LESSONS FROM AGENTIC POST-TRAINING OF MLLMS FOR MULTIMODAL WEB SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal web search requires models to formulate effective queries for knowledge-intensive visual questions, gather relevant evidence from the web, and synthesize grounded answers. We systematically study three aspects of post-training multimodal search agents: evaluation, supervised fine-tuning, and reinforcement learning. First, we quantify the impact of individual evaluation components on measured performance and advocate for a standardized evaluation protocol to enable reliable comparisons across methods. Next, we examine how the distribution of tool calls in the SFT training corpus shapes the student agent's search behavior. We further show empirically that short interaction horizons reduce computational costs while maintaining strong performance, whereas longer search horizons do not consistently improve accuracy. Finally, we investigate branching RL to refine intermediate search decisions using only terminal rewards. This approach generates alternative continuations from shared intermediate states and compares their terminal rewards, enabling more targeted credit assignment to individual search decisions. Our empirical analysis identifies early branching as an effective data-driven heuristic: concentrating branching on a small number of early search decisions yields more useful exploration than random branching and outperforms strategies guided by the model's own uncertainty, measured through token entropy or self-reported confidence. Extensive experiments show that the resulting RL policy improves average accuracy from 59.89% to 63.02% over its SFT initialization and achieves state-of-the-art performance on multiple benchmarks. Together, these findings provide practical guidance for evaluating and post-training multimodal search agents.

1 INTRODUCTION

Multimodal large language models (MLLMs) increasingly interact with external tools to answer questions beyond their parametric knowledge. Multimodal web search is an important setting: given an image and a question, an agent must identify relevant visual entities, formulate queries, retrieve evidence, and synthesize a grounded answer (Geng et al., 2026; Tao et al., 2026; Jiang et al., 2024; Wu et al., 2025). Recent efforts have advanced these capabilities through improved training data, tool environments, and post-training methods (Wu et al., 2025; Huang et al., 2026b; Yao et al., 2026; Peng et al., 2026; Huang et al., 2026a). A common recipe combines supervised fine-tuning (SFT) on teacher-generated search trajectories with reinforcement learning (RL) based on final-answer rewards. Despite this progress, designing an effective post-training pipeline requires understanding how its individual components influence the resulting search policy.

A multimodal search agent operates within a system that includes the interaction protocol, retrieval tools, webpage summarizer, and answer judge. These components determine both the evidence available to the agent and how its answers are assessed. However, existing studies differ in their evaluation configurations, making it difficult to attribute reported improvements to the search policy itself. Interaction budgets introduce another source of variation: allowing more search turns provides opportunities to gather evidence, but also increases computational cost and exposes the agent to further retrieval noise. These considerations raise a fundamental question: *How can we evaluate improvements in multimodal search while disentangling the effects of the policy, its surrounding evaluation system, and its interaction budget?*

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

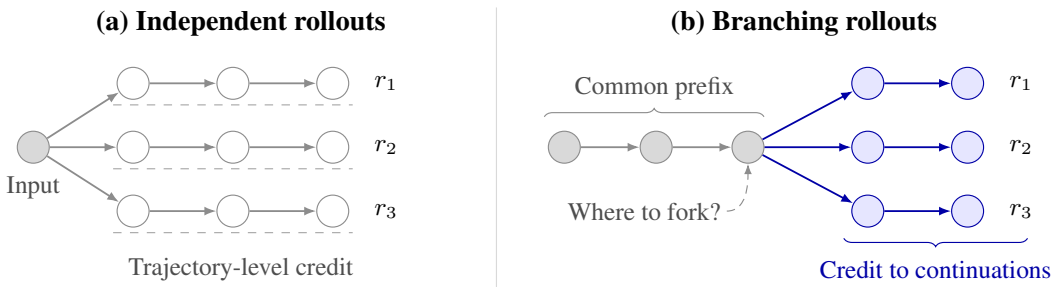


Figure 1: ***(a)*** independent rollouts receive trajectory-level credit. ***(b)*** a common prefix enables more targeted credit assignment to post-fork continuations.

The role of SFT also warrants closer examination. Teacher-generated trajectories provide demonstrations of how to reason, invoke tools, and answer questions, but they additionally encode particular choices about which tools to use and how long to continue searching. Consequently, the value of a trajectory corpus may depend on both the behavior it demonstrates and the capabilities already present in the student. Moreover, a correct final answer does not establish that every preceding search was necessary or that the demonstrated sequence was preferable to alternative searches. This motivates a second question: *What search behavior does SFT transfer, and how do the student initialization and the training-trajectory horizon influence its effectiveness?*

Outcome-based RL offers a way to refine the initial policy through exploration, but learning from final answers presents challenges. In commonly used group-relative training, independently sampled trajectories receive terminal rewards that supervise the reasoning and actions throughout each trajectory. Such feedback does not directly distinguish a useful search decision from an unproductive one within the same successful rollout. Furthermore, groups whose trajectories all receive the same reward provide no relative outcome signal, despite the cost of generating them. Comparing alternative continuations from a shared interaction prefix offers a way to make these comparisons more targeted without requiring intermediate annotations as shown in Figure 1. However, each continuation incurs generation and tool-use costs, making the placement of exploration an important design choice. This leads to a third question: *How can rollout construction provide more informative supervision for intermediate search decisions using only terminal rewards, and where should the exploration budget be allocated?*

In this paper, we conduct a systematic empirical study of multimodal agentic post-training organized around these questions. Our objective is to establish practical guidance through controlled comparisons that separate the effects of evaluation choices, training data, interaction budgets, and RL rollout design. We study these factors within a common agent framework and use the resulting analysis to design a complete post-training recipe. Finally, we evaluate the resulting post-training recipe across seven multimodal search benchmarks spanning visual factuality and complex browsing. Our investigation proceeds in four stages. ❶ We examine the sensitivity of measured performance to the evaluation setup by varying the summarization model, judge model, and judging prompt while controlling the remaining components. ❷ We analyze SFT across student initializations and training stages, tracking both answer accuracy and the tool-use behavior transferred from the training corpus. ❸ We investigate interaction budgets through complementary comparisons of training-trajectory horizons and inference-time turn limits. ❹ We study branching rollouts for RL, comparing alternative continuations from shared intermediate prefixes histories and examining how different fork-selection strategies affect the availability of informative reward comparisons.

2 Related Work

MLLMs with agentic search capability. Search-R1 (Jin et al., 2025) and R1-Searcher (Song et al., 2025) established outcome-based RL recipes for teaching LLMs to interleave reasoning with autonomous text search. Extending this paradigm to multimodal information seeking, MMSearch-R1 (Wu et al., 2025) trains on-demand image and text search with end-to-end RL, while DeepMMSearch-R1 (Narayan et al., 2025) establishes the crop-and-search paradigm, in which the agent first crops the relevant visual entity and then uses it to perform a targeted image

search. DeepEyesV2 (Hong et al., 2026) develops a general agentic multimodal model through cold-start training followed by RL. WebWatcher (Geng et al., 2026) introduces synthetic multimodal cold-start trajectories and BrowseComp-VL, a benchmark for complex visual-textual retrieval. SenseNova-MARS (Chng et al., 2025) proposes BN-GSPO for stable interleaved tool-use RL and HR-MMSearch for high-resolution, search-intensive evaluation. MM-DeepResearch (Yao et al., 2026) contributes Hyper-Search framework for cross-modal question generation and an offline multi-tool search engine for lower-cost RL. Vision-DeepResearch (Huang et al., 2026b) introduces a long-horizon paradigm that performs multi-turn, multi-entity, and multi-scale search under noisy real-world retrieval. OpenSearch-VL (Chen et al., 2026) provides a fully open data and training recipe, including a broad visual-tool environment and fatal-aware GRPO for cascading tool failures. MTA-Agent (Peng et al., 2026) automates verified tool-augmented synthesis of multi-hop multimodal QA trajectories, with cached interaction replay for efficient training Finally, ODE (Huang et al., 2026a) makes intermediate images reusable through an image-bank protocol and evolves SFT and RL data from the current policy's rollouts.

