LEARNING TO ROUTE IN VISUAL SPACE VIA MULTI-STEP EMBEDDING RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents rely on retrieval tools to access external knowledge, yet visual agentic search remains severely bottlenecked by standard single-step retrievers. In current pipelines, the agent must issue text queries for every intermediate step, struggling when visual clues are difficult to describe or when the retriever fails to surface necessary intermediate evidence within its top results. We hypothesize that offloading multi-step navigation across the entire embedding space directly to the retrieval tool resolves this performance bottleneck. To study this systematically, we introduce VHOP, a flexible data generation framework and benchmark with five core difficulty levels testing both visual matching and search planning. Using this framework, we develop VHOP-ROUTER, an end-to-end training pipeline—combining supervised fine-tuning, online imitation learning, and reinforcement learning—that transforms a standard embedding model into an autoregressive multi-step retriever. Operating directly in the visual latent space, VHOP-ROUTER retrieves linked image chains in a single tool call without requiring the agent to formulate intermediate text queries. Experiments show VHOP-ROUTER boosts retrieval performance from under 5% to 76.3%. In agentic search, it improves task success rates by 52.7% and reduces the average token length by 61% from 1886 to 728, whereas upgrading the agent yields only a 3.7% gain. Compared to a strong baseline where the agent retrieves the top 50 results per step, VHOP-ROUTER maintains superior performance while reducing in-context images by 23× and cutting the cumulative API payload by 35×. The models also generalize robustly to unseen difficulty levels and realistic test sets. Ultimately, VHOP and VHOP-ROUTER provide an efficient and effective solution for visual agentic search that leaves native LLM capabilities entirely intact.

1 INTRODUCTION

Agentic search equips large language model (LLM) agents with access to external knowledge bases, including up-to-date information and private data unavailable during training (Yao et al., 2022; Trivedi et al., 2023). Recent work fine-tunes LLMs to improve their reasoning and multi-step search (Jin et al., 2025). These improvements still depend on a fixed retrieval tool to supply relevant evidence. Because the agent sees only the few results returned by this tool, it cannot reason over evidence it never receives.

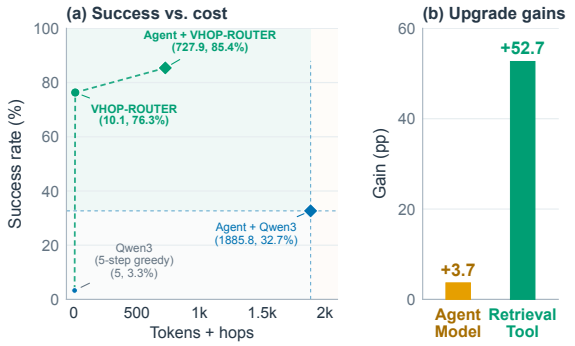


Figure 1: **Task performance and upgrade gains.** Agent denotes Gemini 3.5 Flash. (a) Success rate versus generated tokens plus retrieval hops. Standalone Qwen3 uses five greedy retrieval steps. (b) Gains over Agent + Qwen3 from upgrading the agent model to Gemini 3.1 Pro or replacing the retrieval tool with VHOP-ROUTER, keeping the other component fixed.

This bottleneck is especially important in visual search, where each retrieved image may reveal the clue for the next step. When locating a misplaced item in a photo album, visual clues can be hard to express in a textual

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.

query (Figure 2a) or be omitted from captions (Figure 2b). A standard embedding retriever performs one retrieval step per call, leaving the agent to build the search chain from the returned images. In Figure 2c, the retriever ranks the required next image 39th, outside the top-five results, leaving the red piggy bank needed for the following hop out of the agent's context. In contrast, embedding retrievers can score candidates across the entire indexed corpus.

We hypothesize that the main bottleneck in these scenarios is the retrieval tool itself. We therefore train this tool to navigate directly in visual embedding space over multiple steps, without agent fine-tuning. Shifting multi-hop search from the agent to the embedding model (replacing a single-step model with a multi-step one) significantly boosts task success rates while minimizing token length (Figure 1a). Upgrading to a stronger agent improves performance by only 3.7%, whereas a multi-step embedding model yields a 52.7% increase (Figure 1b). This validates that enhancing the retrieval tool is the far more effective approach. The following sections detail our data, evaluation, and training methods.

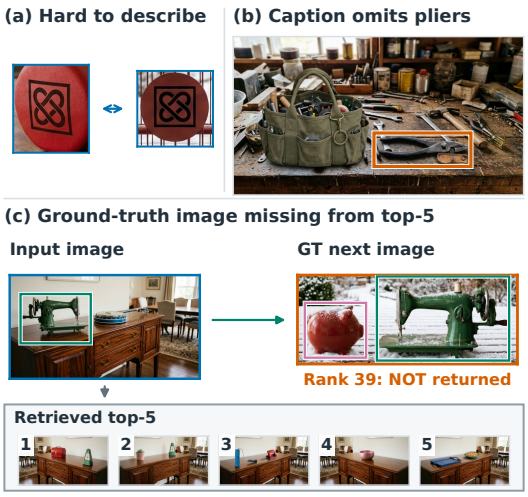


Figure 2: **Visual retrieval bottlenecks.** (a) Exact symbols can be hard to describe. (b) The caption mentions "metal tools" but does not name the boxed pliers. (c) Following the green sewing machine requires the ground-truth next image. The retriever ranks it 39th, so the red piggy bank needed for the following hop is not shown.

To train and evaluate multi-step embedding models, we introduce VHOP, a flexible data generation framework for visual multi-hop image retrieval. It connects images via shared objects and offers controllable hop counts, object distractors, and varying task difficulty levels (Table 1). For reliable training rewards and rigorous evaluation, verifiers use structured records of objects and relations to ensure the ground-truth labels satisfy all detailed query instructions and include all valid answers.

Building on these data and benchmarks, we develop VHOP-ROUTER, an end-to-end training framework that transforms a standard single-step embedding model into an autoregressive multi-step retriever. Training combines supervised fine-tuning (SFT), online imitation learning (Online IL), and reinforcement learning with verifiable rewards (RLVR). At inference, VHOP-ROUTER autoregressively retrieves images and returns a linked image chain in a single tool call based on the initial query and completed hops, without agent text queries for intermediate hops (Figure 3). Because LLM agents remain completely untouched, VHOP-ROUTER is compatible with any LLM-based agent and preserves all native LLM capabilities.

Table 1: **Comparison with related QA and visual-retrieval benchmarks.** I+T denotes image(s) and text. Visual hop clues reveal the next target in an intermediate image; planning involves avoiding or recovering from dead ends; symbolic verification checks answers against structured facts; training recipe indicates an evaluated solver-training procedure.

Benchmark	Query	Output	Visual hop clues	Planning ability	Symbolic verification	Difficulty controls	Training recipe
HotpotQA	Text	Text	–	–	–	–	✓
MuSiQue	Text	Text	–	–	–	–	✓
PhantomWiki	Text	Text	–	–	✓	Depth, size	–
WebQA	Text	Text	–	–	–	–	✓
Encyclopedic-VQA	I+T	Text	–	–	–	–	–
Visual Haystacks	Text	Yes/no	–	–	–	Set size	✓
MuirBench	I+T	Option	–	–	–	–	–
CLEVR	I+T	Text	–	–	✓	Scene/question complexity	✓
ILIAS	Image	Images	–	–	–	–	✓
Ego4D-VQ	Image	Track	–	–	–	–	✓
VHOP (ours)	I+T	Image	✓	✓	✓	Hops, levels, corpus size	✓

Through extensive experiments, we demonstrate that VHOP-ROUTER models achieve significant performance gains in both retrieval-only and agentic search tasks while maintaining superior efficiency. Furthermore, these models generalize well to unseen tasks across varying difficulty levels and hop counts, as well as to realistic test sets, ensuring the practical utility of the trained multi-step embedders.

Our contributions are four-fold:

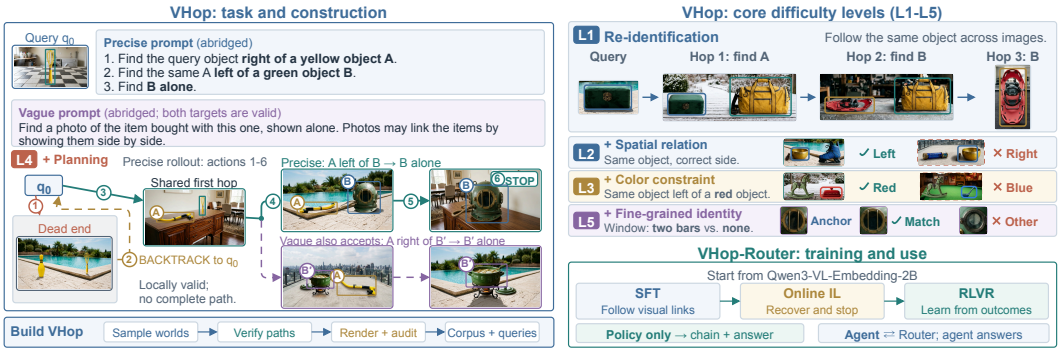


Figure 3: **Overview of VHOP and VHOP-ROUTER. Left:** An L4 search example with precise and vague prompts, dead ends, and backtracking, followed by benchmark construction. **Right:** A complete L1 retrieval chain and small image comparisons for L2, L3, and L5, with the VHOP-ROUTER training stages below. Boxes, arrows, and detail crops are reader annotations. Examples of all difficulty levels appear in Figure 8 in Appendix B.

- We formulate *multi-hop image retrieval in visual latent space*, a novel task where an embedding model navigates independently in the latent embedding space to achieve multi-hop image retrieval.
- We introduce VHOP, a flexible and controllable data generation framework and benchmark. It provides verified ground-truth trajectories across varying, easily extendable difficulty levels.
- We develop VHOP-ROUTER, an end-to-end framework for training autoregressive multi-step embedding models.
- We demonstrate that offloading multi-step search from the agent to the retriever offers a highly effective and efficient solution for overcoming the bottleneck in agentic search, improving the success rate while significantly reducing token length and number of images in the LLM context.

