
+ MTA-DS (replay)	27.62	43.61	43.86	76.47	71.33	15.17	46.34
+ MTA-DS (tool-call)	26.77	47.54	44.36	79.41	73.06	20.79	48.66

Table 4: Accuracy (%) on search-oriented benchmarks with and without external tool calls.

5.2 Analysis of Tool Usage

We analyze how RL training reshapes both search depth (number of turns) and tool-use strategy, compared to the pre-trained model.

Tool use. As shown in Figure 2, before training, the model uses Web Search (72%) and Reverse Image Search (55%) at moderate rates with no clear retrieval strategy, indicating heuristic and unstructured tool selection. After RL training, tool use becomes highly structured: Web Search is nearly universal (99%) and Reverse Image Search rises to 79%, forming a consistent two-stage retrieval strategy (text search followed by visual grounding). Content Extraction decreases to 24% and OCR drops to 16%, suggesting a shift away from low-level image parsing toward active web-based retrieval.

Search depth. As shown in Figure 3, before training, the model exhibits a strongly left-skewed turn distribution: 32% of trajectories stop after one turn and 30% after two, with an average of 2.27 turns, indicating shallow and non-persistent search. After RL training, the distribution shifts substantially rightward, with the largest share of trajectories reaching the 6-turn limit (32%), followed by peaks at 3 (24%) and 4 turns (21%). The average increases to 4.28, demonstrating sustained multi-step reasoning across all benchmarks.

The pre-trained multimodal language model behaves as a shallow agent with inconsistent tool use. RL training transforms it into a systematic multi-step search agent with longer trajectories and a consistent retrieval pipeline. All results are in Appendix C.2.

5.3 Training without External Tool Calls

Training a search agent is expensive, as it requires not only GPUs but also frequent tool usage (e.g., paid web search APIs). To reduce this cost, we explore whether the data generated during QA creation can be reused for RL training. We store all rollout histories from the data creation pipeline and ablation studies in a replay dictionary, where keys correspond to tool inputs (parameters) and values correspond to tool outputs. During training, instead of calling external tools, we retrieve responses from this dictionary. Specifically, we compute the cosine similarity between the current tool query and stored keys, and select the most similar entry if the similarity exceeds 0.75. If no match is found, a tool error is returned. As shown in Table 4, this replay-based approach outperforms the baseline and even achieves better performance than training with FVQA and LiveVQA data using real tool calls. This

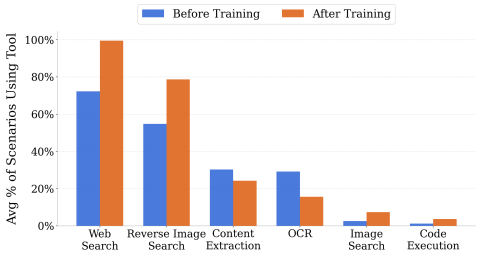

Figure 2: Tool usage frequency across scenarios on all test datasets. Each bar shows the percentage of scenarios in which a given tool is used at least once.

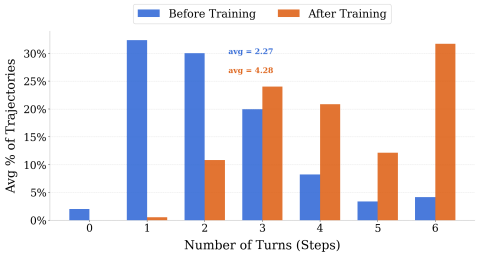

Figure 3: Turn distribution across scenarios on all test datasets. Each bar shows the percentage of trajectories that terminate at a given number of turns.

result suggests that, with our constructed dataset, it is possible to train an effective search agent without incurring additional tool costs, enabling efficient training of open-source models. More details about how we create the cache are shown in Appendix A.8.

6 Conclusion

In this work, we explore how to address a key limitation of Multimodal Large Language Models (MLLMs): their difficulty in complex, multi-step visual reasoning that requires deep search and evidence integration. We propose `MTA-Agent`, a synthesis framework that generates high-quality multi-hop vision-language trajectories. Using a tool-augmented agent over existing VQA datasets, we construct `MTA-Vision-DeepSearch`, a 21K-example dataset designed for long-horizon reasoning. RL training on this data significantly improves open-source models. Our `MTA-DeepSearch-32B` achieves state-of-the-art performance across six benchmarks, outperforming GPT-5 and Gemini-3-Pro under comparable tool settings. It also improves search behavior, increasing average search depth from 1.76 to 3.62 turns and enabling more systematic multi-step retrieval. We further show that training via interaction replay is effective, reducing the cost of tool usage. Our results highlight three key insights: (1) high-quality multi-hop data enables strong long-horizon reasoning; (2) data diversity and verification are critical for performance; and (3) replay-based training improves efficiency without sacrificing accuracy. By releasing our open recipe, dataset, test suite, and rollout histories, we provide a foundation for future multimodal deep-search agents.

References

Yang Chen, Hexiang Hu, Yi Luan, Haitian Sun, Soravit Changpinyo, Alan Ritter, and Ming-Wei Chang. Can pre-trained vision and language models answer visual information-seeking questions? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14948–14968, 2023.

Zhanpeng Chen, Chengjin Xu, Yiyan Qi, and Jian Guo. Mllm is a strong reranker: Advancing multimodal retrieval-augmented generation via knowledge-enhanced reranking and noise-injected training. *arXiv preprint arXiv:2407.21439*, 2024.

Zhuo Chen, Xinyu Wang, Yong Jiang, Zhen Zhang, Xinyu Geng, Pengjun Xie, Fei Huang, and Kewei Tu. Detecting knowledge boundary of vision large language models by sampling-based inference. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 28665–28680, 2025.

Yong Xien Chng, Tao Hu, Wenwen Tong, Xueheng Li, Jiandong Chen, Haojia Yu, Jiefan Lu, Hewei Guo, Hanming Deng, Chengjun Xie, et al. Sensenova-mars: Empowering multimodal agentic reasoning and search via reinforcement learning. *arXiv preprint arXiv:2512.24330*, 2025.

Google DeepMind. Gemini 3. https://deepmind.google, 2025. Large language model.

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models, 2025.

Mingyang Fu, Yuyang Peng, Dongping Chen, Zetong Zhou, Benlin Liu, Yao Wan, Zhou Zhao, Philip S Yu, and Ranjay Krishna. Seeking and updating with live visual knowledge. *arXiv preprint arXiv:2504.05288*, 2025.

Xinyu Geng, Peng Xia, Zhen Zhang, Xinyu Wang, Qiuchen Wang, Ruixue Ding, Chenxi Wang, Jialong Wu, Yida Zhao, Kuan Li, et al. Webwatcher: Breaking new frontier of vision-language deep research agent. *arXiv preprint arXiv:2508.05748*, 2025.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, et al. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. *arXiv preprint arXiv:2505.23885*, 2025.

Peizhou Huang, Zixuan Zhong, Zhongwei Wan, Donghao Zhou, Samiul Alam, Xin Wang, Zexin Li, Zhihao Dou, Li Zhu, Jing Xiong, et al. Mmdeepresearch-bench: A benchmark for multimodal deep research agents. *arXiv preprint arXiv:2601.12346*, 2026.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Xu Tang, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.

Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanmin Wu, Jiayi Lei, Pengshuo Qiu, Pan Lu, Zehui Chen, Chaoyou Fu, Guanglu Song, et al. Mmsearch: Benchmarking the potential of large models as multi-modal search engines. *arXiv preprint arXiv:2409.12959*, 2024.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025.