Agentic Search Benchmarks. Benchmarks for multimodal web-search span factual VQA and long-horizon web browsing. FVQA (Wang et al., 2017) introduced questions that require external knowledge and provides a supporting fact for each answer, while SimpleVQA (Cheng et al., 2025) evaluates broad, static visual factuality with short, easily judged answers. LiveVQA (Fu et al., 2025) instead targets time-sensitive visual knowledge through single- and multi-hop questions derived from recent online news. MMSearch (Jiang et al., 2024) directly evaluates MLLMs as multimodal search engines over image and text retrieval. More recent benchmarks emphasize harder agentic browsing: MM-BrowseComp (Li et al., 2025) requires multimodal retrieval and reasoning over images or videos encountered during web browsing, with verified checklists for analyzing each solution path; BrowseComp-VL (Geng et al., 2026) evaluates complex visual-textual deep research; and VDR-Bench (Huang et al., 2026b) focuses on iterative entity-level localization, visual search, and cross-modal multi-hop reasoning. Finally, MMSearch+ (Tao et al., 2026) evaluates provenance-aware browsing by requiring agents to propagate fine-grained visual cues through iterative image-text retrieval and verify evidence under search noise.

3 METHODOLOGY

We consider a multimodal search policy π_θ that receives a question x, an associated image v, and the interaction history at each turn. At turn t, the policy generates an assistant action a_t, which may contain reasoning and either a tool call or a final answer. A tool call returns an observation o_t, and the resulting history is

$$h_t = (x, v, a_1, o_1, \ldots, a_{t-1}, o_{t-1}), \qquad a_t \sim \pi_\theta(\cdot \mid h_t). \tag{1}$$

The policy may use up to T tool-enabled turns. It terminates when it emits a final answer; if it has not answered within this budget, an additional answer-only turn is required. Each completed trajectory $\tau = (x, v, a_1, o_1, \ldots, a_L)$ receives a terminal outcome reward $r(\tau)$ from the answer judge. Our objective is to improve the intermediate search decisions using only this terminal reward, without step-level labels or a learned process-reward model.

3.1 AGENT FRAMEWORK

We follow the REACT Yao et al. (2022) paradigm, in which the policy interleaves reasoning, tool use, and answer generation over multiple interaction turns. Our interaction environment constrains the available action space according to the stage of the search process. The interaction protocol is:

$$h_1 = (x, v) \xrightarrow[\mathcal{A}_{\text{img}}]{\text{mandatory image search}} h_2 \xrightarrow[\mathcal{A}_{\text{web}} \cup \{\text{ANSWER}\}]{\text{optional research, } t=2:T} h_L \xrightarrow[\text{no tools}]{\text{forced answer if needed}} \tau, \qquad L \le T+1 \tag{2}$$

Here, $\mathcal{A}_{\text{img}}$ and $\mathcal{A}_{\text{web}}$ denote the visual-search and web-search action sets, respectively. At $t = 1$, the policy is restricted to $\mathcal{A}_{\text{img}}$, which includes *image_search* and *crop_and_search*, thereby enforcing an initial visual retrieval step. For $t = 2, \ldots, T$, the available action space is $\mathcal{A}_{\text{web}} \cup \{\text{ANSWER}\}$, where $\mathcal{A}_{\text{web}}$ contains *text_search* and *web_visit*. The policy may either continue searching or terminate by emitting a final answer.

If the policy does not terminate within the T tool-enabled turns, the interaction is extended by one answer-only turn with tool access disabled. Hence, the trajectory length satisfies $L \leq T + 1$. At each turn, the environment exposes only actions permitted by the interaction protocol. To control the model's context length, each retrieved web page is condensed by a fixed summarization model before its contents are appended to the interaction history. Together, these choices define a common interaction interface across policies while allowing the tool-use budget T to be varied independently.

3.2 Effect of evaluation setup

We evaluate on seven multimodal search benchmarks: FVQA (Wang et al., 2017), SimpleVQA (Cheng et al., 2025), LiveVQA (Fu et al., 2025), MMSearch (Jiang et al., 2024), MMSearch+ (Tao et al., 2026), VDR-Bench (Huang et al., 2026b), and BrowseComp-VL (Geng et al., 2026). Benchmark details are provided in the Appendix. Unless explicitly varied, we use the common agent framework in Section 3.1 with a fixed interaction budget, Serper for image and text search, Jina.ai for webpage retrieval, *claude-sonnet-5* for webpage summarization, and *gpt-5-mini* at temperature 0 with the *recall* prompt for answer judging.

Table 1: *Sensitivity of agentic-search performance to the evaluation stack.* For each policy and benchmark, we report the largest score change from varying one evaluation component while holding the others fixed. Red and blue denote negative and positive changes, respectively, while darker purple indicates a larger mean absolute change.

Component toggle	**BC-VL**	**MMS+**	**FVQA**	**Mean $\lvert\Delta\rvert$**
Base policy				
Judge prompt: recall → semantic	−5.52	−1.81	−3.67	**3.67**
Judge model: gpt-5-mini → qwen-235b	−3.01	+1.36	−2.00	**2.12**
Summarizer: qwen-235b → sonnet-5	+3.51	+1.81	+6.00	**3.77**
SFT policy				
Judge prompt: recall → semantic	−5.26	−2.72	−4.67	**4.22**
Judge model: gpt-5-mini → qwen-235b	−3.26	+1.81	+2.34	**2.47**
Summarizer: qwen-235b → sonnet-5	+1.51	+1.81	+4.00	**2.44**

To separate policy improvements from evaluation effects, we vary the summarizer, judge model, and judge prompt individually while holding the remaining components fixed. The stricter *semantic* prompt generally lowers accuracy, by up to 5.52 percentage points, while changing the judge model has a smaller, mixed effect. Changing the summarizer shifts accuracy by up to 6.00 points: unlike the judge, which assesses completed predictions, the summarizer determines which retrieved evidence reaches the policy and can alter subsequent search decisions. Thus, evaluation choices alone can produce differences comparable to reported policy gains.

Table 2: VDR-30B-A3B: published and common-harness accuracy (%) at native (50-turn) and matched (5-turn) budgets.

Evaluation setting	MMSearch+	FVQA	BC-VL
VDR-30B-A3B (published)	28.50	74.20	53.70
VDR-30B-A3B (our harness, $T = 50$)	22.17	68.33	55.64
VDR-30B-A3B (our harness, $T = 5$)	16.74	71.00	53.60
Ours (our harness, $T = 5$)	**33.94**	**80.00**	**59.89**

Table 2 illustrates these differences for VDR-30B-A3B Huang et al. (2026b). Even at its native interaction budget, its accuracy in our harness differs from the published results by up to 6.33 points, reflecting differences in evaluation conditions. Within our harness, increasing the budget from 5 to 50 turns improves MMSearch+ and BrowseComp-VL but decreases FVQA accuracy. Our policy outperforms VDR under both budgets, including by 11.77 points on MMSearch+ at VDR's native budget. These results distinguish published system-level performance from policy comparisons under a shared harness and explicit resource limits. Our headline results use *claude-sonnet-5* and *gpt-5-mini* to remain comparable with prior work under the same evaluation convention. For our SFT model, we additionally report results on a subset of benchmarks using the open-source *Qwen3-235B-A22B* summarizer and judge to provide a reproducible reference. For fair and reproducible evaluation, we encourage the

community to adopt a shared harness with fixed, publicly available open-source models for summarization and judging, together with released prompts and configurations. Comparisons should report common-harness results at both matched and native interaction budgets, as well as token usage, tool calls, and inference cost, since equal turn limits do not necessarily imply equal compute.

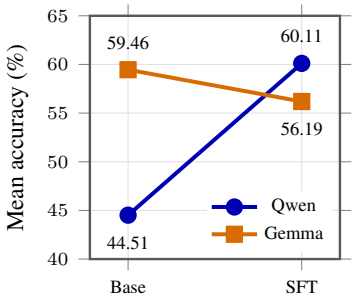
Lesson 1: Evaluation differences obscure policy gains and measured progress.
Evaluation choices affect accuracy. To isolate policy gains, compare methods in a shared harness under matched budgets, and report native-budget results separately. Fixed, publicly available open-source summarizers and judges can make these comparisons more reproducible.

3.3 Supervised Fine-Tuning

We initialize the multimodal search policy with supervised fine-tuning (SFT) on teacher-generated search trajectories. A proprietary MLLM generates these trajectories using the agent framework and interaction protocol described in Section 3.1. We apply thorough quality filtering to the resulting trajectories and use the retained data to supervise the student's search behavior. Additional details on trajectory generation, filtering, and SFT training are provided in Appendix.