2 RELATED WORK

Multi-hop Dense Retrieval (Xiong et al., 2020) and Q-RAG (Sorokin et al., 2026) train retrievers for multi-step text search. VHOP-ROUTER shares this focus on learning the retrieval tool, but learns visual matching and recovery from dead ends. IRCoT (Trivedi et al., 2023) and Search-R1 (Jin et al., 2025) use LLM reasoning to guide retrieval; our policy retrieves image chains directly in latent space without generating intermediate text queries or fine-tuning LLMs. ILIAS (Kordopatis-Zilos et al., 2025) and Ego4D's visual queries (Grauman et al., 2022) search for a given object, whereas VHOP requires discovering the next target from intermediate images. WebQA (Chang et al., 2022) and Visual Haystacks (Wu et al., 2025) evaluate question answering with visual evidence; our task focuses on retrieving the answer image through a chain of visual clues. Finally, CLEVR (Johnson et al., 2017) and PhantomWiki (Gong et al., 2025) control reasoning within scenes and text corpora, respectively; VHOP controls retrieval paths across images, including hop counts and dead ends, and provides verified answers. Appendix A provides further discussion of these methods, benchmarks, and training approaches.

3 VHOP: CONTROLLABLE MULTI-HOP VISUAL SEARCH IN LATENT SPACE

Consider that in real life, photo albums capture everyday moments and can help us locate misplaced objects. A multi-hop retriever can follow visual clues across these photos to find an image showing where an object was last seen, as illustrated in Figure 6. We formalize this search process through VHOP, a controlled setting for studying multi-hop visual retrieval.

Task definition. Given a starting image and text instruction, the solver retrieves a sequence of images from a corpus, using each intermediate image to determine what to search for next. Each retrieval is a *hop*, and the final retrieved image is the answer. In Figure 3, the solver first retrieves an image in which the starting object appears to the right of a yellow object A, and then uses A as the visual clue for the next hop. We consider two types of prompts. *Precise prompts* specify each step using color and spatial constraints without directly naming the objects, providing a controlled setting

for evaluating visual retrieval under explicit guidance. *Vague prompts* omit these constraints and step-by-step instructions, reflecting settings in which users cannot specify the intermediate retrieval steps. We train and evaluate both settings; see details in Section 4.1.

Controllable difficulty levels. VHOP provides five core difficulty levels (L1–L5; Figure 3), each retaining earlier requirements while adding constraints or distractors. L1–L5 link images through the same physical object, with L4 introducing dead ends that require exploration and recovery. In Figure 3, both a yellow tool and a yellow bowling pin appear to the left of the starting object, satisfying the first step, but only the tool permits a valid continuation. A solver choosing the bowling pin must therefore backtrack and try another candidate. The hop count, corpus size, and number of distractors are also configurable. Matching can also be extended to logos and printed text; Appendix F.4 describes these extensions and reports their results.

Symbolic generation, rendering, and verification. We first construct each image collection symbolically, specifying the object pool, object appearances and attributes, co-occurrence and spatial relations, the answer sequence, and distractors. For precise instructions, an independent verifier checks that exactly one sequence satisfies the query across the entire symbolic collection. If an instruction leaves relations unspecified (vague prompt), the verifier records all valid answers, any of which is accepted during evaluation. We then render the images and check the consistency of object identities, colors, and spatial relations. Training and evaluation use disjoint sets of object appearances. Appendices B and C provide generation details and formal guarantees, including trajectory uniqueness for precise instructions; Section 4.1 describes scoring rules and search budgets.

4 VHOP-ROUTER: TRAINING FRAMEWORK

We developed VHOP-ROUTER, an end-to-end training framework that enables embedding models to navigate in visual latent spaces autoregressively through a three-stage process: Supervised fine-tuning (SFT) teaches valid retrieval chains, online imitation learning (Online IL) teaches recovery and stopping, and reinforcement learning with verifiable rewards (RLVR) optimizes final-answer success. The trained policy can search independently and return image chains to VLM agents.

4.1 EXPERIMENTAL SETUP

Training Data. All policy-training data are generated at L4, the first level with locally valid but globally incorrect branches, requiring exploration and backtracking. Data are organized into *worlds*: a world is one independently generated image corpus together with the queries posed over it, and every query is answered using only the corpus of its own world (Appendix B). SFT/Online IL, RLVR, and evaluation use mutually disjoint worlds with distinct objects, backgrounds, and queries. Gold trajectories supervise SFT and Online IL and are withheld for RLVR (Table 2).

Table 2: **Data allocation.** Query counts use precise three-hop prompts.

Split	Worlds	Queries/ world	Images/ world	Gold trajectories
SFT/ Online IL	1	978	4,492	Yes
RLVR	10	91–100	445–450	No

Evaluation setting. We train VHOP-ROUTER exclusively on L4 and evaluate it on L1–L5 without further fine-tuning. Each non-L4 level contains approximately 100 three-hop queries: L1–L3 test transfer to simpler settings, while L5 tests transfer with visually similar object instances. For L4, we use three independent evaluation worlds with 100 queries each for a more robust assessment. All evaluation worlds are disjoint from training, and non-L4 results measure whether the retrieval, backtracking, and stopping behaviors learned on L4 generalize across difficulty levels, corpora, and visual conditions. Table 3 summarizes the statistics.

Table 3: **Evaluation sets.**

Level	Queries/ world	Images/ world
L1	99	300
L2	98	250
L3	97	350
L4	100	450
L5	100	550

Baselines. We consider two types of baselines. *(1) Retriever-only methods* search without an LLM agent. We evaluate the open-source Qwen3-VL-Embedding-2B (Qwen3) and the proprietary Gemini Embedding 2 (GE2) (Shanbhogue et al., 2026) under three strategies. *Single-shot* retrieval

encodes the original multimodal query and returns the top-5 images. *Greedy* retrieval combines the original instruction with the latest retrieved image to select one new image at each of five steps. *History-based* retrieval instead encodes the complete retrieval history. Caption-based variants use cached Gemini-3.1-Flash-Lite captions to test whether textual descriptions preserve the visual clues needed for multi-hop retrieval.

(2) Agentic-search methods use an LLM to set up tool calls with observations (Yao et al., 2022). We use Gemini 3.5 Flash with GE2 or Qwen3: the agent inspects the returned images and formulates its next query. To isolate the effect of the retrieval tool, we keep the agent fixed and replace its retriever with VHOP-ROUTER, which returns an image chain for the agent to inspect before answering or continuing the search. We also evaluate stronger agentic baselines by replacing Flash with Gemini 3.1 Pro (Section 5.2) or returning up to 50 images per retrieval call (Section 5.3).

Metrics. We evaluate both *task performance* and *search efficiency*. Task performance is measured by success rate. Embedding and caption-based baselines succeed if a valid answer appears among the five retrieved images. Standalone VHOP-ROUTER succeeds if it stops with a valid answer at the top of the stack, while agentic systems succeed if their declared final answer is valid. For efficiency, we report the total number of generated tokens, image inputs, and API calls per query. Agentic search allows up to 16 tool calls and 8,000 output tokens per model call. Each VHOP-ROUTER rollout allows up to 16 actions; retrieval, backtracking, and stopping each count as one action.

4.2 AUTOREGRESSIVE RETRIEVAL POLICY

State and actions. Let x be the text instruction, q_0 the query image, and $\mathcal{C}$ the image corpus. At decision t, the state consists of x and the active stack $\sigma_t = (q_0, q_1, \ldots, q_{d_t})$, where $q_1, \ldots, q_{d_t}$ are the images retained after any backtracking and d_t is the current hop count. The action space is $\{\textsc{Select}(a) : a \in \mathcal{C}\} \cup \{\textsc{Backtrack}, \textsc{Stop}\}$. $\textsc{Select}(a)$ appends image a to the stack; BACKTRACK removes its last retrieved image; STOP ends the search and returns its current image stack. The query image q_0 always remains in the active stack.

SELECT A state encoder E_s jointly encodes the instruction x and active stack σ_t as s_t, while an action encoder E_a independently encodes each candidate image a:

$$s_t = E_s(x, \sigma_t), \qquad e_a = E_a(a), \qquad \|s_t\|_2 = \|e_a\|_2 = 1. \tag{1}$$

Both encoders start from Qwen3-VL-Embedding-2B and use LoRA (Hu et al., 2021). We freeze E_a after SFT and reuse its cached corpus embeddings during Online IL, RLVR, and inference. Candidates are scored by cosine similarity, $z_t(a) = s_t^\top e_a$. After masking ineligible images, we select from the top-K candidates $\mathcal{C}_t$ using a softmax with temperature τ:

$$\pi_\theta^{\text{sel}}(a \mid s_t, \mathcal{C}_t) = \frac{\exp(z_t(a)/\tau)}{\sum_{a' \in \mathcal{C}_t} \exp(z_t(a')/\tau)}. \tag{2}$$

BACKTRACK and STOP. Two learned gates control stopping and backtracking. Each is a two-layer MLP over $[s_t; c_t]$, where $c_t \in \mathbb{R}^4$ contains four retrieval statistics, such as the maximum candidate score (Appendix D.3). During stochastic rollouts, the gate probabilities are

$$p_t^{\text{stop}} = \text{sigmoid}(g_{\text{stop}}([s_t; c_t])/\tau_g), \qquad p_t^{\text{bt}} = \text{sigmoid}(g_{\text{bt}}([s_t; c_t])/\tau_g), \tag{3}$$

Here, τ_g is the gate temperature. Write $z_t^{\text{stop}} = \mathbf{1}[a_t = \textsc{Stop}]$ and $z_t^{\text{bt}} = \mathbf{1}[a_t = \textsc{Backtrack}]$ for the two action indicators. The full policy distribution is

$$\pi_\theta(a_t \mid s_t) = \left(p_t^{\text{stop}}\right)^{z_t^{\text{stop}}} \Big[\left(1 - p_t^{\text{stop}}\right)\left(p_t^{\text{bt}}\right)^{z_t^{\text{bt}}} \Big(\left(1 - p_t^{\text{bt}}\right) \pi_\theta^{\text{sel}}(a \mid s_t, \mathcal{C}_t)\Big)^{1 - z_t^{\text{bt}}}\Big]^{1 - z_t^{\text{stop}}}. \tag{4}$$

At evaluation time, the policy acts deterministically: it stops if $p_t^{\text{stop}} \geq 0.5$, otherwise backtracks if $p_t^{\text{bt}} \geq 0.5$, and otherwise selects the highest-scoring image.