Yangning Li, Yinghui Li, Xinyu Wang, Yong Jiang, Zhen Zhang, Xinran Zheng, Hui Wang, Hai-Tao Zheng, Philip S Yu, Fei Huang, et al. Benchmarking multimodal retrieval augmented generation with dynamic vqa dataset and self-adaptive planning agent. *arXiv preprint arXiv:2411.02937*, 2024.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. Infographicvqa. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1697–1706, 2022.

Kartik Narayan, Yang Xu, Tian Cao, Kavya Nerella, Vishal M Patel, Navid Shiee, Peter Grasch, Chao Jia, Yinfei Yang, and Zhe Gan. Deepmmsearch-r1: Empowering multimodal llms in multimodal web search. *arXiv preprint arXiv:2510.12801*, 2025.

OpenAI. Introducing deep research, 2025. URL https://openai.com/index/introducing-deep-research/.

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

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

Xijia Tao, Yihua Teng, Xinxing Su, Xinyu Fu, Jihao Wu, Chaofan Tao, Ziru Liu, Haoli Bai, Rui Liu, and Lingpeng Kong. Mmsearch-plus: Benchmarking provenance-aware search for multimodal browsing agents. *arXiv preprint arXiv:2508.21475*, 2025a.

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, et al. Webshaper: Agentically data synthesizing via information-seeking formalization. *arXiv preprint arXiv:2507.15061*, 2025b.

Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. Mmsearch-r1: Incentivizing lmms to search. *arXiv preprint arXiv:2506.20670*, 2025.

Tongzhou Wu, Yuhao Wang, Xinyu Ma, Xiuqiang He, Shuaiqiang Wang, Dawei Yin, and Xiangyu Zhao. Deepresearch-9k: A challenging benchmark dataset of deep-research agent. *arXiv preprint arXiv:2603.01152*, 2026.

Renjun Xu and Jingwen Peng. A comprehensive survey of deep research: Systems, methodologies, and applications. *arXiv preprint arXiv:2506.12594*, 2025.

Le Xue, Manli Shu, Anas Awadalla, Jun Wang, An Yan, Senthil Purushwalkam, Honglu Zhou, Viraj Prabhu, Yutong Dai, Michael S Ryoo, et al. xgen-mm (blip-3): A family of open large multimodal models. *arXiv preprint arXiv:2408.08872*, 2024.

Huanjin Yao, Qixiang Yin, Min Yang, Ziwang Zhao, Yibo Wang, Haotian Luo, Jingyi Zhang, and Jiaxing Huang. Mm-deepresearch: A simple and effective multimodal agentic search baseline. *arXiv preprint arXiv:2603.01050*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Kaining Ying, Fanqing Meng, Jin Wang, Zhiqian Li, Han Lin, Yue Yang, Hao Zhang, Wenbo Zhang, Yuqi Lin, Shuo Liu, et al. Mmt-bench: A comprehensive multimodal benchmark for evaluating large vision-language models towards multitask agi. *arXiv preprint arXiv:2404.16006*, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, et al. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. *arXiv preprint arXiv:2410.10594*, 2024.

Yu Zeng, Wenxuan Huang, Zhen Fang, Shuang Chen, Yufan Shen, Yishuo Cai, Xiaoman Wang, Zhenfei Yin, Lin Chen, Zehui Chen, et al. Vision-deepresearch benchmark: Rethinking visual and textual search for multimodal large language models. *arXiv preprint arXiv:2602.02185*, 2026.

LLM Disclosure

We used large language models as tools throughout the data synthesis, verification, and evaluation pipeline. In particular, GPT-5-mini was used to filter seed VQA samples for image dependence; GPT-5 and GPT-5.1 were used to rewrite questions, generate candidate hop-level QA pairs, and disambiguate entity answers; and GPT-5 with web search was used as an independent verifier of factual consistency, answer uniqueness, and temporal stability. We also used an LLM-based judge to compute training rewards and evaluate rollout quality during reinforcement learning. In addition, GPT-5, Gemini-2.5-Pro, and Gemini-3-Pro were used as baseline systems in our experiments.

A Method Details

A.1 Definition of an Entity Answer

A core requirement of our dataset is that the *seed answer*—the answer to the first hop, which must be visually grounded—is a **named proper-noun entity**: a specific, uniquely identifiable real-world referent rather than a generic descriptor, number, or action. Concretely, we define an entity as any answer belonging to one of the following ten categories:

1. **People.** Famous individuals who are uniquely named, including actors, musicians, scientists, politicians, historical figures, and athletes (e.g., *Marie Curie*, *Lionel Messi*).
2. **Organizations.** Named institutions such as technology companies, NGOs, universities, sports teams, and government bodies (e.g., *UNICEF*, *Manchester United*).
3. **Locations.** Specific geographic or architectural referents, including cities, countries, landmarks, airports, and nature reserves (e.g., *Eiffel Tower*, *Yellowstone*).
4. **Products.** Commercially named goods such as consumer electronics, vehicles, and software (e.g., *iPhone 15*, *Tesla Model 3*).
5. **Events.** Named occurrences with a defined scope, including global sporting events, historical events, conferences, and cultural festivals (e.g., *FIFA World Cup*, *Diwali*).
6. **Creative Works.** Titled artistic or intellectual productions, including films, television series, books, artworks, and music albums (e.g., *Inception*, *Mona Lisa*).
7. **Science & Technology.** Named scientific entities such as biological species, chemical compounds, devices, AI models, and benchmark datasets (e.g., *GPT-4*, *ImageNet*).
8. **Medical & Health.** Named diseases or medical conditions (e.g., *COVID-19*, *Malaria*).
9. **Financial & Business.** Named companies, stock indexes, and commercial brands (e.g., *Coca-Cola*, *S&P 500*).
10. **Space & Astronomy.** Named celestial objects, space missions, and astronomical events (e.g., *Hubble Telescope*, *Mars*).

Answers that are numbers, adjectives, generic nouns (e.g., "a bridge", "the car"), actions, or multi-entity phrases are explicitly excluded. This constraint ensures that every seed answer is (i) factually verifiable through web search, (ii) a well-defined node in a real-world knowledge graph, and (iii) unambiguously linkable to follow-on questions in the multi-hop chain.

A.2 VQA Data Filtering Pipeline

We describe the filtering pipeline used to extract high-quality entity-grounded VQA pairs from raw datasets. Each candidate question-answer pair passes through five sequential stages; a pair is accepted only if it passes every stage.