We study two effects of SFT: how its impact on task accuracy varies across students, and how it changes interaction-protocol compliance and search behavior. (a) We first train two different students on the same trajectory corpus and compare their accuracy before and after SFT. (b) We then track accuracy, protocol compliance, and tool use across SFT epochs for Qwen.

(a) Base vs SFT performance trend



(b) Tool-call behaviour

Training	Acc.	Tool usage (%) Image	Crop	Text	Visit
Corpus	—	92.6	7.5	90.1	49.6
Base	44.51	67.9	22.8	69.9	53.3
$SFT_{1\text{-epoch}}$	58.49	93.3	6.7	94.1	49.6
$SFT_{2\text{-epoch}}$	**60.11**	95.9	4.1	85.3	49.4
$SFT_{3\text{-epoch}}$	59.25	95.7	4.3	84.5	51.7

Figure 2: ***(a)*** Mean accuracy before and after SFT for Qwen and Gemma. ***(b)*** Tool-call behavior of the trajectory corpus and Qwen before and after SFT.

(a) The accuracy effect of SFT depends on the student. Qwen gains 15.60 points after SFT, whereas Gemma loses 3.27 points as shown in Figure 2(a). Gemma starts from a substantially stronger agentic-search baseline, with 59.46% mean accuracy before SFT compared with 44.51% for Qwen, suggesting that the same SFT corpus need not improve all students. This is because SFT distills not only the interaction protocol but also the search behavior represented in the trajectory corpus. As shown in Figure 2(b), the student's tool-call distribution shifts toward the corpus after SFT. This benefits a weaker initialization such as Qwen, but can overwrite useful search behavior already present in a stronger student such as Gemma.

(b) SFT distills the tool-call behavior of the trajectory corpus. For Qwen, one epoch raises mean accuracy from 44.51% to 58.49%, while further SFT provides little additional gain as shown in Figure 2(b). Notably, the student's tool-use pattern shifts markedly toward the trajectory corpus. Image search increases from 67.9% to 93.3%, close to the corpus rate of 92.6%, while *crop_and_search* decreases from 22.8% to 6.7%, close to the corpus rate of 7.5%. Text search similarly moves from 69.9% to 94.1%, compared with 90.1% in the corpus, while *web_visit* remains near the corpus rate throughout SFT. These results show that SFT rapidly transfers the tool-call behavior represented in the trajectory corpus, while additional epochs yield limited accuracy improvement.

Lesson 2: SFT distills the tool-call behavior of the trajectory corpus, while performance gains depend on the student.

SFT rapidly transfers the tool-call distribution represented in the training trajectories, while its effect on performance depends on the student. The trajectory corpus does more than teach the interaction protocol, it also specifies the search behavior distilled into the policy. This makes data generation and filtering a critical design choice, and the trajectory corpus should reflect the tool-use distribution and search strategy that we want the search policy to adopt.

3.4 Effect of interaction budget at training and inference

We study how the number of interaction turns affects performance through two complementary experiments. (a) We compare SFT policies trained with different trajectory lengths. One policy is trained on five-turn trajectories and evaluated with $T = 5$, while a second policy is trained on thirty-turn trajectories and evaluated with $T = 30$. This comparison tests whether training and operating with longer search trajectories improves performance. (b) We isolate the effect of inference-time interaction turns by keeping the model checkpoint fixed and varying only T. We perform this experiment for the base *Qwen3-VL-30B-A3B-Instruct* model and *GPT-5.2*, while holding the tools, prompts, summarizer, and judge fixed. This allows us to examine whether simply allowing the same policy to search for more turns improves performance.

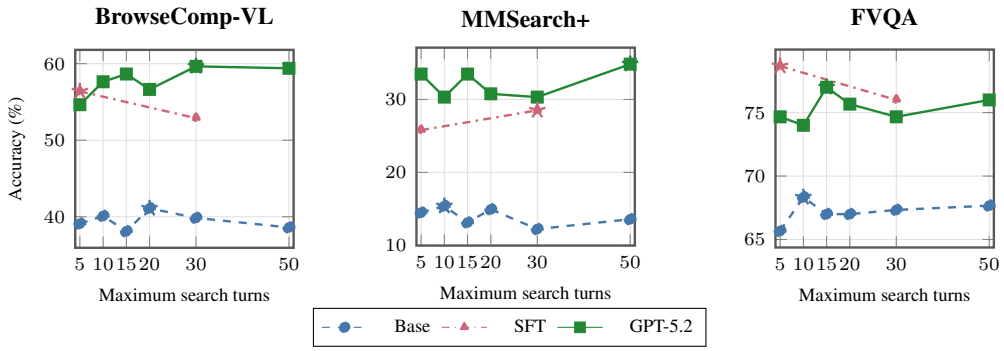


Figure 3: Performance across different numbers of interaction turns. Stars indicate the highest-performing setting for each policy and benchmark.

(a) SFT trajectory length. The SFT comparison (red) in Figure 3 shows that training and operating with longer search trajectories does not yield a consistent performance benefit. Moving from five-turn to thirty-turn SFT improves MMSearch+ from 25.79% to 28.51%, but reduces BrowseComp-VL from 56.39% to 52.88% and FVQA from 78.67% to 76.00%. We observe that longer teacher trajectories are not inherently more useful supervision; they increase the amount of search behavior being distilled without reliably improving the resulting policy. This trade-off is also reflected in efficiency, as the thirty-turn setting increases inference cost and wall-clock time by A and B, respectively, without a corresponding performance gain. The additional compute required by longer SFT trajectories is not justified by the observed downstream performance.

(b) Effect of additional interaction turns. For the base Qwen model (blue) and GPT-5.2 (green) both exhibit non-monotonic performance as T increases. This becomes easier to interpret by separating trajectories that exhaust the five-turn limit from those that terminate earlier. For GPT-5.2 on BrowseComp-VL, 30.3% of examples use all five available tool turns at $T = 5$; allowing these same examples up to 30 turns improves accuracy by 15.70 points, whereas examples that already terminate before five turns change by only 0.36 points. However, additional turns do not uniformly help even when the five-turn limit is reached: the longer run recovers 25 initially incorrect examples but also changes 6 initially correct examples to incorrect, consistent with query drift and the accumulation of irrelevant or conflicting evidence. The same non-monotonic behavior appears across benchmarks and policies. For Qwen, increasing T from 10 to 30 decreases MMSearch+ accuracy from 15.38% to 12.22%, while increasing T from 20 to 50 decreases BrowseComp-VL from 41.10% to 38.60%. GPT-5.2 shows similar drops: on FVQA, accuracy decreases from 77.00% at $T = 15$ to 74.67%

at $T = 30$, and on MMSearch+, from 33.48% at $T = 5$ to 30.32% at $T = 10$. We conclude that additional interaction turns are most useful when the shorter trajectory is truncated by the turn limit, but increasing the number of turns alone does not guarantee better performance.

Lesson 3: More interaction turns do not reliably improve search performance.

Longer SFT trajectories are not inherently more useful supervision and incur substantially higher compute without consistent performance gains. At inference, additional interaction turns help primarily when the shorter trajectory is truncated by the turn limit, and even then the benefit is not guaranteed. These results favor training and operating with a small number of interaction turns for routine information-seeking queries, while longer search may be better reserved for genuinely complex deep-research queries.

3.5 Reinforcement Learning using branching rollouts for step-level credit

The problem. A common post-training recipe for multimodal search agents is supervised fine-tuning followed by outcome-based reinforcement learning. Typically, GRPO samples independent trajectories for each prompt and computes group-relative advantages from their scalar terminal rewards. This setup has two limitations; (1) *credit assignment*: the same trajectory-level advantage is applied across all turns, without distinguishing their individual contributions to the outcome. An unproductive early query is reinforced whenever a later turn recovers, and a well-chosen search is penalised whenever an unrelated downstream step fails. (2) *training-signal availability*: learning from relative outcomes requires reward variation within each group. When all sampled trajectories receive the same reward, their centred rewards $r_i - \bar{r}$ are zero, yielding no outcome-based policy-gradient signal despite the cost of generating them. In our training pool, this occurs for 68.8% of prompts at group size four, comprising 43.6% all-incorrect and 25.2% all-correct groups. These limitations motivate branching rollouts which addresses both at once by replacing part of the independent group with siblings that share a prefix: a comparison between two continuations of the *same* history provides more targeted credit assignment to decisions made after the fork, and it creates reward variance at the point where the decision was made rather than only at the trajectory level.