A rollout $\rho = (a_0, \ldots, a_T)$ records all decisions, including selections later undone by backtracking. Its prefix $\rho_{<t}$ determines the active stack σ_t before decision t. Its log-probability under π_θ is

$$\log \pi_\theta(\rho \mid x, q_0) = \sum_{t=0}^{T} \log \pi_\theta(a_t \mid s_t). \tag{5}$$

4.3 TRAINING PIPELINE

The three stages address successive requirements of sequential retrieval: ranking the correct continuation, recovering from mistakes, and optimizing final-answer success.

4.3.1 SUPERVISED FINE-TUNING

SFT teaches next-hop retrieval from gold trajectory prefixes. For a gold trajectory $g_{1:H}$ and hop $k \in \{1, \ldots, H\}$, the state encoder reads the instruction x, query image q_0, and prefix $g_{1:k-1}$. The InfoNCE objective (Oord et al., 2018) aligns the resulting state with the next gold image g_k:

$$s_{k-1} = E_s(x, (q_0, g_{1:k-1}))\,, \qquad \mathcal{L}_k = -\log \frac{\exp(\langle s_{k-1}, e_{g_k} \rangle / \tau)}{\sum_{a \in \mathcal{C}} \exp(\langle s_{k-1}, e_a \rangle / \tau)}. \tag{6}$$

Training proceeds in two phases: we first freeze E_a and update only E_s, then jointly optimize both encoders. This warm-up stabilizes training and prevents collapse (Figure 9).

4.3.2 ONLINE IMITATION LEARNING FOR RECOVERY AND STOPPING

Online IL trains on states visited during search to teach recovery and stopping (Ross et al., 2011). Each iteration collects fresh rollouts using the current policy, oracle guidance, and dead-end exploration. For each prefix $\rho_{<t}$, we recover the active stack σ_t, encode it as s_t, and obtain the oracle action $a_t^\star$. We freeze E_a and update E_s and the gates using the per-state loss in Eq. 7. Only states collected in the current iteration are used for the update. Algorithm 1 summarizes this procedure.

For oracle labels $y_t^{\mathrm{bt}} = \mathbf{1}[a_t^\star = \textsc{Backtrack}]$ and $y_t^{\mathrm{stop}} = \mathbf{1}[a_t^\star = \textsc{Stop}]$, the loss is

$$\ell_t = -(1 - y_t^{\mathrm{bt}})(1 - y_t^{\mathrm{stop}}) \log \pi_\theta^{\mathrm{sel}}(g_t \mid s_t, \mathcal{C}_t) + B_t(y_t^{\mathrm{bt}}) + S_t(y_t^{\mathrm{stop}}). \tag{7}$$

The selection term applies only when the oracle selects image g_t, which is added to $\mathcal{C}_t$ if absent. The terms $B_t(y)$ and $S_t(y)$ are binary cross-entropy losses for the backtrack and stop gates (Eq. 3), with target $y \in \{0, 1\}$. Appendix D specifies their weights and action masks.

4.3.3 OUTCOME-BASED REINFORCEMENT LEARNING

Starting from the Online IL checkpoint, RLVR trains E_s and the two gate MLPs on the separate RLVR data in Table 2, with E_a fixed. For each query, we sample G rollouts ρ_g and assign $r_g = 1$ only when the policy stops with a valid answer, and 0 otherwise. Each rollout receives a leave-one-out advantage A_g (Ahmadian et al., 2024). We use a PPO-style clipped surrogate (Schulman et al., 2017) with DAPO-style asymmetric clipping (Yu et al., 2026). Algorithm 3 summarizes training, and Figure 10 shows its dynamics. Appendix D.4 details the loss terms, clipping, and KL penalties.

Algorithm 1 Online IL training

Require: SFT checkpoint; oracle $\pi^\star$
1: Freeze action encoder E_a
2: **for** each training iteration **do**
3: Collect fresh rollouts ρ
4: **for** each rollout prefix $\rho_{<t}$ **do**
5: Replay $\rho_{<t}$ to recover σ_t
6: Encode $s_t = E_s(x, \sigma_t)$
7: Obtain $a_t^\star = \pi^\star(x, \sigma_t)$
8: **end for**
9: Update E_s and gates using ℓ_t
10: **end for**

5 RESULTS

We evaluate combinations of four independent configurations. First, *task difficulty* tests capability across query complexities. Second, we vary the *number of hops*. Third, we compare *precise prompts*—which give detailed instructions and constrain valid answers—against *vague prompts* that omit step-by-step guidance to mimic real-world searches. Finally, we train with *fixed* or *mixed* hop counts. By default, training uses L4 (requiring planning and navigation around dead ends), 1–3 mixed hops, and precise prompts. We evaluate L1–L5 and ablate settings to test generalization, alongside a test set reflecting realistic user scenarios.

5.1 SIGNIFICANT IMPROVEMENTS ACROSS DIFFICULTY LEVELS

Table 4 reports three-hop results across L1–L5 for standalone retrieval and agentic search. We train VHOP-ROUTER with precise or vague prompts, using either three-hop-only or mixed-hop data. On

Table 4: **Three-hop success rate (%) across L1–L5.** Left: standalone retrieval. Right: Gemini-3.5-Flash with different tools. Blocks (a) and (b) use precise and vague prompts, respectively. Shaded columns show VHOP-ROUTER with three-hop-only and mixed-hop training; other baselines are shared across both regimes. Within each row, **bold** and underlining mark the best and second-best scores on each side of the dashed rule; ties share a rank.

	Retriever only														Agentic search			
	Embedding retrieval						Caption → embedding						VHOP-ROUTER		GE2	Qwen3	+ VHOP-ROUTER	
	GE2			Qwen3			GE2			Qwen3			3-hop	mixed			3-hop	mixed
Level	1-shot	Greedy	Hist.	1-shot	Greedy	Hist.	1-shot	Greedy	Hist.	1-shot	Greedy	Hist.						
(a) Precise prompt																		
L1 (re-ID)	2.0	6.1	2.0	5.1	9.1	6.1	1.0	0.0	1.0	6.1	4.0	3.0	**66.7**	61.6	60.6	73.7	**85.9**	80.8
L2 (spatial)	4.1	5.1	4.1	4.1	8.2	4.1	1.0	2.0	1.0	6.1	2.0	3.1	43.9	**59.2**	58.2	69.4	62.2	**80.6**
L3 (color)	3.1	5.2	2.1	3.1	7.2	2.1	1.0	0.0	0.0	0.0	1.0	0.0	44.3	**85.6**	56.7	77.3	58.8	**89.7**
L4 (dead ends)	1.3	3.7	1.7	1.0	3.3	1.0	0.0	0.0	0.0	0.7	1.0	0.3	48.0	**76.3**	19.3	32.7	59.7	**85.4**
L5 (twins)	1.0	2.0	2.0	3.0	4.0	2.0	0.0	0.0	0.0	1.0	1.0	1.0	42.0	**72.0**	15.0	28.0	57.0	**84.0**
(b) Vague prompt																		
L1 (re-ID)	1.0	1.0	1.0	3.0	5.1	2.0	0.0	0.0	1.0	3.0	2.0	2.0	**46.5**	41.4	11.1	9.1	**67.7**	45.5
L2 (spatial)	2.0	4.1	6.1	0.0	8.2	2.0	2.0	2.0	2.0	0.0	2.0	2.0	**55.1**	36.7	6.1	0.0	**77.6**	51.0
L3 (color)	0.0	6.2	4.2	4.2	6.2	4.2	2.1	4.2	0.0	0.0	0.0	0.0	**56.2**	29.2	6.2	6.2	**72.9**	33.3
L4 (dead ends)	2.0	1.3	1.3	0.7	0.0	0.7	0.0	2.0	0.0	0.0	0.0	0.0	38.0	**42.0**	6.0	2.0	**55.3**	50.7
L5 (twins)	4.0	2.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**42.0**	24.0	0.0	2.0	**50.0**	32.0

all these four training configurations, VHOP-ROUTER improved retrieval and agentic search across 5 difficulty levels significantly over baseline retrievers.

With our default mixed-hop training with precise prompts, standalone VHOP-ROUTER reaches **76.3%** success on L4, compared with **3.7%** for the strongest standard single-step retriever. It also improves agentic search: replacing Qwen3 with VHOP-ROUTER while keeping the agent fixed as Gemini 3.5 Flash raises success from **32.7% to 85.4% at in-domain L4 difficulty** and from **28.0% to 84.0% on L5 with zero-shot transfer**.