Stage 1: Vision Requirement Check

We first verify that the question *genuinely requires* visual information to answer. Questions answerable from general knowledge alone (e.g., "What is the capital of France?") are discarded. This check is performed by GPT-5-mini with the following prompt:

Prompt: Vision Requirement Check

```
Analyze this question and determine if it requires visual information from an image
to answer:
Question: {question}
```

`Consider these scenarios that REQUIRE an image to provide a UNIQUE answer:`

- Questions about specific visual elements that make the answer unique ("What team do the standing players play for?")
- Questions about text, signs, names, or writing visible in the image ("What name can be found on the blackboard?")
- Questions about specific people, objects, or entities shown in the image ("Who is this person?", "What company logo is shown?")
- Questions about visual attributes that identify specific entities ("What color jersey is the team wearing?")
- Questions about spatial relationships that identify specific locations ("Where are these animals located?")
- Questions using demonstrative pronouns referring to specific visual elements ("this person", "that building", "these objects")
- Questions where the image provides the *only* way to determine the specific answer (not general knowledge)

`Consider these scenarios that DO NOT require an image:`

- General knowledge questions with standard answers ("What is the capital of France?")
- Questions about well-known facts that don't depend on visual context ("When was Einstein born?")
- Questions about definitions, explanations, or concepts that are universally known
- Questions where the answer can be determined from text alone
- Questions that would have the same answer regardless of what image is shown

`Please only return 'Yes' if the question REQUIRES an image to answer, 'No' if it can be answered without visual information.`

Stage 2: Q&A Rewriting and Free-Form Validity Check

Raw VQA answers often contain chain-of-thought reasoning traces, and questions may be phrased as multiple-choice. We use GPT-5 to simultaneously (i) rewrite the question into a clean free-form format that explicitly references "the image", (ii) extract the concise answer entity from any reasoning trace, (iii) produce a `complete_answer` with minimal disambiguating context (at most three words), and (iv) flag questions that are *inherently multiple-choice*—i.e., questions that cannot be rephrased as free-form without providing specific comparison options (e.g., "This person is playing a similar sport to whom?"). Such questions are rejected at this stage.

Prompt: Q&A Rewriting

Here is a question and answer pair where the answer contains reasoning:
Please rewrite this as a clean question-answer pair where:

1. The question should mention "in the image" and remove any answer choices.
2. Ensure the question makes sense as a free-form question without choices. Questions like "This person is playing a similar sport to whom?" require choices to be answerable and should be marked as invalid.
3. If the question cannot be reasonably converted to free-form without choices, respond with {''valid'': false, ''reason'': ''Question requires multiple choice options to be answerable''}.
4. The answer should be just the final entity/answer without any reasoning or explanation.
5. Provide both a short answer (output_answer) and a complete_answer:
   - Add descriptive context *from the provided context* to disambiguate it.
   - Keep the new description less than 3 words—add the minimum context needed for disambiguation.

Examples of disambiguation:

- "Apple" (technology company) → complete_answer: "Apple, a technology company"
- "Birmingham" (city in Alabama) → complete_answer: "Birmingham, city in Alabama"
- "Baltimore Orioles" (baseball team) → complete_answer: "Baltimore Orioles, a Baseball team"

Format your response as JSON:

```
// If valid:
{
  "valid": true,
  "output_question": "[clean question mentioning 'in the image']",
  "output_answer": "[direct short answer only]",
  "complete_answer": "[add context ONLY if ambiguous AND
                       ONLY using info from input_answer]"
}
// If invalid:
{
  "valid": false,
  "reason": "[explanation why it cannot be converted]"
}
```

Input:

```
{
  "input_question": "{question}",
  "input_answer": "{answer}"
}
```

Stage 3: Named Entity Classification

We verify that the rewritten answer is a *named proper noun entity* belonging to one of ten high-level categories (Table 5). This check is performed by GPT-5. Answers that are numbers, generic terms, actions, or vague phrases are rejected.

Category	Example entities
People	Barack Obama, Marie Curie, Lionel Messi
Organizations	Google, UNICEF, Manchester United
Locations	Tokyo, Eiffel Tower, Yellowstone
Products	iPhone 15, Tesla Model 3, Photoshop
Events	FIFA World Cup, Diwali, TED Conference
Creative Works	Inception, Mona Lisa, Harry Potter
Science & Technology	GPT-4, ImageNet, Bengal tiger
Medical & Health	COVID-19, Malaria
Financial & Business	Coca-Cola, S&P 500
Space & Astronomy	Mars, Hubble Telescope

Table 5: Entity categories used for filtering.

Prompt: Named Entity Classification

```
Here is a question and answer pair:
Question: {question}
Answer: {answer}

Check carefully if the answer is a named, proper noun entity from these specific
categories:
```
[Full category list as in Table 5, with subcategory examples]

```
Note: The reasoning process is not part of the answer! Answers with multiple
entities or non-entity answers should be returned as 'No'. Exclude: Numbers,
actions, generic entities, or vague terms.

Please only return 'Yes' or 'No'.
```

Stage 4: Entity Validity Verification

As a secondary guard, GPT-5-mini independently checks whether the extracted answer string constitutes a valid, specific proper noun (e.g., rejecting generic terms that may have slipped through Stage 3). The prompt mirrors the category taxonomy in Table 5 and again requests a binary `Yes/No` response.

Prompt: Entity Validity Verification

```
Is this phrase a valid, proper noun entity? {entity}.

I am looking for specific categories of entities:
```
[Full category list with subcategory examples, same as Stage 3]

```
Exclude: Numbers, actions, generic entities, or vague terms.

Please only return 'Yes' or 'No'.
```

Stage 5: Visual Answer Verification

Finally, GPT-5 with vision and web search is used to confirm that the proposed answer is factually correct given the image(s). This guards against errors introduced by the original dataset or the rewriting step. The model receives the image(s) via the Responses API and is permitted to issue web search queries to verify entity-specific facts. A pair is accepted only if the model responds with a definitive `Yes`; an `Unsure` response triggers rejection.

Prompt: Visual Answer Verification (GPT-5 with Vision + Web Search)

```
You are an expert at verifying answers to visual questions. You have extensive
knowledge about entities, landmarks, people, brands, and locations worldwide.

Question: {question}
Proposed Answer: {answer}

Please carefully analyze the image(s) and determine if the proposed answer is
correct. Use your extensive knowledge and web search to verify factual claims about
entities.

Important:
```

- The answer should be factually accurate based on what is visible in the image(s).
- Consider the answer correct if it is semantically equivalent (e.g., "NYC" and "New York City").
- Be strict about factual accuracy—if the answer is wrong, provide the correct answer.
- Use your knowledge of world facts, famous people, landmarks, brands, and locations to verify.
- Use web search when needed to verify specific facts.

`Respond with ONE of the following:`

- `"Yes"` if the answer is correct.
- `"Unsure"` if you cannot determine correctness with confidence.

Pipeline Summary

Table 6 summarizes the five stages, the model used at each stage, and the rejection criterion.

Table 6: Summary of the VQA filtering pipeline.

Stage	Check	Model	Reject if...
1	Vision requirement	GPT-5-mini	Question answerable without image
2	Q&A rewriting	GPT-5	Question cannot be free-form; conversion fails
3	Entity classification	GPT-5	Answer is not a named proper noun entity
4	Entity validity	GPT-5-mini	Answer is a generic or vague term
5	Visual verification	GPT-5 (vision + web)	Answer is factually incorrect or uncertain

A.3 Tools

The QA agent is equipped with four tools that cover complementary modalities of information retrieval:

1. **Web Search.** Given a text query, this tool issues a search request via the Tavily API and returns up to 10 results, each consisting of a URL, title, short snippet, and—when available—the full extracted page content in Markdown format. Queries are constrained to always include the complete disambiguated entity name to prevent irrelevant retrievals.
2. **Web Reader.** Given a URL, this tool fetches and returns the full textual content of the target webpage. It is used to deepen coverage of a specific source when the snippet returned by web search is insufficient for synthesizing a well-grounded question.
3. **Google Lens.** Given an image, this tool performs a reverse image search and returns visually similar results with their associated titles and snippets. It is particularly useful for identifying specific objects, landmarks, or entities that are visually distinctive but difficult to describe in text alone.
4. **Image Search.** Given a text query, this tool retrieves image URLs along with auto-generated descriptions and captions. It is used when visual context about the current entity—such as product appearance, architectural style, or logo design—can surface facts that text-only pages do not explicitly state.