Branching cost and the small interaction budget. Branching provides additional continuation at turn-level, but each alternative continuation incurs additional generation, retrieval, and summarization costs. Recursively forking at every turn would therefore be expensive: with width k at each of the $T - 1$ turns, a fully expanded tree contains

$$N_{\text{full}} = k^{T-1}, \tag{3}$$

leaves, assuming that all trajectories reach the maximum depth. For $k = 3$, this yields 81 terminal trajectories at $T = 5$ and approximately 7×10^{13} at $T = 30$, making full expansion computationally prohibitive. We instead use a *vine* structure, in which alternative continuations branch only from the original trajectory, or *trunk*, and are not recursively expanded. For m selected fork points with widths $k_1, \ldots, k_m$, where each width includes the trunk continuation, the number of leaves is

$$N_{\text{vine}} = 1 + \sum_{j=1}^{m} (k_j - 1), \tag{4}$$

which grows linearly with the number of fork points. As shown in Section 3.4, training and operating with longer trajectories does not consistently improve performance. We therefore initialize RL from the policy trained on five-turn SFT trajectories and retain $T = 5$ during RL, allocating compute to sibling comparisons rather than extending individual search trajectories. Even with the vine structure, however, forking every turn may produce redundant comparisons or identical rewards. We therefore study model-dependent signals, including entropy and self-confidence, together with data-driven heuristics to determine where the branching budget is most effectively allocated.

Tree construction. For each prompt, we roll out a single *trunk* trajectory through the five-turn framework, then sample alternative continuations at selected turn boundaries. Each continuation runs to completion under the same agent framework, tools, and interaction protocol as the trunk.

Let $\tau^{(0)}$ denote the trunk, and let $\mathcal{F} = \{f_1, \ldots, f_m\}$ be the selected fork turns. At fork f_j, all siblings condition on the same trunk prefix

$$h_{f_j}^{(0)} = (x, v, a_1^{(0)}, o_1^{(0)}, \ldots, a_{f_j-1}^{(0)}, o_{f_j-1}^{(0)}), \tag{5}$$

but independently sample the continuation

$$\tau_{f_j:}^{(j,\ell)} \sim \pi_\theta\Big(\cdot \mid h_{f_j}^{(0)}\Big), \qquad \ell = 1, \ldots, k_j - 1. \tag{6}$$

The leaf set is therefore $\mathcal{T}(x) = \{\tau^{(0)}\} \cup \{\tau^{(j,\ell)}\}_{j,\ell}$ and has the size given by Equation 4. This construction changes the sampled data but leaves the agent framework, reward, and optimization objective unchanged.

(i) Fork boundary. We fork at turn boundaries so that siblings regenerate their reasoning and can change the tool, query, or decision to answer. If we fork only after the trunk's reasoning block, siblings inherit reasoning that often already states which query to issue, limiting query diversity. At temperature 0.8, 27% of sibling queries sampled this way were byte-identical to the trunk's query. *(ii) Fork depth and width.* Only the trunk forks, forming a vine with two fork points of width five and $1 + (5 - 1) + (5 - 1) = 9$ leaves. Equal widths isolate depth effects from width effects. If only one fork point is reached, we widen that fork to retain nine leaves. We retain a prompt for training if at least two leaves survive; otherwise, we exclude its group from advantage estimation, including singleton groups from trunks that never fork. *(iii) Compute matching.* We sample one tree per prompt, yielding nine trajectories whenever a fork occurs, matching the nine independent rollouts per prompt used by plain GRPO. This matches trajectory counts for a fair comparison, though actual rollout cost can differ with prefix sharing and trajectory length. The RLOO advantage estimator and prefix-cancellation derivation are provided in Appendix E.

We compare rollout strategies by the proportion of informative groups, defined as groups containing trajectories with different terminal rewards and therefore providing a nonzero group-relative learning signal. We generate rollouts using the SFT policy on 1,000 randomly selected samples from the RL training pool. We compare four branching strategies under a matched branching budget and tree structure. GRPO samples independent trajectories for each prompt, whereas branching generates alternative continuations from selected turn boundaries along a trunk trajectory, with each continuation sharing the history preceding its fork. Early-turn branching selects two early trunk turns. Entropy-based branching selects the two trunk turns with the highest token entropy, while self-confidence-based branching selects the two turns that the policy itself rates as least confident. Random branching selects two trunk turns uniformly at random.

Table 3: Comparison of informative rollout groups with p (McNemar)

Rollout strategy	**Informative (%)**	p
GRPO (i.i.d)	40.88	0.031
Branching (random)	37.16	0.002
Branching (entropy)	38.18	0.003
Branching (self-confidence)	38.68	0.006
Branching (early turn)	**43.07**	—

As shown in Table 3, early-turn branching yields a statistically significantly higher proportion of informative rollout groups than GRPO and the other branching strategies. These results suggest that model-dependent signals such as entropy and self-confidence do not reliably identify fork points that produce diverse rollout outcomes. Instead, our empirical comparison supports the simple heuristic of branching at early turns, which produces more informative rollout groups.

Lesson 4: Early-turn branching, a data driven heuristic produces more informative rollout groups for RL training.

Under a matched rollout count, branching at early turns produces more groups with differing terminal rewards than independent sampling or branching based on entropy, self-confidence, or random turn selection. This supports a data-driven heuristic that concentrates branching on early search decisions to obtain more informative comparisons for reinforcement learning.

4 Results

Final post-training design choice. Based on the studies above, our final post-training recipe starts with Qwen3-VL-30B-A3B-Instruct and supervised fine-tunes it on the filtered teacher trajectories to learn the agent protocol and search behavior. We retain the five-turn interaction budget ($T = 5$), since the experiments in Section 3.4 show that longer trajectories increase cost without providing consistent gains. Reinforcement learning is then initialized from this SFT policy and uses the branching rollout described in Section 3.5. For each prompt, we generate one trunk and fork the earliest eligible turns, producing a nine-leaf vine from two fork points of width five. The leaves are

Table 4: *Comparison with direct-inference, agentic-inference, and multimodal deep-search baselines.* We report accuracy (%) on seven benchmarks and the average across benchmarks when all scores are available. The highest and second-highest values within each model group are shown in **red** and **blue**, respectively.

Model	BC-VL	VDR-Bench	MMSearch	MMSearch+	FVQA	SimpleVQA	Live-VQA	Avg
Direct Inference								
GPT-4o	35.58	6.60	25.73	4.97	51.00	54.00	47.00	32.13
GPT-5.2	36.84	9.20	30.99	3.61	36.33	49.33	52.33	31.23
Claude-4.6-Sonnet	29.82	7.80	38.01	9.04	48.00	53.00	49.66	33.62
Claude-4.7-Opus	36.34	**11.60**	**40.35**	**10.40**	**56.00**	53.00	44.66	**36.05**
Gemini-3.0-flash	**47.11**	**19.80**	**67.25**	**23.52**	**65.33**	**71.33**	**69.00**	**51.91**
Gemma4-26B-A4B-IT	28.07	3.60	21.05	1.80	32.33	48.67	44.66	25.74
Qwen3-VL-8B-Instruct	25.06	2.80	9.94	2.71	27.66	51.67	45.33	23.60
Qwen3-VL-30B-A3B-Instruct	27.31	2.40	19.88	3.16	34.66	52.33	46.66	26.63
Qwen3.5-35B-A3B	**37.59**	5.60	27.48	3.16	36.66	**60.33**	57.33	32.59
Qwen3.6-35B-A3B	35.83	6.60	26.31	5.88	35.00	56.33	**59.33**	32.18
Agentic Inference								
GPT-4o	41.60	17.40	64.32	19.45	67.66	72.00	65.33	49.68
GPT-5.2	52.13	29.40	76.60	28.95	69.33	70.33	68.33	56.44
Claude-4.6-Sonnet	50.87	27.60	79.53	23.52	68.33	77.00	77.00	57.69
Claude-4.7-Opus	**61.40**	**34.40**	**85.96**	**44.34**	**82.66**	**81.00**	**84.66**	**67.77**
Gemini-3.0-flash	**58.39**	**37.00**	**84.79**	**51.13**	**80.00**	**80.33**	**84.33**	**68.00**
Gemma4-26B-A4B-IT	52.38	31.40	78.36	25.79	77.66	73.00	82.00	60.08
Qwen3-VL-30B-A3B-Instruct	37.84	19.20	60.23	10.85	60.00	61.67	60.33	44.30
Multimodal Deep Search Agents								
WebWatcher-32B	27.00	–	55.30	–	–	59.00	58.70	–
SenseNova-MARS-32B	–	–	74.27	–	72.61	74.14	60.83	–
MM-DeepResearch-32B	43.00	–	69.00	–	70.10	67.60	68.00	–
Vision-DeepResearch-30B-A3B	53.70	**37.80**	69.60	28.50	74.20	–	77.60	–
OpenSearch-VL-30B-A3B	41.10	**33.50**	68.70	–	73.20	**74.90**	67.40	–
MTAAgent-32B	53.77	–	**82.35**	**31.93**	76.00	–	–	–
ODE-30B-A3B	46.10	26.40	69.70	28.10	69.30	71.00	–	–
Ours (SFT)	**56.39**	25.80	78.36	26.69	**78.33**	74.67	**79.00**	**59.89**
Ours (RL)	**59.89**	27.80	**81.87**	**33.94**	**80.00**	**75.67**	**82.00**	**63.02**