5.2 RETRIEVER UPGRADES YIELD 14X THE GAIN OF AGENT UPGRADES

Table 5: **Upgrading agent vs. upgrading tool**

	Qwen3		VHOP-ROUTER
Level	Flash	Pro	Flash
L1	73.7	78.8	**80.8**
L2	69.4	77.6	**80.6**
L3	77.3	**90.7**	89.7
L4	32.7	36.3	**85.4**
L5	28.0	34.0	**84.0**

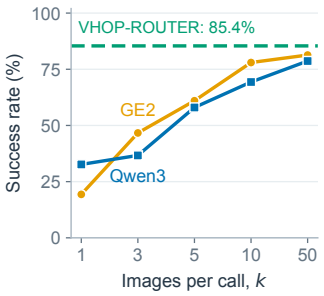
On L4 and L5, upgrading the tool yields larger gains than upgrading the agent. Replacing Flash with Gemini 3.1 Pro while keeping Qwen3 adds only 3.7 percentage points on L4, compared with 52.7 points upgrading tool from Qwen3 to VHOP-ROUTER (Figure 1b, Table 5). On L5, the corresponding gains are 6.0 and 56.0 points. Appendix Figure 11 reports both tool training configurations across all levels.

5.3 CAN RETRIEVING MORE IMAGES MATCH A LEARNED MULTI-STEP RETRIEVER?

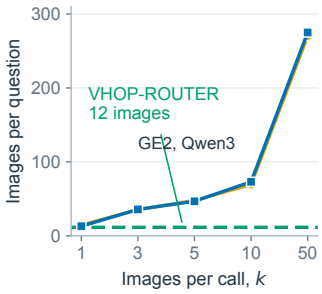
Agentic baselines in Table 4 return only the top-ranked image ($k = 1$) per call. Could a single-step retriever close the gap by returning more candidates for the agent to reason over? We evaluate GE2 and Qwen3 at $k \in \{1, 3, 5, 10, 50\}$, keeping Gemini 3.5 Flash fixed. The L4 test corpus contains 450 images, so $k = 50$ exposes more than one ninth per call. Increasing k improves success, but neither retriever matches VHOP-ROUTER (85.4%) even at $k = 50$ (Figure 4). Visual context also grows rapidly. With Qwen3 at $k = 50$, the agent averages 275 retrieved images in context per question and a cumulative API payload of 1,163 image inputs across turns. In contrast, Flash with VHOP-ROUTER averages 2.1 tool calls, 12 returned images, and a cumulative API payload of 33 images. It thus **outperforms while requiring** $23\times$ **fewer images in context and a** $35\times$ **smaller cumulative API payload** than the $k = 50$ baselines. All results use precise three-hop test queries; Appendix Table 14 includes the three-hop-trained tool and other levels.

5.4 GENERALIZATION ON ONE- AND TWO-HOP QUERIES

We evaluate VHOP-ROUTER on one- and two-hop queries across L1–L5, using the same checkpoints and held-out worlds as in the three-hop evaluation. Under the default precise mixed-hop con-



(a) Success rate.



(b) Images in context.

Retrieval tool	k	Calls	Imgs	Sent
GE2	1	14.5	15	120
	3	11.8	35	249
	5	9.6	48	300
	10	6.9	69	344
	50	5.4	270	1,084
Qwen3	1	13.0	13	104
	3	11.9	36	255
	5	9.3	47	295
	10	7.3	73	392
	50	5.5	275	1,163
VHOP-ROUTER	chain	2.1	12	33

Table 6: Tool call usage.

Figure 4: **Wider retrieval does not outperform VHOP-ROUTER.** Gemini 3.5 Flash uses precise three-hop L4 test queries; VHOP-ROUTER uses precise mixed-hop training. **Left:** Success rate; dashed line: Flash with VHOP-ROUTER. **Middle:** Returned images per question. **Right:** Mean calls, returned images (*Imgs*), and image occurrences across agent requests (*Sent*), per question.

figuration, the standalone VHOP-ROUTER achieves a **99.7%** success rate on one-hop L4 queries, showing strong performance on single-step retrieval. Furthermore, it reaches **90.0%** success on two-hop L4 queries. Together, these results show that one mixed-hop policy handles both shorter query types without further fine-tuning. Comprehensive results for all four training configurations and evaluation levels are detailed in Appendix Tables 11 and 12.

5.5 ROBUSTNESS TO ADDITIONAL DISTRACTOR OBJECTS

In the standard evaluation (Table 4), each hop image contains two foreground objects. To test robustness, we add a third object to every hop image in the L4 test set. These objects introduce competing visual cues without changing the answers. This modification substantially increases retrieval difficulty (Figure 5): success for the strongest standalone policy, VHOP-ROUTER with precise mixed-hop training, drops from 76.3% to 52.2%. However, it still exceeds the strongest agentic baseline with a single-step retriever (27.4%) by 24.8 percentage points, retaining a substantial advantage under this unseen visual shift.



(a) Standard two-object hop image.



(b) The same setting with an added distractor.

Table 7: **L4 answer accuracy (%).** VHOP-ROUTER policy alone.

Method	Training configuration	2 objects	3 objects
VHOP-ROUTER	Precise, 3-hop	48.0	30.4
	Precise, mixed-hop	**76.3**	**52.2**
	Vague, 3-hop	38.0	22.0
	Vague, mixed-hop	42.0	17.3
Flash + GE2	-	19.3	16.4
Flash + Qwen3	-	32.7	27.4

Figure 5: **Robustness to a foreground distractor. Left:** A standard VHOP image. **Middle:** A variant with a third, unseen distractor. **Right:** Retrieval success with distractor.

5.6 GENERALIZATION TO REALISTIC QUERIES AND SETTINGS

All preceding experiments use procedurally generated images and prompts. To test the generalization capability in real-world scenarios, we curate another test set with realistic queries and settings including 50 questions and 386 images. Annotators write the questions and specify everyday scenes. The images are re-rendered to remove privacy-sensitive details while preserving the intended content. Each query requires following visual clues to locate a target item, with an image of the corresponding box serving as the final answer (Figure 6).

Table 8: **Task success on realistic queries.** Answer found among returned images (%).

Agentic Search Method	Training configuration	Success rate
Flash + VHOP-ROUTER	Precise, 3-hop	**36.0**
	Precise, mixed-hop	34.0
Flash + GE2	-	22.0
Flash + Qwen3	-	26.0

Without additional training, Flash with VHOP-ROUTER retrieves the correct answer among its returned images on **36.0%** of queries with precise three-hop training and **34.0%** with precise mixed-

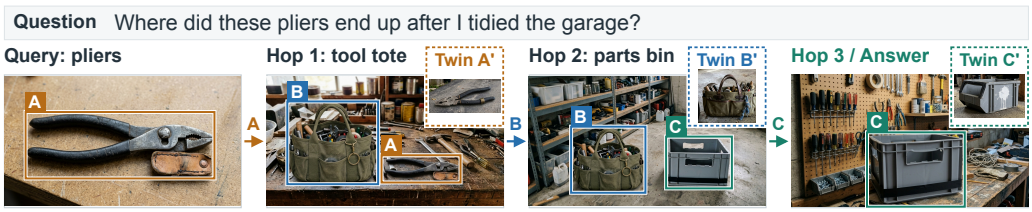


Figure 6: **A realistic retrieval task.** Dashed insets show corpus distractors depicting the same object types but different physical instances. Zoom in for details.

hop training, compared with **22.0%** for Flash with GE2 and **26.0%** with Qwen3 (Table 8). Full results, including standalone VHOP-ROUTER, appear in Appendix F.3.

6 ANALYSIS

6.1 CONTROLLING ROLLOUT LENGTH IN RLVR

The default RLVR objective rewards only answer correctness, without a search cost (Section 4.3.3). Training thus lengthens rollouts and increases budget exhaustion (Figure 10). To test whether a length penalty shortens rollouts, we anneal the terminal reward as $r_\lambda = r - \lambda|\rho|$, $\lambda \in \{0, 0.02, 0.05\}$.

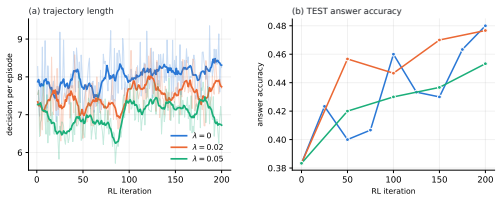


Figure 7: **RLVR length-penalty dynamics. (a)** Mean $|\rho|$. **(b)** L4 test success rate.

Increasing λ consistently shortens rollouts (Figure 7). On the L4 test set, $\lambda = 0.02$ achieves accuracy comparable to the unpenalized objective, whereas $\lambda = 0.05$ degrades performance. A modest penalty thus removes unnecessary actions without sacrificing accuracy; a stronger penalty affects the accuracy reward. Further results appear in Appendix E.4 and Table 15.

6.2 TRAJECTORY PATTERN ANALYSIS

Representative trajectories in Appendix G illustrate the contributions of each training stage and the agent. **SFT** improves visual matching, retrieving the correct first hop where the untrained model fails, but can still follow an incorrect spatial relation. **Online IL** learns backtracking and stopping but may stop on a wrong branch. **RLVR** refines these decisions through final-answer rewards, recovering from the same initial mistake as Online IL, completing the correct chain, and stopping at the answer (Figure 13). The agent further complements VHOP-ROUTER through **answer selection** and **re-querying**: it selects the correct answer from the returned chain when VHOP-ROUTER continues searching until its budget runs out (Figure 19), or revises the query and calls the same policy again when the answer is missing, recovering the complete correct image chain (Figure 18).

7 CONCLUSION AND LIMITATIONS

We address visual agentic search bottlenecks by offloading multi-hop search directly to the retrieval tool. We introduce VHOP, a benchmark and data generation framework, alongside VHOP-ROUTER, an end-to-end training pipeline for autoregressive multi-step embedders. VHOP-ROUTER significantly boosts retrieval and agentic search success rates while drastically reducing LLM context sizes and API payloads and preserving all native LLM capabilities. Scaling the data generation framework to more objects remains challenging because it relies on explicitly defined relationships. The trained policy also struggles to generalize beyond the hop counts seen during training. Future work will focus on scaling both VHOP and VHOP-ROUTER and incorporating real-world datasets.