Together, these tools allow the agent to gather evidence through both text and visual channels. At each hop, the agent selects the most appropriate tool and query based on the nature of the current entity and the type of bridging fact it seeks to uncover.

A.4 Context Chunking

Source Selection. GPT-5 selects up to three sources from the retrieved results by scoring each candidate on three criteria: (1) the content must mention a_{k-1}; (2) it should contain non-obvious facts unlikely to be known without retrieval; and (3) it must include proper-noun entities that could serve as the next answer a_k. Previously visited URLs are excluded to encourage diversity across hops.

Passage Selection. Each selected source is split into fixed-length windows. Each window is scored by a lightweight heuristic: it receives $+10$ per complete sentence and $+2$ per occurrence of common English function words (e.g., *the*, *and*, *is*), rewarding coherent natural-language prose over boilerplate or markup; windows containing HTML or JSON structures are penalized. Windows that contain a_{k-1} receive a large additive bonus to prioritize entity-relevant passages. The top-scoring windows are concatenated in reading order to form the context passed to the Q&A extraction step.

A.5 Q&A Candidate Generation

For each retrieved source, the system runs a four-step pipeline to produce a verified Q&A candidate (q, a, c), where q is the question, a is the answer entity, and c is the supporting context excerpt. One candidate is attempted per source; across up to ten sources per hop, this yields a pool of candidates that are later merged and validated.

A.5.1 Step 1: Content Summarization.

Raw webpage content is often long and only partially relevant. GPT-5 first produces a focused 200–400 word summary of the page chunk, emphasizing facts about the current entity that could serve as Q&A fodder—specific dates, locations, relationships, affiliated organizations, and other named entities. Generic well-known facts are explicitly excluded. The summarization prompt is:

Prompt: Content Summarization (GPT-5)

```
Summarize the key facts from this webpage content that are relevant to
``{entity_name}''.

Title: {title}
Search Query: {search_query}
Content: {raw_content}

Focus on:
```

- Facts about {entity_name} (dates, locations, relationships, achievements)
- Other entities mentioned (people, companies, places) that could be answer candidates
- Specific numbers, dates, or names (these make good answers)

```
Provide a concise summary (200--400 words) highlighting the most interesting and
specific facts. Do NOT include generic information that everyone knows.
```

A.5.2 Step 2: Q&A Extraction.

Given the summarized content, GPT-5.1 extracts a single factual question–answer pair. For all hops except the final one, the answer must be a named proper-noun entity; for the final hop, numeric and date answers are also permitted to widen coverage. Six hard constraints are enforced: (1) the question must contain the current entity name; (2) the answer must be unique—exactly one correct answer; (3) the answer must not duplicate any previous hop's answer; (4) removing the entity from the question must make the answer indeterminate; (5)

the question must be standalone, without references to "the passage" or "the context"; and (6) the question must ask about a meaningful relationship or attribute.

Prompt: Q&A Extraction (GPT-5.1, intermediate hops)

```
You will be given content about ``{entity_name}''. Find a simple factual question
that involves ``{entity_name}'' and has a specific entity as the answer.

The answer should be a proper noun entity from one of these categories:
{entity_categories}

Search Query Used: {search_query}
Content: {content}

AVOID DUPLICATE ANSWERS:
Previous answers already used: {previous_answers}
Your new answer MUST be different from ALL previous answers listed above.

CRITICAL REQUIREMENTS:
1. The question must have a UNIQUE (singular) answer — exactly ONE correct answer
2. The question must include "{entity_name}" in the question
3. The question must be standalone (no references to "the passage" or "the context")
4. Removing "{entity_name}" from the question should make it have multiple possible
   answers
5. The question should be USEFUL and NATURAL
6. Ask about meaningful relationships, key facts, or important attributes
7. Do not add any redundant details in the questions

The question must have ONLY ONE POSSIBLE ANSWER --- AVOID:
• Multiple possible answers (e.g., "What movies has X appeared in?")
• Time-dependent answers without time constraint (e.g., "Who is the current CEO?")
• Subjective answers (e.g., "What is the best...")
• Weak relational phrases ("associated with", "related to", "connected to")
• References to source material ("in the contexts", "mentioned in")

Return a JSON object:

{
  "question": "<question with {entity_name},
               ONLY ONE possible answer>",
  "answer":   "<specific entity name>",
  "contexts": "<exact text excerpt supporting this Q&A>"
}

If no suitable question can be created, return:
{"question": "", "answer": "", "contexts": ""}
```

A.5.3 Step 3: Question Simplification.

Extracted questions sometimes contain redundant temporal or locational qualifiers that do not contribute to making the answer unique (e.g., "Which company is the first company Elon Musk created in 1995?" where the year is unnecessary). GPT-5.1 checks for and removes such redundancies while guaranteeing that the simplified question retains the same unique answer and still contains the entity name.

Prompt: Question Simplification (GPT-5.1)

```
You are given a question that may have redundant or unnecessary details. Simplify
it while preserving the unique answer.

Original Question: {question}
Answer: {answer}
Entity in Question: {entity_name}

Remove unnecessary constraints/details that don't contribute to making the answer
unique.

Guidelines:
```

1. Keep the question grammatically correct and natural
2. The simplified question must still have the SAME unique answer
3. Remove redundant temporal info (years, dates) if not needed for uniqueness
4. Remove redundant location info if not needed for uniqueness
5. Keep the entity "{entity_name}" in the question

```
Do NOT simplify if all details are necessary for uniqueness, or if the question is
already concise.

Return:

{
  "needs_simplification": <true|false>,
  "reason": "<why simplification is/isn't needed>",
  "simplified_question": "<simplified question, or
                           original if no change needed>"
}
```

A.5.4 Step 4: Factual Verification.

The (simplified) question and answer are verified by GPT-5.1 against five criteria before being accepted as a candidate. A candidate is only accepted if it passes *all* five checks; otherwise it is discarded and the next source is tried.

Prompt: Q&A Verification (GPT-5.1)

```
Verify if this question-answer pair is factually correct and has a unique answer:

Question: {question}
Answer: {answer}
Entity: {entity_name}
Source URL: {url}
Context: {context}

Please verify:
```

1. Is the answer factually correct according to the given context?
2. Does the question contain the given entity "{entity_name}"?
3. Is this answer unique (not multiple possible answers)?
4. Is this answer stable over time (not changing)?
5. Removing "{entity_name}" from the question, is it impossible to get the unique answer?

```
Return ''VERIFIED'' if all criteria are met, ''REJECTED'' if any fail.
Explain your reasoning briefly.
```

Candidates that pass verification receive a *complete answer*—a short disambiguating phrase appended to the bare answer (e.g., *"Birmingham, city in Alabama"*)—generated by GPT-5.1 from the supporting context.