optimized jointly with RLOO using only their terminal outcome rewards. This design spends rollout compute on comparing alternative search decisions from shared histories, rather than on extending a single trajectory or introducing step-level reward annotations.

Main results. Table 4 compares our final post-training receipe with direct-inference, agentic-inference, and multimodal deep-search baselines. SFT alone reaches an average accuracy of 59.89%, substantially improving on the agentic result of the underlying Qwen3-VL-30B-A3B-Instruct model (44.30%). Branching RL further raises the average to 63.02%, a gain of 3.13 points over SFT. The gains are consistent across all seven benchmarks: $+3.50$ on BrowseComp-VL, $+2.00$ on VDR-Bench, $+3.51$ on MMSearch, $+7.25$ on MMSearch+, $+1.67$ on FVQA, $+1.00$ on SimpleVQA, and $+3.00$ on Live-VQA. The largest improvement occurs on MMSearch+, where accuracy rises from 26.69% to 33.94%, suggesting that the branching objective is particularly useful for questions requiring stronger search decisions. These results show that the gains from branching RL are not confined to a single dataset and that the compact five-turn agent remains competitive with both specialized open-weight search agents and strong proprietary agentic systems, with a small budget of five interaction turns.

5 Conclusion

In this work, we conduct an empirical investigation into evaluation and post-training for multimodal web search agents. Our findings show that evaluation choices substantially affect measured accuracy, SFT transfers corpus-specific tool-use behavior with student-dependent benefits, and longer search horizons do not consistently improve performance. Early-turn branching yields more informative reward comparisons, enabling targeted policy updates using only terminal rewards. These findings guide the design of our practical post-training recipe. The resulting five-turn agent improves average accuracy from 59.89% to 63.02%, with gains across all seven benchmarks.

AI USE STATEMENT

We used generative AI tools (LLM-based coding and writing assistants) in the preparation of this work, as follows.

Writing. We used AI assistance to polish English prose, grammar, clarity, and concision throughout the paper, and to draft and revise portions of the methods, evaluation-setup, and appendix text from author-supplied material (experimental records, measured numbers, and outlines). All such text was read, edited, and verified against the underlying experimental artifacts by the authors. No claim, result, or interpretation in this paper originates from a model: every quantitative statement is traceable to an artifact produced by our own experiments, and we re-derived the reported numbers from those artifacts rather than accepting generated values.

Code and figures. We used AI assistance to write, refactor, and debug code: analysis and evaluation scripts, plotting code for the figures, and parts of the training and evaluation harness. We also used it for LaTeX formatting (tables, figure environments, and document structure). AI-assisted analysis scripts were reviewed by the author(s), and their outputs were checked by recomputing key statistics independently and by verifying agreement with previously measured values where available.

Tasks for which we did not use generative AI. The research questions, hypotheses, and claimed contributions of this paper were formulated by the authors. AI tools were not used for experimental design, the choice of benchmarks, ablations, training configurations, and evaluation protocol nor to decide what to conclude from the results.

Distinct from the above: AI systems as objects of study. This paper studies multimodal search agents, so large models appear inside the method itself: a proprietary MLLM generates the supervised fine-tuning trajectories, a separate model summarizes retrieved pages, and LLM judges score correctness. These are components of the system under evaluation, documented in the main text and appendix, and are not authoring assistance.

We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The main text and appendix document the agent framework, evaluation setup, and trajectory-generation and filtering procedures used in our experiments. Section 3.1 specifies the agent framework, tool interface, and turn structure, while Section 3.2 describes the fixed evaluation stack, including the search and page-retrieval providers, page summarizer, and judge model and prompt. Appendix C describes the evaluation benchmarks and reports their sample counts (Table 7). Both judge prompts and their token budgets appear in Appendix B, and Appendix A reports the complete factorial analysis over the summarizer, judge model, and judge prompt for the three benchmarks included in the sensitivity analysis. For the five-turn supervised fine-tuning corpus, Appendix D.1 documents the teacher model, generation settings, per-turn caps, and five question pools. Appendix D.2 specifies the ordered filters, grader model and settings, grading criteria, filtering counts (Table 8), and corpus statistics. Appendix D.3 documents the Qwen SFT training configuration, hardware, and epoch-sweep protocol.

REFERENCES

Shuang Chen, Kaituo Feng, Hangting Chen, Wenxuan Huang, Dasen Dai, Quanxin Shou, Yunlong Lin, Xiangyu Yue, Shenghua Gao, and Tianyu Pang. Opensearch-vl: An open recipe for frontier multimodal search agents. *arXiv preprint arXiv:2605.05185*, 2026.

Xianfu Cheng, Wei Zhang, Shiwei Zhang, Jian Yang, Xiangyuan Guan, Xianjie Wu, Xiang Li, Ge Zhang, Jiaheng Liu, Yuying Mai, et al. Simplevqa: Multimodal factuality evaluation for multimodal large language models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4637–4646. IEEE, 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.

Mingyang Fu, Yuyang Peng, Benlin Liu, Yao Wan, and Dongping Chen. Livevqa: Live visual knowledge seeking. *arXiv e-prints*, pp. arXiv–2504, 2025.

Xinyu Geng, Peng Xia, Zhen Zhang, Xinyu Wang, Qiuchen Wang, Ruixue Ding, Chenxi Wang, Jialong Wu, Kuan Li, Yida Zhao, et al. Webwatcher: Breaking new frontiers of vision-language deep research agent. In *International Conference on Learning Representations*, volume 2026, pp. 61240–61271, 2026.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, and Guohai Xu. Deepeyesv2: Toward agentic multimodal model. In *International Conference on Learning Representations*, volume 2026, pp. 114851–114872, 2026.

Shijue Huang, Hangyu Guo, Guanting Dong, Chenxin Li, Junting Lu, Xinyu Geng, Zhaochen Su, Zhenyu Li, Shuang Chen, Hongru Wang, et al. Towards on-policy data evolution for visual-native multimodal deep search agents. *arXiv preprint arXiv:2605.10832*, 2026a.

Wenxuan Huang, Yu Zeng, Qiuchen Wang, Zhen Fang, Shaosheng Cao, Zheng Chu, Qingyu Yin, Shuang Chen, Zhenfei Yin, Lin Chen, et al. Vision-deepresearch: Incentivizing deepresearch capability in multimodal large language models. *arXiv preprint arXiv:2601.22060*, 2026b.

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.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Shilong Li, Xingyuan Bu, Wenjie Wang, Jiaheng Liu, Jun Dong, Haoyang He, Hao Lu, Haozhe Zhang, Chenchen Jing, Zhen Li, et al. Mm-browsecomp: A comprehensive benchmark for multimodal browsing agents. *arXiv preprint arXiv:2508.13186*, 2025.