Reproducibility statement

The stdlib-only generator and verifier are deterministic from stored seeds and θ. Appendix B specifies rendering checks, data splits, and prompts; Appendix D gives training procedures and hyperparameters. Section 4.1 defines baselines and metrics.

AI use statement

We used AI tools to assist with manuscript drafting and editing, figure preparation, and reference checking. Generative models were also used to render benchmark images (Appendix B) and produce captions for retrieval baselines (Section 4.1). The authors take responsibility for the final content, including the text, figures, references, and experimental claims.

Ethics statement

This work studies multi-hop visual retrieval using a controlled benchmark with generated images. For the evaluation with human-authored queries, images are re-rendered to remove privacy-sensitive details while preserving the intended content. Generated images may reflect biases of the rendering models and may not fully represent real-world photo collections. When applied to personal images, multi-hop retrieval could reveal sensitive information by linking visual clues across images. Such applications should obtain appropriate consent, enforce access controls, and protect personal information.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Yingshan Chang, Mridu Narang, Hisami Suzuki, Guihong Cao, Jianfeng Gao, and Yonatan Bisk. Webqa: Multihop and multimodal qa. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16495–16504, 2022.

Albert Gong, Kamilė Stankevičiūtė, Chao Wan, Anmol Kabra, Raphael Thesmar, Johann Lee, Julius Klenke, Carla P Gomes, and Kilian Q Weinberger. Phantomwiki: On-demand datasets for reasoning and retrieval evaluation. *arXiv preprint arXiv:2502.20377*, 2025.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18995–19012, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6693–6702. IEEE, 2019.

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.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017.

Giorgos Kordopatis-Zilos, Vladan Stojnić, Anna Manko, Pavel Suma, Nikolaos-Antonios Ypsilantis, Nikos Efthymiadis, Zakaria Laskar, Jiri Matas, Ondrej Chum, and Giorgos Tolias. Ilias: Instance-level image retrieval at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14777–14787, 2025.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, et al. Qwen3-vl-embedding and qwen3-vl-reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. *arXiv preprint arXiv:2601.04720*, 2026.

Thomas Mensink, Jasper Uijlings, Lluis Castrejon, Arushi Goel, Felipe Cadar, Howard Zhou, Fei Sha, André Araujo, and Vittorio Ferrari. Encyclopedic vqa: Visual questions about detailed properties of fine-grained categories. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3090–3101. IEEE, 2023.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

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

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Madhuri Shanbhogue, Zhe Li, Shanfeng Zhang, Gustavo Hernández Ábrego, Shih-Cheng Huang, Aashi Jain, Daniel Salz, Sonam Goenka, Chaitra Hegde, Ji Ma, et al. Gemini embedding 2: A native multimodal embedding model from gemini. *arXiv preprint arXiv:2605.27295*, 2026.

Artyom Sorokin, Nazar Buzun, Aleksandr Anokhin, Egor Vedernikov, Petr Anokhin, Mikhail Burtsev, and Evgeny Burnaev. Q-rag: Long context multi-step retrieval via value-based embedder training. In *International Conference on Learning Representations*, volume 2026, pp. 32481–32506, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 2023.

Fei Wang, Xingyu Fu, James Y Huang, Zekun Li, Qin Liu, Xiaogeng Liu, Mingyu Derek Ma, Nan Xu, Wenxuan Zhou, Kai Zhang, et al. Muirbench: A comprehensive benchmark for robust multi-image understanding. In *International Conference on Learning Representations*, volume 2025, pp. 62624–62650, 2025.

Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. On the theoretical limitations of embedding-based retrieval. In *International Conference on Learning Representations*, volume 2026, pp. 11202–11216, 2026.

Tsung-Han Wu, Giscard Biamby, Jerome Quenum, Ritwik Gupta, Joseph E Gonzalez, David Chan, et al. Visual haystacks: A vision-centric needle-in-a-haystack benchmark. In *International Conference on Learning Representations*, volume 2025, pp. 64592–64615, 2025.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, et al. Answering complex open-domain questions with multi-hop dense retrieval. *arXiv preprint arXiv:2009.12756*, 2020.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

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.

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. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

A Additional Related Work

Multi-hop retrieval and question answering. HotpotQA (Yang et al., 2018) requires combining evidence from multiple Wikipedia articles, while MuSiQue (Trivedi et al., 2022) composes single-hop questions into connected reasoning chains and filters out shortcuts. Multi-hop Dense Retrieval (Xiong et al., 2020) learns to retrieve a passage conditioned on earlier evidence. IR-CoT (Trivedi et al., 2023) interleaves retrieval with reasoning so that each informs the next step. VHOP studies a visual form of this dependency: a retrieved image supplies the object or mark needed to find the next image, and the final answer is itself an image.

Visual retrieval and memory. CLIP (Radford et al., 2021) learns a shared image–text embedding space that supports retrieval by similarity. Instance-level benchmarks such as ILIAS (Kordopatis-Zilos et al., 2025) test whether a system can find the same particular object among many distractors. Ego4D's visual-query task (Grauman et al., 2022) uses an object image to locate its last appearance in an egocentric video. VHOP extends such visual search to a chain of linked objects: the starting image may not show the final target, so directly matching that image to the answer is insufficient. The solver must use intermediate images to determine what to retrieve next.

Multimodal and multi-image reasoning. WebQA (Chang et al., 2022) combines evidence retrieval from image or text sources with natural-language answer generation. Encyclopedic-VQA (Mensink et al., 2023) asks detailed questions about visually identified entities and provides a Wikipedia knowledge base. Visual Haystacks (Wu et al., 2025) tests question answering over large image collections, including questions that require evidence from multiple images. Muir-Bench (Wang et al., 2025) covers a broad range of multi-image tasks, including visual retrieval. VHOP focuses on a specific question: can a system repeatedly recover a new visual clue, follow it through a corpus, and recover when a locally plausible match does not lead to an answer? Its generator makes these dependencies and alternative branches explicit for controlled evaluation.

Controlled evaluation. CLEVR (Johnson et al., 2017) uses generated scenes and executable question programs to diagnose compositional visual reasoning. GQA (Hudson & Manning, 2019) generates compositional questions from scene graphs of real images. PhantomWiki (Gong et al., 2025) generates consistent document collections and questions with adjustable reasoning difficulty and corpus size. VHOP follows this approach of making task structure controllable, while placing the evidence across separate images. Its verifier checks satisfying trajectories over the pooled corpus before rendering. The guarantees are conditional on faithful rendering; the additional retrieval bounds also require the solver restrictions and sampling assumptions stated in Appendix C. Separately, Weller et al. (2026) establish limitations imposed by embedding dimension on the relevance sets a single-vector retriever can represent. Our analysis concerns missing intermediate evidence and locally ambiguous branches, rather than embedding dimension.

Agents and learned search policies. ReAct (Yao et al., 2022) interleaves language reasoning with actions and external observations. Search-R1 (Jin et al., 2025) trains language models to issue search queries using outcome-based reinforcement learning. Q-RAG (Sorokin et al., 2026) trains embeddings for multi-step text retrieval through a value-based objective. Our method also adapts retrieval representations. It operates on image sequences and learns selection, backtracking, and stopping from imitation and final-answer rewards. The trained policy can also serve as a tool for an agent that revises queries or selects an answer from returned images.

Imitation learning and outcome supervision. Collecting expert labels on states visited by the learner addresses the distribution shift that motivates DAgger (Ross et al., 2011). Our Online IL stage uses this idea but updates only on the current iteration's states, without accumulating a dataset across iterations; we therefore do not call the algorithm DAgger or invoke its guarantees. The subsequent RLVR stage uses a terminal answer check, a leave-one-out baseline (Ahmadian et al., 2024), and a clipped policy objective (Schulman et al., 2017; Yu et al., 2026). We adapt these components to a visual retrieval policy with explicit selection, backtracking, and stopping decisions.

B Benchmark Generation and Verification

VHop separates task construction from image rendering. The generator first specifies objects, their links across images, and the constraints in each question. A symbolic verifier checks the resulting search problem over the entire corpus. Rendering then turns this specification into images. This separation provides exact task labels while allowing the visual appearance and search difficulty to vary.

B.1 Difficulty Levels and World Construction

Table 9: **Cumulative difficulty levels.** Each level retains the requirements of the preceding levels.

Level	Added requirement	Required behavior
L1	Object identity	Match the same physical object across views.
L2	Spatial relations	Reject an otherwise matching scene with the wrong relation.
L3	Color constraints	Check the color specified in the instruction.
L4	Dead ends	Recover after a plausible branch fails to continue.
L5	Appearance twins	Distinguish instances with similar type, color, and appearance.
L6	Shared logos	Follow links defined by a shared logo.
L7	Shared text	Follow links defined by a shared printed string.

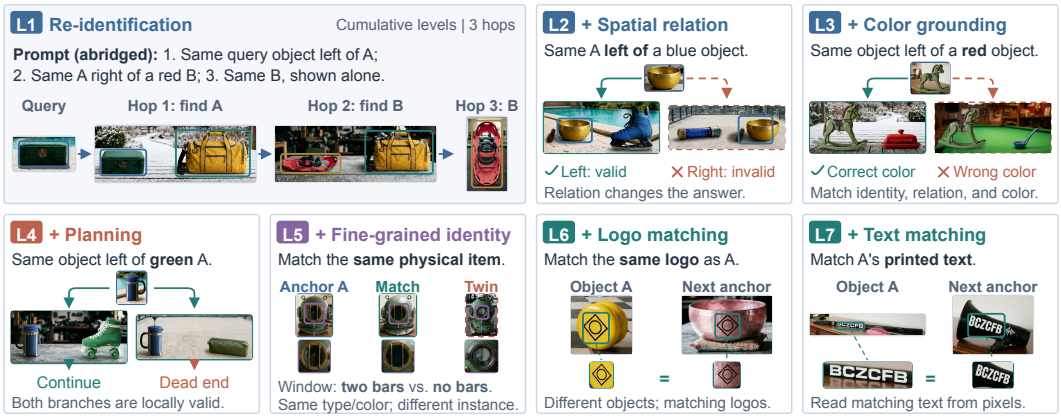


Figure 8: **Visual examples of the difficulty levels.** L1 shows a complete retrieval chain. L2 and L3 add spatial and color constraints; L4 introduces dead ends; and L5 requires distinguishing look-alike objects. L6 and L7 extend the matching rule to logos and printed text. The examples come from separate questions. Boxes, arrows, and enlarged details are reader annotations.