A.6 Candidate Selection via Difficulty Filtering

When multiple Q&A candidates are generated from different sources, we select the one that produces the most challenging merged question rather than the most semantically

diverse answer. For each candidate, we measure how easily a weak model could reconstruct the reasoning path. Concretely, Qwen3-VL-32B is prompted to produce the search query it would use to answer the merged question $\tilde{q}_k$. This weak-model query r_{weak} is then compared against the actual query r_{actual} used to retrieve the supporting evidence, using token-overlap Jaccard similarity after removing stop words:

$$\text{sim}(r_{\text{weak}}, r_{\text{actual}}) = \frac{|\mathcal{T}(r_{\text{weak}}) \cap \mathcal{T}(r_{\text{actual}})|}{|\mathcal{T}(r_{\text{weak}}) \cup \mathcal{T}(r_{\text{actual}})|}$$

where $\mathcal{T}(\cdot)$ denotes the set of content tokens after stop-word removal. A low similarity score indicates that the weak model would search differently from how the evidence was actually found, implying the question requires genuine multi-hop reasoning. We select the candidate with the lowest similarity score, subject to a threshold of 0.6; if all candidates exceed this threshold, the hop is rejected and a new search query is attempted. Answer diversity across hops is enforced separately at the prompt level: the Q&A synthesis prompt explicitly lists all previous answers $\{a_1, \ldots, a_{k-1}\}$ and instructs the model to generate an answer distinct from all of them.

A.7 Answer Disambiguation

Short entity answers (e.g., *Apple*) are often ambiguous and may cause the next-hop search to retrieve irrelevant results. After a candidate (q_k, a_k, c_k) passes verification, GPT-5.1 appends a short disambiguating phrase drawn from the supporting context c_k to produce the complete answer $\tilde{a}_k$. The disambiguated form is used exclusively as the seed for the search query at hop $k+1$; the original a_k is retained as the ground-truth answer label.

Prompt: Answer Disambiguation (GPT-5.1)

```
Given a question, answer, and supporting context, create a ``complete answer'' that
adds descriptive information to make the answer unique and unambiguous.

Context: {context}
Question: {question}
Short Answer: {answer}

RULES:
1. Add descriptive context from the provided context to disambiguate the answer.
2. Keep the added description to fewer than three words.

Examples:
• ``Apple'' → ``Apple, a technology company''
• ``Birmingham'' → ``Birmingham, city in Alabama''
• ``Baltimore Orioles'' → ``Baltimore Orioles, a Baseball team''
• ``google.com'' → ``google.com, a website''
• ``Comcast'' → ``Comcast, a company''

Return ONLY the complete answer, nothing else.
```

The disambiguating phrase is always grounded in c_k and is never hallucinated. The length constraint (fewer than three words) ensures the phrase remains concise enough to serve as an effective search seed without over-specifying the query.

A.8 Tool Response Cache Construction

To accelerate training and avoid redundant API calls during reinforcement learning rollouts, we build a static tool-response cache that maps structured lookup keys to pre-collected tool outputs. The cache covers all five tools used by the agent: **web search**, **web read**, **image search**, **reverse image search**, and **OCR**. Below we describe the two data sources, the key-construction scheme, and the quality filtering applied to each tool.

A.8.1 Data Sources

The cache is populated from two complementary sources.

Rollout JSONL files (training-time). During early training iterations, the agent's rollout records are saved as JSONL files. Each record contains a `tool_interact_info` list, where each element stores the raw action string—with tool-specific XML tags such as `<text_search_text>`, `<web_read>`, `<ocr_tool>`, `<text_search_image>`, and `<image_search_text>`—and the corresponding observation list. Extraction scripts parse these files, recover the tool-invocation parameters and raw responses, and emit a flat {`cache_key` $\rightarrow$ `response`} JSON dictionary.

Trajectory JSON files (inference-time). We also save full trajectories as per-question JSON files during training data creation process. Each trajectory encodes the question context and the sequence of action steps, each with both a raw `observation` and an LLM-generated `observation_summary`. A second set of extraction scripts parses these files and builds two parallel dictionaries: one mapping context-aware keys to observation summaries, and one mapping keys to raw observations.

A.8.2 Cache Key Construction

A central design choice is that cache keys are *context-sensitive*: the same tool invocation may legitimately return a differently framed response depending on the question being answered. Keys are therefore composite strings joined by a `||` delimiter, with all components lowercased for normalization.

Web Search and Image Search. The primary key encodes both the search query and the question:

$$\texttt{key} = \texttt{query} \parallel \texttt{question} \tag{1}$$

If no question context is available, the key degrades to `query` alone. For inference trajectories a secondary key `query||original` stores the raw observation independently of question context.

Web Read. The retrieved URL takes the role of the query:

$$\texttt{key} = \texttt{url} \parallel \texttt{question} \tag{2}$$

with fallback `url` when no question is present, and `url||original` for the raw observation.

Reverse Image Search. This tool accepts both an image URL and an optional text refinement query, yielding a three-part key:

$$\texttt{key} = \texttt{image_url} \parallel \texttt{query} \parallel \texttt{question} \tag{3}$$

Components are included only when non-empty, degrading gracefully to `image_url||query`, `image_url||question`, or `image_url` alone.

OCR. The image source (URL or local path) serves as the primary identifier:

$$\texttt{key} = \texttt{image_source} \parallel \texttt{question} \tag{4}$$

with fallback `image_source` alone for raw observation storage.

This two-level scheme—question-contextualized summaries and raw observations—lets the training environment serve either a concise, relevance-filtered response or the full API output depending on what the rollout infrastructure requests.

A.8.3 Response Extraction

From rollout files. Each `tool_interact_info` entry is identified by detecting the tool's XML tag in the `action` field. The query or URL is extracted from the tag's content (*e.g.*, `<text_search_text>query</text_search_text>`). For reverse image search, the tag content may encode both URL and query as `image_url||query`. The response is reconstructed by joining all non-empty strings in the `obs` list, stripping enclosing `<result>...</result>` tags, and splitting on the sentinel `"Response:"` to retain only the substantive output.

From trajectory files. Each trajectory step is inspected for the matching `action_type` (*e.g.*, `"web_search"`, `"ocr"`, `"reverse_image_search"`). The query or URL is read from `action_parameters`, and the question is resolved from the top-level `question` field, falling back to nested `trajectory.question` or the first step's question. The `observation` and `observation_summary` fields are then stored under their respective keys.

A.8.4 Quality Filtering

Before inserting any entry, responses are validated to exclude API failures and empty results. A response is rejected if it:

- is empty, `None`, or shorter than 10 characters;
- contains tool-specific error markers (*e.g.*, *"search failed"*, *"api error"*, *"execution failed"*, *"timeout"*, *"rate limit exceeded"*, *"quota exceeded"*);
- contains semantically empty results (*e.g.*, *"no search results found"*, *"no image results"*, *"no detailed information"*, *"no content extracted"*);
- begins with an error-class word (*"error"*, *"failed"*, *"exception"*, *"invalid"*, *"empty"*) within its first three tokens.

Tool-specific exceptions apply where needed. For OCR, a response of *"Text found in image: No text detected."* is considered valid since it faithfully represents an image containing no text. For web search and reverse image search, an occurrence of *"no results found"* embedded within a longer response (>50 characters) is not treated as a failure.

When multiple records share the same cache key, **the first valid response is retained** and subsequent duplicates are discarded, ensuring deterministic cache entries across training runs.

A.9 Output Format

Each extractor emits a JSON file. For rollout-sourced caches the format is a flat dictionary: [language=json, basicstyle=] `"query||question": "Response text ...", "query": "Response text ..."`

For inference-sourced caches the output contains two sub-dictionaries: [language=json, basicstyle=] "by$_{q}$*uery$_{q}$uestion"* : *"query||question"* : *"observation$_{s}$ummary..."*,*"by$_{q}$uery$_{o}$riginal"* : *"query||original"* : *"rawobservation..."* (with analogous structure for URL-based and image-URL-based tools).

At training time, the agent's tool server looks up incoming invocation parameters in the appropriate cache file. On a hit, the stored response is returned immediately without any external API call, eliminating the dominant source of latency and cost variability during policy-gradient rollouts.