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.

Xiangyu Peng, Can Qin, An Yan, Xinyi Yang, Zeyuan Chen, Ran Xu, and Chien-Sheng Wu. Mtagent: An open recipe for multimodal deep search agents. *arXiv preprint arXiv:2604.06376*, 2026.

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, Teng Yihua, 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. In *International Conference on Learning Representations*, volume 2026, pp. 70385–70422, 2026.

Peng Wang, Qi Wu, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. Fvqa: Fact-based visual question answering. *IEEE transactions on pattern analysis and machine intelligence*, 40 (10):2413–2427, 2017.

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*, 2(4), 2025.

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 Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

APPENDIX

This appendix presents the full evaluation results, judge prompts, benchmark and dataset details, trajectory generation and filtering procedures, and the RLOO advantage estimator.

Appendix A **Full evaluation results.** Results for every combination of summarizer, judge model, and judge prompt, plus additional comparisons of GRPO, TreeGRPO, and our RL policy.

Appendix B **Exact judge prompts.** Recall and semantic-equivalence prompts, judge settings, and verdict parsing.

Appendix C **Benchmarks and dataset statistics.** Benchmark descriptions, evaluation sample counts, and aggregation procedures.

Appendix D **SFT trajectory and training details.** Teacher trajectory generation, quality filtering, retained corpus statistics, and Qwen SFT training configuration and epoch sweep.

Appendix E **RLOO advantage estimation.** Leave-one-out advantages, the policy-gradient objective, and shared-prefix cancellation.

A FULL EVALUATION RESULTS

Table 5 reports the full factorial evaluation used in the component-sensitivity analysis in Table 1. For each policy-summarizer pair, the four judge configurations score the same stored predictions. Comparisons between judge configurations are therefore paired and can be assessed using McNemar's test. Changing the summarizer affects the rollout itself, so each summarizer is evaluated using a separate set of trajectories. The sweep includes $14{,}720$ judge calls, with no empty responses or call errors. All evaluated examples therefore remain in the accuracy denominators, as defined in Appendix C.2.

Table 5: Complete evaluation-stack sweep. Accuracy (%) for the base and SFT policies under all combinations of two summarizers, two judge prompts, and two judge models. The maximum paired component effects summarized in Table 1 are computed from these results. *Mean* is the unweighted mean of the three benchmark accuracies, not an item-pooled accuracy.

Policy	Summarizer	Judge prompt	Judge model	BC-VL ($n = 399$)	MMS+ ($n = 221$)	FVQA ($n = 300$)	Mean
Base	qwen-235b	Recall	gpt-5-mini	37.59	13.12	65.00	38.57
Base	qwen-235b	Recall	qwen-235b	38.35	14.48	66.67	39.83
Base	qwen-235b	Semantic	gpt-5-mini	35.84	11.76	65.33	37.64
Base	qwen-235b	Semantic	qwen-235b	32.83	12.67	63.67	36.39
Base	sonnet-5	Recall	gpt-5-mini	39.35	14.03	71.00	41.46
Base	sonnet-5	Recall	qwen-235b	39.10	14.03	72.00	41.71
Base	sonnet-5	Semantic	gpt-5-mini	37.59	13.57	70.33	40.50
Base	sonnet-5	Semantic	qwen-235b	36.34	13.57	68.33	39.41
SFT	qwen-235b	Recall	gpt-5-mini	53.13	27.15	76.33	52.20
SFT	qwen-235b	Recall	qwen-235b	52.38	26.24	76.67	51.76
SFT	qwen-235b	Semantic	gpt-5-mini	51.63	24.43	73.67	49.91
SFT	qwen-235b	Semantic	qwen-235b	48.37	25.79	72.00	48.72
SFT	sonnet-5	Recall	gpt-5-mini	54.64	26.24	77.33	52.74
SFT	sonnet-5	Recall	qwen-235b	53.88	28.05	79.67	53.87
SFT	sonnet-5	Semantic	gpt-5-mini	51.38	24.43	76.33	50.71
SFT	sonnet-5	Semantic	qwen-235b	48.62	26.24	76.00	50.29

A.1 ADDITIONAL RL COMPARISONS

Table 6 compares GRPO, TreeGRPO, and our RL policy on BrowseComp-VL, MMSearch+, and FVQA. GRPO samples nine independent trajectories per prompt. TreeGRPO branches at two successive fork points with a branching factor of three at each, yielding $3 \times 3 = 9$ terminal trajectories. Our method instead uses a vine structure with two fork points of width five along the original trunk,

yielding $1 + (5 - 1) + (5 - 1) = 9$ terminal trajectories. Thus, the methods match the nominal trajectory budget per prompt, although token usage and tool-call costs may differ.

Our RL policy achieves the highest accuracy on all three benchmarks. TreeGRPO improves BrowseComp-VL accuracy over GRPO but reduces accuracy on MMSearch+ and FVQA, whereas our method improves performance across all three benchmarks.

Table 6 compares GRPO, TreeGRPO, and our RL policy on BrowseComp-VL, MMSearch+, and FVQA. Our policy achieves the highest accuracy across all three benchmarks, exceeding GRPO by 6.01, 7.70, and 3.54 percentage points, respectively.

Table 6: Additional RL comparison. Accuracy (%) on BrowseComp-VL, MMSearch+, and FVQA. The highest score on each benchmark is shown in bold.

Method	BC-VL	MMSearch+	FVQA
GRPO	53.88	26.24	78.33
TreeGRPO	57.14	23.07	77.33
Ours (RL)	**59.89**	**33.94**	**81.87**

B Exact Judge Prompts

The two prompts below are reproduced verbatim. The placeholders {`question`}, {`correct_answer`}, and {`response`} are replaced with the question, reference answer, and policy's complete final report, respectively. Both prompts request the verdict before the explanation, allowing a truncated response to be parsed if the verdict is present. A response with no parsable verdict is scored as incorrect.

Judge settings. The reply budget is $1{,}024$ tokens on BrowseComp-VL, MMSearch+, FVQA, LiveVQA, SimpleVQA and VDR-Bench, and $4{,}096$ on the MMSearch vision subset. For reasoning models this is sent as `max_completion_tokens` and must therefore cover both hidden reasoning and visible output tokens. For a representative BrowseComp-VL judge prompt of approximately $6{,}100$ characters, gpt-5-mini used 64 reasoning tokens and 54 visible output tokens, well below the limit. If the token limit is reached and no content is returned, the response is scored as incorrect.

The **recall prompt** (Appendix B.1) is the established judge prompt used by the benchmark evaluation protocol. We use it without modification so that our reported numbers remain comparable to published ones; it is used for all headline evaluation scores in this paper. The **semantic prompt** (Appendix B.2) is the stricter alternative used in the sensitivity analysis of Table 1, and it is also the reward judge for reinforcement learning, where reference answers are long-form and the recall framing is unreliable (see Section 3.5). Teacher trajectories are filtered using a separate grader and prompt (Appendix D.2).

B.1 Recall prompt (default; headline evaluation scores)

Recall judge prompt

```
You are an impartial judge evaluating whether a deep research report
contains the correct answer.

[Question]
{question}

[Correct Answer]
{correct_answer}

[Deep Research Report]
{response}

Task: Determine if the deep research report contains the correct answer
anywhere in its content. Don't use your internal knowledge to answer the
question.
```

```
Instructions:
1. Read through the entire research report carefully
2. Look for the correct answer anywhere in the report (it may be embedded
   in paragraphs, tables, or sections)
3. Check if the information in the report is consistent with the correct
   answer
4. The answer does NOT need to be in a specific format or labeled as
   "final answer"
5. Provide your reasoning
6. Return [CORRECT] if the report contains the correct answer,
   [INCORRECT] if it doesn't or contradicts it

Output format:
Answer: [CORRECT/INCORRECT]
Reasoning: [your explanation]
```

Verdicts are parsed by searching the reply for `[CORRECT]` or `[INCORRECT]`; if neither bracketed token is present, the reply is searched for the bare words, and anything still unresolved is scored incorrect.