A *world* is one image corpus and its associated questions. A *chain family* contains a gold chain and its related distractors. The generator controls hop count, matching rules, relations, colors, family count, and the numbers of distractors and forks. It assigns fresh matching keys to each family and records every image containing each key. The final portrait is included in the searchable corpus. Query images are fresh views of the starting objects.

Distractors target different errors. Relation and color decoys violate a stated constraint. A dead end satisfies the local match but lacks a valid continuation. An appearance twin resembles the correct object while representing a different instance. Extra portraits share query-specified properties with the answer, preventing those properties alone from identifying it. Images and filenames are shuffled without exposing their roles. The corpus is available from the start; the task requires discovering the links between images, rather than revealing hidden corpus entries.

B.2 Prompt Variants and Data Separation

Precise instructions state the sequence of matching rules, relations, and filters. They refer to an intermediate object through the preceding image, without giving its identity, logo value, or printed string in advance. Vague instructions omit a relation and may admit more than one answer. The verifier enumerates the acceptable endpoints after that omission; evaluation accepts any member of

Algorithm 2 Generating and verifying a VHOP world

Require: Difficulty level, hop count, family count, and object/mark pools
for each chain family **do**
    sample matching rules, spatial relations, and color constraints
    assign fresh keys and construct the gold scenes and final portrait
    add the level's decoys, dead ends, twins, and matching distractor portraits
    construct a precise query without naming the intermediate key values
end for
pool the images and shuffle their identifiers
for each query **do**
    enumerate all satisfying trajectories over the pooled corpus
    check the unique precise path and the intended distractor structure
    enumerate acceptable endpoints for the vague variant, if supported
    reject and regenerate any family that fails verification
end for
render the verified world and check its images against the specification
return image corpus, query images, instructions, and verified labels

this set. Vague variants are reported for L1–L5. One- and two-hop questions are verified suffixes of the three-hop chains and use the same image corpora.

Training and evaluation use separate worlds and separate pools of object appearances and marks. SFT and Online IL use worlds with full trajectory labels. RLVR uses a separate practice pool and obtains its learning signal from the final-answer verifier. Development worlds support model selection; the reported L4 test results use three held-out worlds. World identifiers such as w_0 are local to their split, so a training w_0 and a development w_0 do not denote the same corpus.

B.3 Rendering Conditions

Each rendering prompt specifies the objects, their colors and marks, and their spatial arrangement. Reference images maintain an object's identity or a mark across views. A portrait prompt requests the relevant object alone. The question text is generated separately and does not reveal the intermediate matching values.

The symbolic labels describe the intended scene. For these labels to remain valid in the images, rendering must preserve object identity across views, distinguish different instances where required, reproduce colors and marks, and show the stated relations unambiguously. An image audit checks these properties against the manifest; images with visible violations require regeneration. These conditions are also the assumptions needed to transfer the symbolic guarantees in Appendix C to pixels. The verifier does not, by itself, certify rendering fidelity or rule out every possible captioning shortcut.

C Symbolic Guarantees

This section states what the generator guarantees about its symbolic worlds, and the assumptions needed for the retrieval bounds. A unique symbolic answer does not by itself guarantee that a rendered image is unambiguous. Applying these statements to pixels also requires the rendering conditions in Appendix B.

Definitions. An object's *link key* is its identity, logo, or printed string, depending on the matching rule in the instruction. A scene contains an *anchor* that matches the preceding object and a *witness* that supplies the next key. The witness must satisfy the stated spatial relation and any color constraint. A portrait contains the final object alone. Let H be the total number of retrieval hops, including the final portrait; the three-hop task therefore has two scene hops and one portrait hop.

For a query Q and pooled corpus D, let $\mathrm{Sat}(Q, D)$ be the set of image sequences that satisfy every matching rule, relation, and filter, and end in an eligible portrait. The acceptable answer set $\mathcal{A}(Q, D)$ contains the last image of each such sequence. Precise queries are verified to have one satisfying

sequence. For vague queries, the verifier enumerates the sequences allowed by the omitted relation and accepts all their endpoints.

Theorem 1 (Whole-corpus uniqueness)**.** *Suppose each chain family has fresh link keys, every matched scene has a unique anchor and witness, each non-forked gold prefix has one valid continuation, and each alternative branch at a fork has no satisfying completion. If the terminal key has exactly one eligible portrait, then a precise query has exactly one satisfying trajectory in the pooled corpus.*

Proof. The constructed gold trajectory satisfies the query. Consider a different satisfying trajectory and its first departure from the gold trajectory. Fresh keys prevent a continuation through another family. At a non-forked prefix, continuation uniqueness rules out the departure. At a fork, the alternative has no satisfying completion. A change in the final image is excluded by terminal-portrait uniqueness. Thus no different satisfying trajectory exists. The verifier checks these conditions over the pooled corpus, including decoys and appearance twins. □

Theorem 2 (Reusing appearance templates)**.** *Reusing an object's appearance template across families preserves symbolic uniqueness if link keys remain fresh and the conditions of Theorem 1 remain satisfied.*

Proof. Symbolic matching depends on link keys and scene structure, not on the appearance template used to render an object. Reusing a template therefore does not add an object to a key's matching set. For an identity link, different instances must still be distinguishable in the rendered images. □

Assumptions for direct retrieval. A pairwise scorer assigns $f(Q, z)$ to each candidate image z using only the query and that image. It has no retrieved intermediate evidence, corpus-level features, or metadata that reveals an image's role. Consider n candidate portraits that agree on all answer properties supplied by the query, such as color and matching rule. The required symmetry assumption is that, conditioned on the evidence available to this scorer, each of these portraits is equally likely to be the answer. This requires role-independent sampling of keys and appearances and symmetric handling of collisions. Omitting a key from the query alone is insufficient to establish this symmetry.

Theorem 3 (Direct pairwise answer bound)**.** *Under the conditional symmetry above, if $n \geq m + 1$ portraits share the answer's query-specified properties and at least one scene hop precedes the answer, any pairwise scorer has top-1 answer accuracy at most $1/(m + 1)$, averaged over the sampled worlds.*

Proof. Conditioned on the scorer's available evidence, the answer index is uniform over the n portraits. Any selected index has probability $1/n$ of being correct. Selecting an image outside this set cannot increase success. Averaging over the evidence gives the bound. A solver that retrieves an intermediate image and reads its next key has additional evidence, so this bound does not apply to it. □

Proposition 1 (Completing a chain with a tight retrieval budget)**.** *Consider a policy that follows the query's local transitions and must retrieve a complete H-hop chain using at most H image retrievals. At each fork t, suppose there are d_t incorrect branches and, conditioned on a correct prefix and the policy's available evidence, the correct branch is uniform among the $1 + d_t$ choices. If an incorrect branch cannot complete the query, the probability of completing the chain is at most $\prod_{t:\, d_t>0}(1 + d_t)^{-1}$.*

Proof. The budget leaves no retrievals for testing a wrong branch and returning to the correct one. The conditional probability of choosing correctly at fork t is at most $(1 + d_t)^{-1}$. Applying the chain rule over successive forks gives the product. Unconditional independence between forks is not required. □

This proposition concerns complete-chain retrieval under the stated local information restriction. It does not bound finding the answer anywhere in a returned set, guessing the final portrait directly, or using a larger budget for recovery. Its budget counts image retrievals; the experimental decision

budget also counts BACKTRACK and STOP. The benefit of additional search is measured empirically.

D TRAINING AND EVALUATION DETAILS

D.1 DATA, MODELS, AND EVALUATION

All four training configurations in Table 4 use L4 data and Qwen3-VL-Embedding-2B (Li et al., 2026). They vary the prompt type (precise or vague) and training hop counts (three hops only or a mixture of one, two, and three). The fully labeled seed world contains 978 questions and 4,492 corpus images. SFT and Online IL use its gold trajectories. RLVR uses separate practice worlds and a final-answer verifier; it does not train on their intermediate gold actions. The detailed optimization settings below describe the precise three-hop reference run, which produces `rloo10w10_200`.

For the three-hop results, the precise-prompt sample sizes at L1–L7 are $(99, 98, 97, 300, 100, 94, 95)$; the vague-prompt sizes at L1–L5 are $(99, 49, 48, 150, 50)$. L4 pools three held-out test worlds, while the other levels use development world w_0. Development scores are used for model selection and should not be read as independently held-out estimates. Embedding and caption baselines measure whether an acceptable answer is among their five returned images. For a standalone policy, correctness requires STOP with an acceptable image at the top of the active stack. Agent accuracy checks the returned answer, including the evaluator's fallback if there is no valid final declaration. Finding an answer among retrieved images is a separate metric, defined for the human-authored evaluation in Appendix F.3.

The active stack σ_t contains the query image and the retrieved images that have not been removed by backtracking. Its encoding is $s_t = E_s(x, \sigma_t)$. The rollout ρ records every decision, including selections later undone. Candidate masks and stored parent scores are maintained by the search procedure. Thus a popped image leaves subsequent state-encoder inputs but its earlier decision and training record remain in the rollout. This distinction applies in both Online IL and RLVR.