B Experiment Details

B.1 Human Validation Protocol

To assess the quality of the generated multi-hop questions, we conduct a human annotation study on `MTA-Vision-DeepSearch` covering five seed VQA datasets: FVQA, InfoSeek, InfoVQA, News, and OK-VQA. Annotators evaluate each sample along two independent dimensions.

Annotation Interface. We prepare a structured Excel workbook with one tab per benchmark dataset. Each row corresponds to one multi-hop sample and contains the following columns, color-coded by hop group for readability:

- **id**: unique sample identifier.
- **question**: the final merged multi-hop question.
- **image url**: a clickable link to the source image (hosted on S3).
- **understand question?**: annotator response (True / False dropdown).
- For each hop $k = 1, \ldots, K$:
  - **hop** k **question**: the sub-question for this hop.
  - **hop** k **answer**: the expected answer for this hop.
  - **hop** k **url**: a clickable reference link (image URL for Hop 1; retrieved webpage URL for Hop $k \geq 2$).
  - **hop** k **correct?**: annotator response (True / False dropdown).

All response columns use in-cell True/False dropdowns to standardize input and minimise annotation error.

Task 1: Image Necessity Check. Prior to the comprehension check, annotators assess whether the final multi-hop question *genuinely requires* the image to answer. A question passes this check if the image provides the only means to determine the specific answer—for instance, by identifying a person, landmark, logo, or object that would be otherwise unspecified. Questions answerable from general knowledge alone, without reference to the image, are flagged.

Task 2: Overall Question Comprehension. Annotators first read the final multi-hop question and open the image URL. They assess whether the question is grammatically sound, logically coherent, and unambiguous given the image. A question is marked TRUE if its intent is clearly understandable—even if the phrasing is not perfectly natural—and FALSE if it is confusing, contradictory, or uninterpretable without additional context.

Task 3: Per-Hop Factual Accuracy. For each hop, annotators verify the correctness of the (sub-question, answer) pair against the provided reference. Specifically:

- **Hop 1** is answered from the image; annotators open the image URL and check whether the stated answer accurately describes the visual content.
- **Hops** $k \geq 2$ are answered from retrieved webpages; annotators open the hop URL and verify whether the answer is factually supported by the page content.

A hop is marked TRUE if the sub-question and answer are factually correct, supported by the reference, and logically consistent with the preceding hop. It is marked FALSE if the answer is incorrect, hallucinated, or breaks the reasoning chain.

B.2 Search Agent Implementation Details

B.2.1 Agent Architecture Overview

Our multimodal search agent follows the **ReAct** (Reasoning and Acting) paradigm, interleaving natural-language reasoning steps with grounded tool executions to iteratively

gather information before producing a final answer. The agent is implemented as an abstract base class (`MultimodalDeepResearchTesterBase`) that supports different interchangeable inference backends such as OpenAI GPT-5, vLLM (for locally-served open-weight models), and Google Gemini. All backends share the same tool interface, prompt templates, and loop logic.

B.2.2 Tool Registry

The agent exposes a fixed registry of seven tools. Each tool is defined with a natural-language description, typed parameters, an XML-style invocation format, and a usage example. The active tool set is configurable per run; unused tools are omitted from the system prompt.

Tool Name	XML Tag	Purpose
`web_text_to_text_search`	`<text_search_text>`	Text-to-text web search; returns snippets and URLs
`web_text_to_img_search`	`<text_search_image>`	Text-to-image web search; returns image descriptions
`web_url_reader`	`<web_read>`	Full-page content extraction from a URL
`web_image_to_text`	`<image_search_text>`	Google Lens reverse-image search; identifies objects
`ocr_tool`	`<ocr_tool>`	OCR extraction of all visible text from an image
`ipython_code`	`<python>`	IPython code execution with persistent state

Table 7: Tool registry available to the multimodal search agent.

B.2.3 ReAct Loop

The agent runs up to **6 iterations** of the Reason–Act–Observe cycle. The cycle is implemented via a stateful chat message list that grows with each step.

Step 1 – Initialization. The chat history is seeded with a system prompt that includes the tool registry descriptions and stopping criteria. On the first iteration, the research question and the associated image (base64-encoded inline) are included in the first user message.

Step 2 – Reasoning. The model receives the full chat history and produces a structured JSON response:

```
{
  "reasoning": "...",
  "action": {
    "action_type": "web_search",
    "action_description": "...",
    "action_parameters": {"query": "..."},
    "expected_outcome": "..."
  },
  "should_stop": false,
  "confidence": 0.8
}
```

The JSON is parsed by scanning for the first balanced $\{\ldots\}$ substring, falling back to a default web-search action on parse failure.

Step 3 – Reflection. Before executing the action, a reflection sub-step can be enabled. The model is prompted to critique the proposed action—evaluating query quality, redundancy with prior steps, and tool choice—and may modify the `action_parameters` or switch to a different tool.

Step 4 – Action Execution. The action JSON is translated into an XML-tagged string (e.g., `<text_search_text>query</text_search_text>`) and dispatched as an HTTP `POST` request to

a tool server at `/get_observation`. The server returns a JSON observation. URL placeholders (`[IMAGE]`) in action parameters are resolved to the actual image URL at dispatch time.

Step 5 – Observation. The raw server response is either used directly or compressed by a GPT-5 summarizer (one prompt per tool type) and appended to the chat history as a user message.

Step 6 – Stopping. The loop exits when: (a) the model sets `should_stop: true` with `confidence` > 0.7; (b) two consecutive tool calls fail; (c) the estimated context length exceeds the configured token limit (the last ReAct cycle is removed and the agent proceeds directly to the final answer); or (d) the maximum of 6 iterations is reached.

B.2.4 Context Management

Token length is estimated as $\lfloor \text{total characters}/4 \rfloor$ for GPT and Gemini backends, and by the vLLM tokenizer for local models. Image content is approximated at 256 tokens per image. When the estimate exceeds `max_context_tokens` (default 128k), the most recent full ReAct cycle—observation turn, reasoning turn, and reflection messages if enabled—is removed from the chat history before the final answer step.

B.2.5 Final Answer Generation

After the loop terminates, the model is prompted to synthesize its findings and produce a concise answer enclosed in a LaTeX box:

> *Based on the research findings provided, provide a direct answer to the research question. . . . Put your final concise answer inside* `\boxed{}`.

The boxed expression is subsequently extracted for automatic evaluation.

B.3 LLM-as-Judge Evaluation

We evaluate agent trajectories using `check_answer_correctness`, which calls GPT-5 as a judge to perform a binary correctness check after each trajectory completes. The judge receives the following prompt:

Prompt: Answer Correctness Check (GPT-5)

System: You are an expert answer evaluator. Respond only with valid JSON.

User: You are an expert answer evaluator. Compare the model's answer to the ground truth and determine if they are semantically equivalent.

Question: {question}
Ground Truth Answer: {ground_truth}
Model's Answer: {final_answer}

Instructions:

- The model's answer is correct if it conveys the same core information as the ground truth, even if worded differently.
- Minor differences in phrasing, formatting, or additional context are acceptable if the key answer is correct.
- Look for the actual answer, which might be inside `\boxed{}`.
- Additionally, check if the ground truth answer appears anywhere in the model's reasoning/explanation outside of `\boxed{}`. This indicates correct reasoning even if the final answer format is different.