B.2 SEMANTIC PROMPT (SENSITIVITY ANALYSIS; RL REWARD JUDGE)

Semantic-equivalence judge prompt

```
You are an impartial judge deciding whether a model's answer means the
SAME THING as a reference answer.

[Question]
{question}

[Reference Answer]
{correct_answer}

[Model Response]
{response}

Task: decide whether the model response gives the same answer to the
question as the reference answer does. Judge MEANING, not wording. Do not
use your own knowledge of the subject to decide who is right: the
reference answer is correct by definition, and your only job is to
compare the two.

Procedure:
1. From the question, identify exactly what is being asked for (an
   entity, a date, a quantity, a cause, a list of items, ...).
2. From the reference answer, extract only what it ASSERTS as that
   answer. The reference may be a long write-up containing reasoning,
   source citations, hedging, or LaTeX such as \boxed{...}; ignore that
   scaffolding and keep the substantive claim. If the reference marks a
   final answer (for example inside \boxed{}), that marked span IS the
   answer and the surrounding prose is only support.
3. From the model response, extract the answer it COMMITS to. If it lists
   several mutually exclusive candidates without settling on one, or
   states that it could not determine the answer, it has not committed to
   anything.
4. Compare the two extracted answers.

Treat as the SAME answer:
- paraphrase, translation, transliteration, or a different language
- aliases, abbreviations, maiden/married names, official vs common names
  for the same entity
- different formatting, units, precision or rounding that denotes the
  same value
- additional correct detail in the model response beyond what the
  reference states
- the model reaching the answer by different reasoning or from different
  sources

Treat as a DIFFERENT answer:
- a different entity, value, date or claim, even a closely related or
  adjacent one
- no commitment: several mutually exclusive candidates, or an explicit
  failure to determine
- less specific than the question requires (the question asks which year,
  the response gives only a decade)
- the reference answer requires several components and the model supplies
  only some of them
```

```
- the model asserts the reference's answer in one place but contradicts
  it in another

Output format, verdict FIRST:
Answer: [CORRECT/INCORRECT]
Reasoning: [at most 4 sentences: the answer you extracted from each side,
and why they match or do not]
```

Comparison of judging criteria. The recall prompt emphasizes whether the report contains the reference answer, whereas the semantic prompt explicitly requires a committed answer with the specificity and completeness requested by the question. These criteria can produce different judgments for reports that list unresolved candidates or provide only part of a required answer. Both prompts instruct the judge to reject contradictions; therefore, a contradictory report is not a guaranteed case of disagreement. The semantic prompt generally produces lower scores in Table 5, but not in every configuration: for the base policy with the qwen-235b summarizer and gpt-5-mini judge, FVQA accuracy is 65.00% under recall judging and 65.33% under semantic judging. The aggregate results do not establish how disagreement varies with answer length.

C Benchmarks and Dataset Statistics

C.1 Benchmark descriptions

We evaluate on seven benchmarks covering visual factual knowledge and multimodal web search. The descriptions below summarize the capabilities assessed by each benchmark and its role in our evaluation. References to the original benchmarks are provided in the main paper.

FVQA. FVQA evaluates fact-based visual question answering in settings where the image alone is insufficient to answer the question. Each question is paired with a structured supporting fact. Answering correctly requires identifying a visual concept and connecting it to external factual or commonsense knowledge.

SimpleVQA. SimpleVQA evaluates visual factual knowledge across a broad range of tasks and domains using concise, objectively gradable answers. It tests whether a model can combine image understanding with stable factual or commonsense knowledge.

LiveVQA. LiveVQA evaluates dynamic visual knowledge using recent online material, including news, videos, and academic content. It includes both direct questions about visual entities and multi-hop questions that require retrieving information that may postdate the model's training data.

MMSearch. MMSearch evaluates multimodal search capabilities, including query formulation, retrieval, reranking, evidence summarization, and end-to-end question answering. We use only the vision subset, which focuses on image-grounded questions requiring external search.

MMSearch+. MMSearch+ evaluates browsing tasks that require tracking evidence provenance. Agents must identify fine-grained visual details in the input image and use them in successive image and text searches. The questions require cross-validating evidence under noisy retrieval conditions.

VDR-Bench. VDR-Bench evaluates visual deep research under realistic, noisy retrieval conditions. It tests entity localization, region-level or cropped-image search, text browsing, and cross-modal multi-hop reasoning beyond near-exact matching of whole images.

BrowseComp-VL. BrowseComp-VL extends persistent web-browsing tasks to visual inputs, using questions that are deliberately indirect or obscure the target entity. Agents must use the image to identify an initial search anchor, retrieve dispersed textual and visual clues, and combine them through multi-step browsing and planning.

C.2 Evaluation sample counts

Table 7 lists the number of examples evaluated for each benchmark. Every run evaluates all examples in the selected benchmark file, without further subsampling or a row-index offset. The main comparison and the evaluation-stack sweep therefore use the same examples for each benchmark included in both analyses.

Table 7: Evaluation sample counts. n is the number of evaluated examples; *Distinct questions* counts unique question strings after whitespace and punctuation normalization. Repeated question templates across images reduce this count, particularly for SimpleVQA. *Sweep* n gives the number of examples used in Table 5; a dash indicates that the benchmark is not included in that sweep. The total for distinct questions is the sum of the per-benchmark counts.

Benchmark	n	Distinct questions	Sweep n
FVQA	300	277	300
SimpleVQA	300	119	—
LiveVQA	300	300	—
MMSearch	171	161	—
MMSearch+	221	201	221
VDR-Bench	500	500	—
BrowseComp-VL	399	397	399
Total	**2,191**	1,955	920

Evaluation and aggregation. The default evaluation uses Serper for image and text search, Jina.ai for webpage retrieval, `claude-sonnet-5` for webpage and image-result summarization, and `gpt-5-mini` with the recall prompt of Appendix B.1 (see that appendix for the judge's token budget). The final system uses a budget of five tool-enabled turns, followed by an answer-only turn if necessary. Accuracy is the fraction of judged examples scored as correct. Missing, empty, or malformed predictions are scored as incorrect and remain in the denominator. An example is excluded from the denominator only if the judge call raises an exception. No such exceptions occurred in the 14,720 calls in Table 5, so the denominators in that table equal the full evaluated sample counts. Tool failures are retried inside the rollout and otherwise surface to the policy as an error string in the observation; they do not remove a question from the evaluation. Each reported cell corresponds to a single evaluation run. We report variation across repeated runs separately rather than averaging the runs.

D Trajectory Generation, Filtering, and SFT Training

D.1 Teacher trajectory generation

We generate teacher trajectories using `claude-opus-4-6-v1` within the agent framework described in the main paper. The teacher uses the same tools, turn constraints, and summarized observations as the student at rollout time. Pages and image results retrieved during generation are condensed by `claude-sonnet-5`.

The turn structure matches the evaluation protocol:

1. **Turn 1: visual search.** The teacher must call `image_search` or `crop_and_search`; it cannot provide a final answer on this turn.
2. **Turns 2–5: research or answer.** On each turn, the teacher may call `text_search` or `web_visit`, or provide a final answer. A final answer ends the trajectory.
3. **Turn 6: answer only.** If all five tool-enabled turns have been used, the teacher must provide a final answer on the sixth turn.

The tool-turn budget is therefore $T{=}5$, matching the final system's inference budget.

Generation settings. Each tool-use turn contains exactly one tool call. Batched arguments are capped at three queries per `text_search`, three URLs per `web_visit`, and five boxes per `crop_and_search`; calls beyond the cap within a turn are folded into one call and truncated to the cap, so the recorded call is exactly the call that was dispatched. Sampling uses temperature 1.0 (the teacher's extended-thinking default) with a 65,536-token generation limit, at concurrency 10 and 100 requests per minute per gateway. Retry budgets are three attempts per tool call, five per model call, and a 600-second wall-clock limit per turn; a turn that yields no usable tool call after all resamples terminates the trajectory as `incomplete_flow`.

Question pools. The training questions come from the five pools listed in Table 8.

D.2 Trajectory quality filtering

Generated trajectories pass through four checks in a fixed order, and a trajectory is discarded if *any* enabled check fails. Table 8 reports the counts.