D.2 SUPERVISED FINE-TUNING

The state and action encoders use separate LoRA adapters over a frozen backbone, with rank 32 and scaling parameter 32. The adapters modify the query, key, and value projections and the MLP projections. Embeddings use last-token pooling and are normalized to unit length, with dimension 2048. The reference SFT recipe uses learning rate 2×10^{-5}, batch size 8 per GPU, and a linear schedule over four epochs.

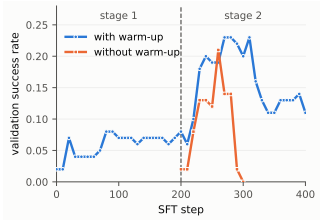


Figure 9: **State-encoder warm-up stabilizes SFT.**

The first stage trains only the state encoder on gold prefixes with full-corpus InfoNCE at temperature 0.05. The next stage introduces hop-level hard negatives, including designed dead ends and constraint violations, and also updates the action encoder. Hard-negative images are re-encoded with gradients and their scores replace the corresponding entries in the cached corpus logits. These re-encoded images provide the action encoder's gradient; the rest of the cached index is treated as constant during an update. Later stages freeze the action encoder and its corpus index.

D.3 ONLINE IMITATION LEARNING

Rollout collection. Online IL runs for 120 iterations with a 14-decision budget per rollout. The oracle selects the next gold image from a valid prefix, backtracks from an incorrect prefix, and stops at the completed chain. At iteration i, the expert-control and forced-excursion probabilities are

$$\beta_i = \max\left(0, 1 - \frac{i}{60}\right), \qquad p_i = 0.6 \max\left(0, 1 - \frac{i}{80}\right). \tag{8}$$

At a valid prefix with an available dead end, the rollout enters a dead end with probability p_i. Otherwise, it follows the oracle with probability β_i and the current policy with probability $1 - \beta_i$.

Every visited state receives an oracle label, regardless of how its action was chosen. Each iteration trains only on the states collected in that iteration; states from earlier iterations are not retained for training. Each record stores the stack, candidate mask, parent maximum score, and oracle action. During the update, we re-encode the recorded stack and recompute the current retrieval scores and c_t, retaining the recorded mask and parent score. A later backtrack does not remove earlier records.

Online IL losses and gradients. Selection uses the top-8 eligible candidates, with the expert's next gold image appended if absent, and a softmax temperature of 0.05. The gate losses in Eq. 7 are

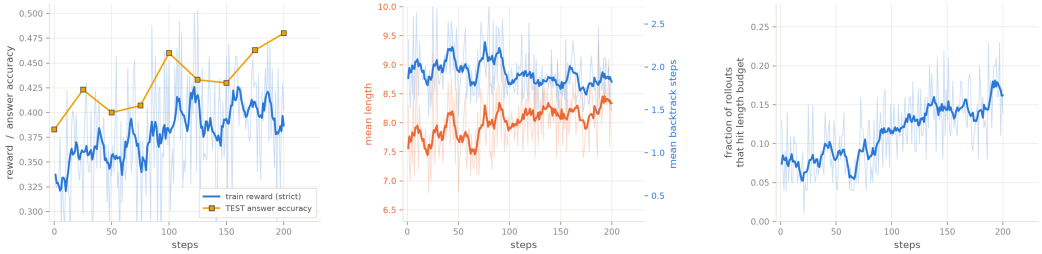
$$B_t(y) = -w_+ y \log p_t^{\mathrm{bt}} - (1-y)\log(1-p_t^{\mathrm{bt}}),$$
$$S_t(y) = -y \log p_t^{\mathrm{stop}} - (1-y)\log(1-p_t^{\mathrm{stop}}), \tag{9}$$

where $w_+ = 4$ compensates for the relative scarcity of backtrack targets. The gate probabilities use Eq. (3) with $\tau_g = 1$. We set $B_t(y) = 0$ at the root and $S_t(y) = 0$ wherever stopping is disabled. For the procedural evaluations, three-hop-only checkpoints permit stopping once the stack depth reaches the question's hop count; mixed-hop checkpoints permit it after any retrieval. This depth floor is an evaluation constraint and should be kept distinct from the learned stop decision. The per-state losses are averaged over the collected dataset:

$$\mathcal{L}_{\text{Online IL}} = \frac{1}{|\mathcal{D}_i|} \sum_{t=1}^{|\mathcal{D}_i|} \ell_t. \tag{10}$$

The action encoder and its corpus index remain frozen. Selection gradients reach the state encoder through the retrieval scores, while gate gradients reach both the corresponding MLP and the state encoder through s_t. The state embedding is not detached on backtrack or stop examples. The four retrieval-context features are treated as constants: the current maximum score, the top-1/top-2 gap, the parent maximum, and their difference.

D.4 Outcome-Based RL Details



(a) Reward and test accuracy (b) Length of ρ and backtracks (c) Truncation rate

Figure 10: **Training dynamics of RLVR.** (a) Training reward and test accuracy. (b) Rollout length and number of backtracks. (c) The fraction of rollouts that exceed budget.

Rewards and advantages. For each input (x, q_0), we sample G rollouts ρ_g from the behavior policy μ, a snapshot of the current policy at the start of the iteration. Let $\mathcal{A}$ be the input's certified answer set: a singleton for a precise question, or the set of valid answers for a vague question. With T_g denoting the final decision index, the terminal verifier returns

$$r_g = \mathbf{1}\left[a_{g,T_g} = \textsc{Stop} \ \wedge \ \mathrm{top}(\sigma_{g,T_g}) \in \mathcal{A}\right]. \tag{11}$$

Each decision in ρ_g receives the same leave-one-out advantage

$$A_g = r_g - \frac{1}{G-1} \sum_{g' \neq g} r_{g'}, \tag{12}$$

where the sum is over the other rollouts for the same input. Groups with identical rewards have zero advantage and contribute no policy-gradient signal.

Algorithm 3 RLVR training

Require: Online IL policy π_θ; terminal answer verifier; frozen action encoder
freeze a reference copy π_0 of the initial policy
for each iteration **do**
  snapshot the current policy as the behavior policy μ
  for each sampled query **do**
    generate G complete rollouts ρ_g using μ
    for every prefix $\rho_{g,<t}$, record its active stack $\sigma_{g,t}$, masks, context, and behavior probabilities
    compute terminal rewards r_g and leave-one-out advantages A_g
  end for
  for each inner update epoch **do**
    re-encode retained stacks as $s_{g,t} = E_s(x, \sigma_{g,t})$; recompute candidate scores and gate probabilities
    update E_s and the gates with clipped branch terms and KL penalties
  end for
end for

Table 10: **Basic RL policy-gradient form.**

Sampled action a_t	Basic term ℓ_t^{PG}
SELECT(a)	$-A_g \log \pi_\theta^{\mathrm{sel}}(a \mid s_t, \mathcal{C}_t)$
BACKTRACK	$-A_g \log p_t^{\mathrm{bt}}$
STOP	$-A_g \log p_t^{\mathrm{stop}}$

Branch-local credit assignment. For decision t in rollout ρ_g, let $p_{g,t}$ denote the sampled-branch probability factor in Table 10, and $\bar{p}_{g,t}$ its stored value under μ. The SELECT and gate probabilities follow Eqs. (2) and (3). We omit the complementary gate factors in Eq. (4): $(1 - p_t^{\mathrm{stop}})(1 - p_t^{\mathrm{bt}})$ for SELECT and $(1 - p_t^{\mathrm{stop}})$ for BACKTRACK. This generally yields a biased surrogate for the full policy gradient obtained from Eq. (5). This choice prevents a SELECT policy-loss term from directly changing the gates through their rejection probabilities.

The branch terms share one objective and optimizer. SELECT gradients reach E_s; BACKTRACK and STOP gradients reach E_s and the respective gate MLP. The policy loss for SELECT has no direct gate-MLP gradient. Shared-encoder updates and KL penalties can still change other branch probabilities; E_a remains fixed.

DAPO-style clipping and negative-advantage weighting. For batch reuse, we use a PPO-style ratio surrogate (Schulman et al., 2017) with DAPO's asymmetric clipping bounds (Yu et al., 2026):

$$\omega_{g,t}(p) = \frac{p}{\bar{p}_{g,t}}, \tag{13}$$

$$\mathcal{J}_{g,t}(p) = -w(A_g) \min\{\omega_{g,t}(p) A_g,\ \mathrm{clip}(\omega_{g,t}(p), 1 - \epsilon_{\mathrm{lo}}, 1 + \epsilon_{\mathrm{hi}})\, A_g\}, \tag{14}$$

with

$$w(A) = \begin{cases} 1, & A \geq 0, \\ \lambda_{\mathrm{neg}}, & A < 0. \end{cases} \tag{15}$$

Setting $\epsilon_{\mathrm{hi}} > \epsilon_{\mathrm{lo}}$ permits larger probability increases for positive-advantage decisions before clipping. The additional weight $0 < \lambda_{\mathrm{neg}} < 1$ reduces the magnitude of negative-advantage terms without reversing their sign. At $p = \bar{p}_{g,t}$, with $w = 1$ and fixed probability support, the surrogate's gradient matches the $-A_g \log p$ form in Table 10.

The implemented policy objective sums the clipped terms:

$$\mathcal{L}_{\mathrm{PG}} = \frac{1}{N_{\mathrm{roll}}} \sum_{\rho_g \in \mathcal{B}} \sum_{t=0}^{T_g} \mathcal{J}_{g,t}(p_{g,t}). \tag{16}$$

Here, $\mathcal{B}$ contains the nonzero-advantage rollouts, while N_{roll} counts all sampled rollouts, including zero-advantage groups. States excluded by the training filters below contribute zero policy loss. During batch reuse, a SELECT term is evaluated only while the sampled image remains in the current top-K set $\mathcal{C}_t$.