Respond with a JSON object:
`{"is_correct": true/false, "is_correct_reasoning": true/false, "reason": "brief explanation"}`

Where:

- `"is_correct"`: true if the answer (especially in `\boxed{}`) matches the ground truth.
- `"is_correct_reasoning"`: true if the ground truth can be found in the reasoning/explanation outside of `\boxed{}`, even if `"is_correct"` is false.

The judge returns two boolean fields:

- **`is_correct`**: `true` if the content enclosed in `\boxed{}` is semantically equivalent to the ground truth, allowing for minor differences in phrasing or formatting.
- **`is_correct_reasoning`**: `true` if the ground-truth answer appears anywhere in the model's reasoning or explanation *outside* of `\boxed{}`, even when `is_correct` is `false`. This distinguishes cases where the model arrived at the correct answer through valid reasoning but failed to format it correctly in the final box.

C More Study

C.1 Ablation of Data generation pipelines

We iterate the desgin of data generation pipelines multiple times, each time, our 4 authors label 100 generation QA and update the prompts and design.

We first identify an issue called **image redundancy**. During merging, specific entities may make a question answerable without the image. We find that 14% of samples suffer from this issue. For example, consider the merged question: "What organization was established to coordinate the policies and operations of the United States representatives on the Fund and the Bank in the country where this shop's official currency was first recognized as a currency of the world?" Here, the image only depicts the United States, so the question can be answered without it. To address this, we add a validation step: GPT-5 is asked to answer the question *without* the image. If the answer matches the one obtained with the image, the image is considered redundant. After applying this step, the redundancy rate drops to 6%.

We also observe that 15% of samples suffer from **inconsistent entities**. Even when the entity name is the same, the actual entity may differ. For example, "Newark" in one answer refers to Newark, CA, while in the next question it refers to Newark, NJ. To address this, we add descriptive context to each entity, which is then used in subsequent question generation. For

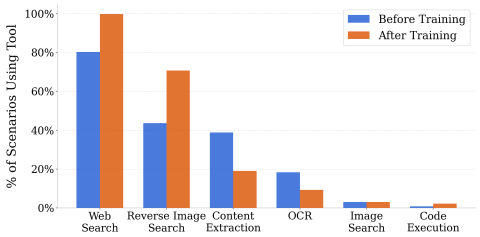

Figure 4: Tool usage distribution on BrowseComp-VL.

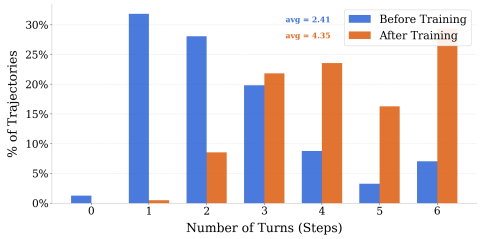

Figure 5: Turn distribution on BrowseComp-VL.

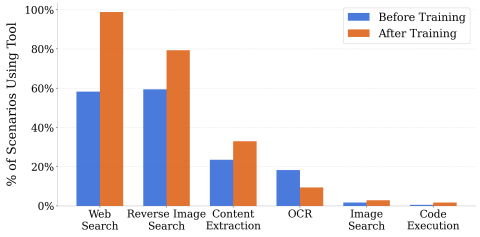

Figure 6: Tool usage distribution on MMSearch-VL.

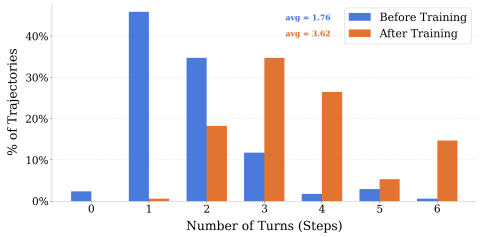

Figure 7: Turn distribution on MMSearch-VL.

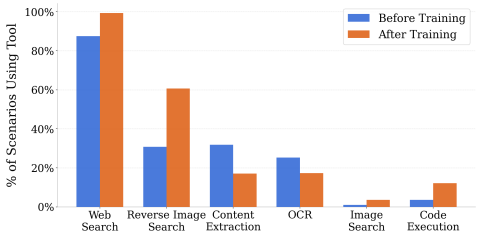

Figure 8: Tool usage distribution on HR-MMSearch-VL.

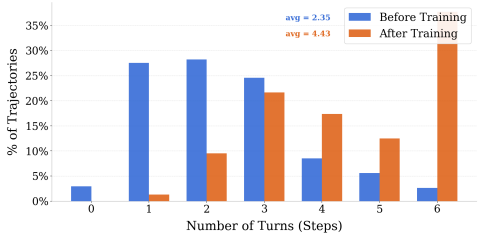

Figure 9: Turn distribution on HR-MMSearch-VL.

instance, "Newark, a city in CA, US" ensures consistent reference across questions. After applying this, the inconsistency rate drops to 3%. In addition, we find that 12% of questions suffer from **contextual dependency**: these questions are not standalone, often referring to "provided content," "context," or "mentions" within the task rather than forming natural, independent questions. We address this by adding prompts during question generation and validation. After this step, the rate of contextual dependency drops to 2%.

C.2 Dataset-Specific Tool Use and Search Depth Analysis

We present per-dataset breakdowns of tool usage and turn distributions for all five benchmarks evaluated in this work. For each benchmark, we compare three settings: the base model before RL training (Before Training), the model after RL training (After Training), and GPT-5 as an oracle reference. Tool usage is measured as the percentage of scenarios in which each tool is invoked at least once. Turn distribution shows the percentage of trajectories terminating at each number of search steps.

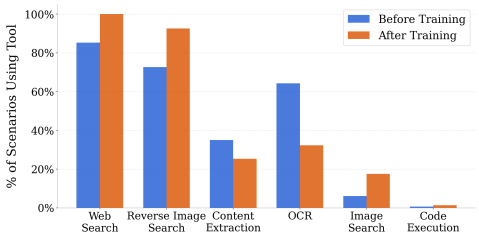

Figure 10: Tool usage distribution on MMSearch-Plus.

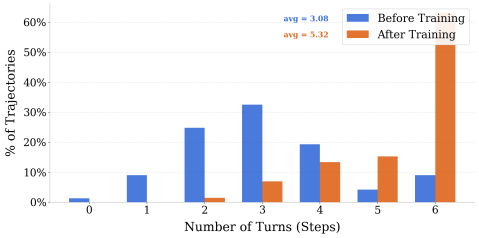

Figure 11: Turn distribution on MMSearch-Plus.

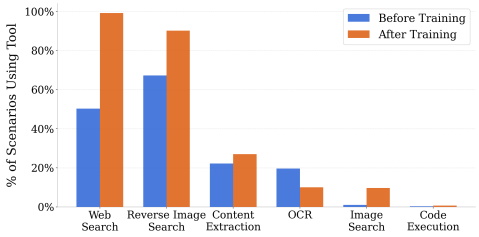

Figure 12: Tool usage distribution on FVQA.

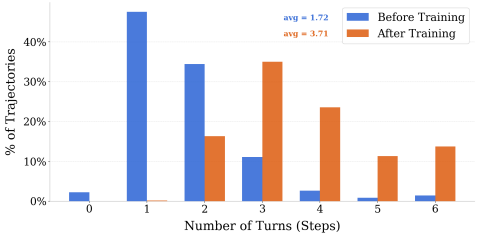

Figure 13: Turn distribution on FVQA.

D Examples

Example 1

Image: See Figure 14.