1. **Format validity.** The recorded conversation must parse, every tool call must name a declared tool and type-check against its schema, each tool-use turn must contain exactly one call, the batched-argument caps must hold, and the turns must follow the mandatory-image-then-research phase order.
2. **Image-count consistency.** The number of image placeholders in the rendered conversation must equal the number of images attached to the row.
3. **Complete flow.** This check is available in the filtering pipeline but was disabled for the corpus used in this paper.
4. **Answer correctness.** An LLM grader receives the question, the reference answer and the trajectory's final report, and returns `VERDICT: CORRECT` or `VERDICT: INCORRECT`. A trajectory with an empty final answer fails; a row with no recorded reference answer passes unchecked. The grader is `claude-opus-4-7` — a different model from the teacher, from the evaluation judge, and from the RL reward judge — called at temperature 1.0 with a 512-token budget under its own strict-grader prompt, which asks whether the reference answer is *fully* present, treats paraphrase as acceptable, and rejects partial matches that omit an essential element.

Because the complete-flow check was disabled, the retained corpus contains 17 trajectories with an `incomplete_flow_research_1` termination (0.16% of the 10,328 retained examples). The format check validates the phase order of the recorded turns but does not guarantee a complete trajectory.

Retained corpus statistics. The retained corpus contains 10,328 trajectories, one per question identifier, and 9,106 distinct normalized question strings. The difference reflects repeated question templates within the pools, particularly in `infoseek`, which contains 374 distinct strings across 1,112 trajectories.

Trajectory length and termination. Trajectories contain a mean of 3.29 tool-enabled turns, with a median of 3 and a range of 1–5. The counts for lengths 1, 2, 3, 4, and 5 are 830, 2,389, 2,381, 2,361, and 2,367, respectively. The termination categories are `answer_early` 7,944 (76.9%), `answer` 1,710 (16.6%), `answer_forced` 657 (6.4%) and `incomplete_flow_research_1` 17 (0.16%).

Token lengths and supervision. Token lengths were measured by encoding all 10,328 rows using the training template, with truncation disabled, so they include image-token expansion (0 encode failures). Total sequence length has mean 8,711, median 8,032, $p90$ 13,785, $p99$ 19,874 and maximum 68,410 tokens. Two trajectories exceed the 65,536-token training limit. Left truncation, which can remove the system prompt and image content, therefore affected 0.02% of the corpus. Supervised (loss-carrying) tokens have mean 1,143, median 1,020 and maximum 9,936, the median fraction of tokens contributing to the loss is 12.4%; the rest is system prompt, tool schemas, images and tool responses, which are masked.

Table 8: Trajectory filtering, in execution order, for the five-turn SFT corpus. Each rejected trajectory is assigned to the first check it fails, so the rejection counts and the retained count sum to the number generated. A trajectory may fail multiple checks; counting failures independently gives 2,707 format failures and 6,256 correctness failures. The image-count check rejected nothing, and the complete-flow check was disabled.

Stage	Trajectories	Share of generated
Generated	16,777	100.0%
rejected: format invalid	−2,707	16.1%
rejected: image-count mismatch	−0	0.0%
rejected: incomplete flow *(check disabled)*	−0	0.0%
rejected: answer judged incorrect	−3,742	22.3%
Final retained trajectories	**10,328**	**61.6%**
Unique question identifiers	**10,328**	—
distinct question strings	9,106	—

Question pool	Generated	Rejected	Retained	Tool turns (mean)
`redsearcher_mm_sft_5k`	5,816	1,015	4,801	4.14
`livevqa`	4,981	2,070	2,911	2.82
`infoseek`	2,980	1,868	1,112	1.86
`evqa`	2,000	1,244	756	2.42
`vision_deepresearch_sft_1k`	1,000	252	748	2.75
Total	**16,777**	**6,449**	**10,328**	**3.29**

Tool-use statistics. The corpus contains 34,030 tool calls, 3.29 per trajectory. Of these, 17,959 are `text_search` calls carrying 46,319 queries (2.58 per call) and 5,743 are `web_visit` calls carrying 9,599 URLs (1.67 per call). We report two complementary measures of tool use, each with a different denominator. The *share of all tool calls* is over the 34,030 calls, counting every call: `text_search` 52.77%, `image_search` 28.09%, `web_visit` 16.88%, `crop_and_search` 2.26%. The *share of trajectories using the tool*, which is the quantity plotted in Figure 2(b), counts each trajectory once if it calls the tool at least once, regardless of how often: `image_search` 92.55%, `text_search` 90.09%, `web_visit` 49.63%, `crop_and_search` 7.45%. These measures capture different aspects of tool use. Most trajectories include an `image_search` call because the first turn requires either `image_search` or `crop_and_search`. In contrast, `text_search` can be called repeatedly within a trajectory, increasing its share of all calls.

D.3 SFT TRAINING CONFIGURATION

We fine-tune Qwen3-VL-30B-A3B-Instruct using the `megatron sft` interface in MS-Swift on the five question pools described in Appendix D.1, after applying the filtering procedures in Appendix D.2. We use full-parameter fine-tuning of the language model, while keeping the vision encoder and multimodal aligner frozen. Conversations are formatted using the `qwen3_vl` template, the Swift template backend, and the `hermes` agent template. The maximum sequence length is 65,536 tokens, with left truncation for sequences exceeding this limit.

Training settings. Each run uses one node with eight NVIDIA H200 GPUs with 141 GB of memory each. The global batch size is 16, and the micro-batch size is 4. The learning rate is set to 10^{-5}, with a minimum learning rate of 10^{-7} and a warmup fraction of 0.05 of the total training iterations. The mixture-of-experts auxiliary loss coefficient is 10^{-6}. We use tensor parallelism of degree 4, context parallelism of degree 2, expert parallelism of degree 8, and pipeline parallelism of degree 1. Sequence parallelism and the Flash attention backend are enabled. Full activation recomputation uses the uniform method with one layer per recomputation group.

Epoch sweep. We conduct separate runs for one, two, and three training epochs, keeping the remaining training configuration fixed. Each run starts from the same pretrained model rather than continuing from another sweep point. The warmup and learning-rate schedule are defined relative to each run's total training duration; consequently, the one- and two-epoch runs are not equivalent

to intermediate checkpoints of the three-epoch run. Checkpoints are saved every 200 training steps, with the two most recent checkpoints retained.

E RLOO Advantage Estimation

For a prompt with $n = |\mathcal{T}(x)| > 1$ surviving leaves, let $r_i = r(\tau_i)$ denote the terminal reward of leaf i. We compute its leave-one-out (RLOO) advantage using the mean reward of the other leaves as a baseline:

$$b_{-i} = \frac{1}{n-1} \sum_{j \neq i} r_j, \tag{7}$$

$$\widehat{A}_i^{\mathrm{RLOO}} = r_i - b_{-i} = \frac{n}{n-1} \left(r_i - \bar{r} \right), \qquad \bar{r} = \frac{1}{n} \sum_{j=1}^{n} r_j. \tag{8}$$

These advantages satisfy $\sum_{i=1}^{n} \widehat{A}_i^{\mathrm{RLOO}} = 0$. We use each leaf's terminal advantage to weight the policy gradients at its trained assistant-token positions. Denoting these positions by $\mathcal{M}_i$, the policy-gradient loss is

$$\mathcal{L}_{\mathrm{RLOO}}(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \widehat{A}_i^{\mathrm{RLOO}} \sum_{u \in \mathcal{M}_i} \log \pi_\theta(y_{i,u} \mid y_{i,<u}, x, v) \,. \tag{9}$$

Equation 9 shows how branching concentrates learning on alternative continuations without introducing a process reward. Sibling leaves share the same history before a fork but sample different continuations, whose gradients are weighted by their terminal rewards relative to the group. For a prefix p shared by every leaf, identical token contexts and training masks give a total gradient contribution proportional to

$$\left(\sum_{i=1}^{n} \widehat{A}_i^{\mathrm{RLOO}} \right) \nabla_\theta \log \pi_\theta(p \mid x, v) = 0. \tag{10}$$

Thus, under Equation 9, the common prefix contributes no net policy gradient, while alternative continuations receive updates weighted by their relative outcomes. This cancellation applies only to prefixes shared by the entire group, and a prefix shared by a subset of leaves need not cancel. With turn-level forks, the shared tokens before the first fork receive no net update. The model instead learns from the different reasoning and actions after the fork, favoring continuations with better final rewards. Branching requires no step-level labels, separate reward model, or modified policy loss; it changes only which trajectories are compared within each RLOO group.