Individual Hops:

1. **Q (Image):** What is the brand of the vehicle in the image?
   A: Ferrari
2. **Q:** Who founded Ferrari?
   A: Enzo Ferrari
3. **Q:** Which flying ace's prancing horse emblem did Enzo Ferrari adopt for his cars?
   A: Francesco Baracca
4. **Q:** Who was the sculptor of the monument dedicated to Francesco Baracca?
   A: Domenico Rambelli
5. **Q:** In which city is Domenico Rambelli's monument to Francesco Baracca located?
   A: Lugo

Merged Questions (step-by-step composition):

1. *After Hop 1→2:* Who founded the brand of the vehicle in the image?
2. *After Hop 2→3:* Which flying ace's prancing horse emblem was adopted for the cars of the brand of the vehicle in the image?
3. *After Hop 3→4:* Which sculptor created the monument dedicated to the flying ace whose prancing horse emblem was later adopted for the cars of the brand of the vehicle in the image?
4. *After Hop 4→5 (Final):* In which city is the monument located that honors the flying ace whose prancing horse emblem was later adopted for the cars of the brand of the vehicle in the image?

Final Answer: *Lugo*

Figure 14: Image for Example 1.

Example 2

Image: See Figure 15.

Individual Hops:

1. **Q (Image):** What organization is represented by the emblem in the image?
   A: Beşiktaş J.K.
2. **Q:** In which city is Beşiktaş J.K.'s home stadium located?
   A: Istanbul
3. **Q:** What is the name of the underground metro line in Istanbul that is considered the world's second-oldest?
   A: Istanbul Tünel
4. **Q:** Which sultan granted the concession to build the Istanbul Tünel?
   A: Sultan Abdülaziz
5. **Q:** Which British monarch invested Sultan Abdülaziz with the Order of the Garter?
   A: Queen Victoria

Merged Questions (step-by-step composition):

1. *After Hop 1→2:* In which city is the home stadium of the organization represented by the emblem in the image located?
2. *After Hop 2→3:* What is the name of the underground metro line in the city where the home stadium of the organization represented by the emblem in the image is located, which is considered the world's second-oldest?
3. *After Hop 3→4:* Which sultan granted the concession to build the underground metro line in the city where the home stadium of the organization represented by the emblem in the image is located, which is considered the world's second-oldest?
4. *After Hop 4→5 (Final):* Which British monarch invested with the Order of the Garter the sultan who granted the concession to build the underground metro line in the city where the home stadium of the organization represented by the emblem in the image is located, which is considered the world's second-oldest?

Final Answer: *Queen Victoria*

Figure 15: Image for Example 2.

Example 3

Image: See Figure 16.

Individual Hops:

1. **Q (Image):** Which book in the image predicted the fall of Detroit into poverty?
   A: *Stand on Zanzibar*
2. **Q:** Who wrote *Stand on Zanzibar*?
   A: John Brunner
3. **Q:** In which Scottish city did John Brunner die?
   A: Glasgow
4. **Q:** Which saint is associated with giving Glasgow its name meaning 'dear green place'?
   A: Saint Mungo
5. **Q:** In which Scottish town was Saint Mungo born?
   A: Culross

Merged Questions (step-by-step composition):

1. *After Hop 1→2:* Who wrote the book in the image that predicted the fall of Detroit into poverty?
2. *After Hop 2→3:* In which Scottish city did the author of the book in the image that predicted the fall of Detroit into poverty die?
3. *After Hop 3→4:* Which saint is associated with giving the name meaning 'dear green place' to the Scottish city where the author of the book in the image that predicted the fall of Detroit into poverty died?
4. *After Hop 4→5 (Final):* In which Scottish town was the saint born who is associated with giving the name meaning 'dear green place' to the Scottish city where the author of the book in the image that predicted the fall of Detroit into poverty died?

Final Answer: *Culross*

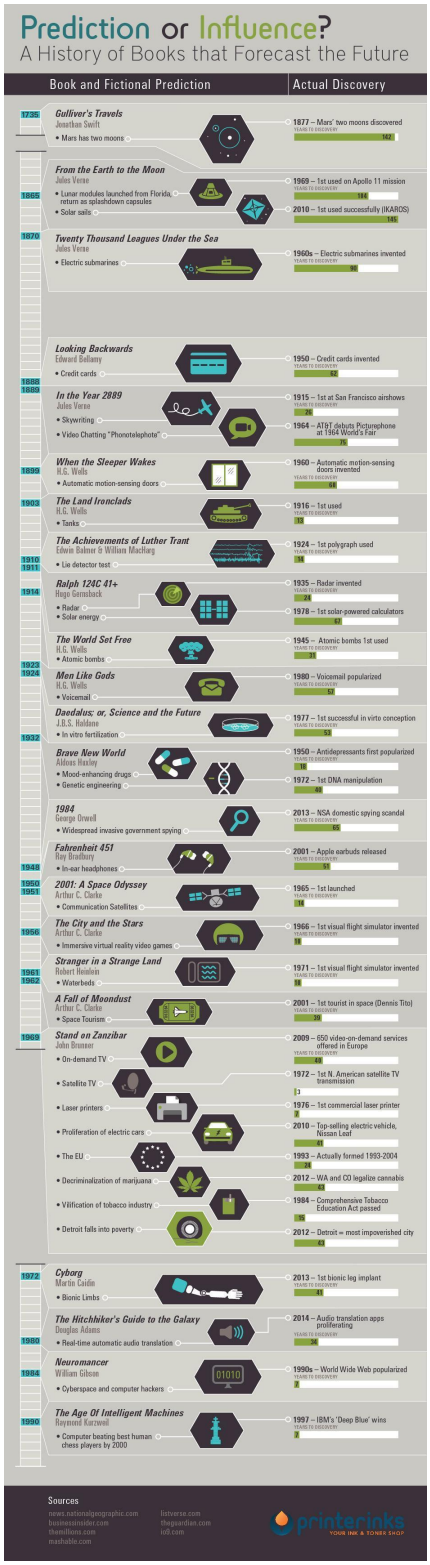

Figure 16: Image for Example 3.

Example 4

Image: See Figure 17.

Individual Hops:

1. **Q (Image):** What holiday is the boy in the image likely celebrating?
   A: Halloween
2. **Q:** Which city is known as the Halloween Capital of the World?
   A: Anoka, Minnesota
3. **Q:** From which lake does the Rum River in Anoka, Minnesota flow out?
   A: Mille Lacs Lake
4. **Q:** Which U.S. president issued the executive order that first placed Spirit Island in Mille Lacs Lake under federal protection?
   A: Woodrow Wilson
5. **Q:** On which transport ship did Woodrow Wilson travel to the peace negotiations after World War I?
   A: USS George Washington

Merged Questions (step-by-step composition):

1. *After Hop 1→2:* Which city is known as the capital of the holiday the boy in the image is likely celebrating?
2. *After Hop 2→3:* From which lake does the river flowing out of the city known as the capital of the holiday the boy in the image is likely celebrating originate?
3. *After Hop 3→4:* Which U.S. president issued the executive order that first placed Spirit Island in the lake from which the river flowing out of the city known as the capital of the holiday the boy in the image is likely celebrating originates, under federal protection?
4. *After Hop 4→5 (Final):* On which transport ship did the U.S. president who first placed Spirit Island in the lake from which the river flowing out of the city known as the capital of the holiday the boy in the image is likely celebrating originates, travel to the peace negotiations after World War I?

Final Answer: *USS George Washington*

Figure 17: Image for Example 4.