RL regularization. The full objective adds behavior and reference KL penalties:

$$\mathcal{L}_{\mathrm{RL}} = \mathcal{L}_{\mathrm{PG}} + \beta_{\mathrm{beh}}\mathcal{K}_{\mathrm{beh}} + \beta_{\mathrm{ref}}(i)\mathcal{K}_{\mathrm{ref}}. \tag{17}$$

The behavior policy μ is the policy that generated the current rollout batch. We record its selection probabilities before updating the model. The behavior penalty sums $\mathrm{KL}(\pi_\theta^{\mathrm{sel}} \,\|\, \mu^{\mathrm{sel}})$ over retained SELECT states and divides by N_{roll}. Both distributions use the candidate set stored during rollout. The stored probabilities remain fixed through all updates on this batch; they are refreshed only when the updated policy collects the next batch. Thus, behavior KL limits changes from the policy at the start of the current iteration, even when the batch is reused for several gradient updates. The reference policy π_0 is a frozen copy of the initial Online IL policy. On a subsample of retained states, $\mathcal{K}_{\mathrm{ref}}$ adds the SELECT KL on the stored candidate set when selection was taken, and Bernoulli KLs for each permitted gate. All KLs use the current policy as their first argument. The reference sum is normalized by the number of reference states, floored at 16. Its weight decreases during training to stabilize early updates while allowing later improvement without gold-trajectory supervision.

RL configuration. The precise three-hop reference run uses eight data-parallel GPUs. At each iteration, each rank samples 4 practice questions and $G = 8$ stochastic rollouts per question, each limited to 16 decisions. The selection temperature is $\tau = 0.05$ and the gate temperature is $\tau_g = 0.5$. Each group stays on one rank, so leave-one-out baselines require no communication. The state-encoder LoRA learning rate is 5×10^{-6} and the gate-MLP rate is 3×10^{-4}. The action encoder and corpus index remain frozen throughout. We reuse each batch for two inner epochs with DAPO-style asymmetric clipping ($\epsilon_{\mathrm{lo}} = 0.2$, $\epsilon_{\mathrm{hi}} = 0.3$) and $\lambda_{\mathrm{neg}} = 0.5$.

Gradient-bearing states are restricted to $|\sigma_t| \leq 9$, including the query image (equivalently, chain depth $d_t \leq 8$), to bound activation memory. If more than 128 eligible states remain per rank, we sample 128 uniformly and scale the policy loss by the original count divided by 128. The behavior-KL weight is $\beta_{\mathrm{beh}} = 0.05$. Reference KL is evaluated on at most 48 retained states per rank, with $\beta_{\mathrm{ref}}(i) = 0.3$ for iterations 1–75 and 0.1 thereafter. The reference run lasts 200 iterations (≈ 14 hours) and yields checkpoint `rloo10w10_200`.

Length-penalty variant. This variant keeps the terminal verifier unchanged but substitutes $\tilde{r}_g = r_g - \lambda n_{\mathrm{decisions}}(\rho_g)$ for r_g in Eq. (12). The decision count includes SELECT, BACKTRACK, and STOP; strict reward remains the evaluation and checkpoint-selection criterion. The reference run uses $\lambda = 0$. The recorded $\lambda = 0.02$ run uses one-state gradient microbatches and raises the gradient-bearing stack limit to 12 images to accommodate the additional failed-rollout states with nonzero advantages. This implementation difference should be considered alongside the reward change. Appendix E.4 examines the effect of charging for each decision.

Roles of the optimization components. The leave-one-out baseline compares outcomes for the same query without normalizing by a small within-group reward standard deviation. Asymmetric clipping limits changes during batch reuse, while the negative-advantage weight reduces the contribution of failed rollouts. Behavior KL constrains updates relative to the policy that collected the batch. Reference KL limits drift from the initial Online IL policy.

E Additional Results and Ablations

E.1 One- and Two-Hop Queries

Tables 11 and 12 report the complete results on verified suffixes of the three-hop queries. The corpus and checkpoints are unchanged. The three-hop-only policies retain the question-specific minimum stop depth; mixed-hop policies can stop after any retrieval. As in the main table, embedding baselines receive credit if the answer is among five results, whereas the standalone policy must stop on one correct answer. This distinction is especially relevant on one-hop queries, where a top-5 list contains five possible answers.

Table 11: **One-hop success rate (%).** Same corpora, checkpoints, and scoring conventions as Table 4. L4 uses test worlds w_1–w_3; other levels use development world w_0. Caption→Qwen3 retains the query image as pixels. In block (b), the agent uses the learned per-step retriever; the other trained-agent blocks use the chain tool. Vague prompts are evaluated on L1–L5. Bold marks the best result within each training block on each side of the dashed rule.

		Retrieval methods													Agentic search		
		Embedding retrieval						Caption → embedding						VHOP-ROUTER	GE2	Qwen3 + VHOP-ROUTER	
		GE2			Qwen3			GE2			Qwen3						
Level	n	1-shot	Greedy	Hist.	1-shot	Greedy	Hist.	1-shot	Greedy	Hist.	1-shot	Greedy	Hist.				
(a) Precise prompt, 3-hop training																	
L1 (re-ID)	99	90.9	77.8	78.8	**98.0**	92.9	96.0	27.3	16.2	18.2	92.9	73.7	73.7	69.7	80.8	74.7	**96.0**
L2 (spatial)	98	92.9	77.6	81.6	**93.9**	89.8	90.8	27.6	10.2	14.3	86.7	64.3	70.4	67.3	81.6	78.6	**95.9**
L3 (color)	97	93.8	79.4	83.5	**96.9**	89.7	93.8	14.4	4.1	7.2	84.5	66.0	70.1	74.2	85.6	82.5	**95.9**
L4 (dead ends)	300	90.7	70.0	73.7	**97.3**	88.0	93.7	14.7	8.0	9.7	86.7	64.0	65.3	67.3	77.7	75.7	**96.7**
L5 (twins)	100	89.0	67.0	76.0	**98.0**	88.0	95.0	14.0	6.0	8.0	86.0	66.0	73.0	59.0	78.0	66.0	**94.0**
L6 (logo)	94	**57.4**	42.6	47.9	55.3	47.9	50.0	6.4	5.3	4.3	56.4	40.4	43.6	28.7	58.5	62.8	**80.9**
L7 (text)	95	**49.5**	37.9	36.8	41.1	31.6	33.7	5.3	1.1	2.1	**49.5**	27.4	29.5	12.6	73.7	75.8	**80.0**
(b) Precise prompt, mixed-hop training																	
L1 (re-ID)	99	90.9	77.8	78.8	**98.0**	92.9	96.0	27.3	16.2	18.2	92.9	73.7	73.7	**98.0**	**80.8**	74.7	69.7
L2 (spatial)	98	92.9	77.6	81.6	93.9	89.8	90.8	27.6	10.2	14.3	86.7	64.3	70.4	**96.9**	**81.6**	78.6	68.4
L3 (color)	97	93.8	79.4	83.5	96.9	89.7	93.8	14.4	4.1	7.2	84.5	66.0	70.1	**97.9**	**85.6**	82.5	78.4
L4 (dead ends)	300	90.7	70.0	73.7	97.3	88.0	93.7	14.7	8.0	9.7	86.7	64.0	65.3	**99.7**	**77.7**	75.7	76.7
L5 (twins)	100	89.0	67.0	76.0	**98.0**	88.0	95.0	14.0	6.0	8.0	86.0	66.0	73.0	96.0	**78.0**	66.0	74.0
L6 (logo)	94	**57.4**	42.6	47.9	55.3	47.9	50.0	6.4	5.3	4.3	56.4	40.4	43.6	50.0	58.5	**62.8**	59.6
L7 (text)	95	**49.5**	37.9	36.8	41.1	31.6	33.7	5.3	1.1	2.1	**49.5**	27.4	29.5	27.4	73.7	**75.8**	74.7
(c) Vague prompt, 3-hop training																	
L1 (re-ID)	99	46.5	21.2	27.3	**73.7**	45.5	55.6	24.2	9.1	10.1	24.2	13.1	15.2	47.5	15.2	14.1	**79.8**
L2 (spatial)	98	56.1	30.6	40.8	**76.5**	59.2	64.3	17.3	9.2	10.2	25.5	13.3	13.3	45.9	32.7	20.4	**90.8**
L3 (color)	96	55.2	28.1	35.4	**65.6**	47.9	55.2	11.5	7.3	7.3	22.9	10.4	12.5	33.3	19.8	15.6	**79.2**
L4 (dead ends)	300	47.0	19.3	26.3	**66.3**	41.3	47.7	13.3	6.7	7.7	18.0	7.7	9.3	26.3	21.0	21.3	**77.0**
L5 (twins)	100	34.0	14.0	19.0	**58.0**	31.0	40.0	10.0	6.0	6.0	13.0	7.0	5.0	31.0	27.0	20.0	**82.0**
(d) Vague prompt, mixed-hop training																	
L1 (re-ID)	99	46.5	21.2	27.3	73.7	45.5	55.6	24.2	9.1	10.1	24.2	13.1	15.2	**87.9**	15.2	14.1	**97.0**
L2 (spatial)	98	56.1	30.6	40.8	76.5	59.2	64.3	17.3	9.2	10.2	25.5	13.3	13.3	**93.9**	32.7	20.4	**100.0**
L3 (color)	96	55.2	28.1	35.4	65.6	47.9	55.2	11.5	7.3	7.3	22.9	10.4	12.5	**92.7**	19.8	15.6	**96.9**
L4 (dead ends)	300	47.0	19.3	26.3	66.3	41.3	47.7	13.3	6.7	7.7	18.0	7.7	9.3	**89.7**	21.0	21.3	**97.0**
L5 (twins)	100	34.0	14.0	19.0	58.0	31.0	40.0	10.0	6.0	6.0	13.0	7.0	5.0	**84.0**	27.0	20.0	**95.0**

Table 12: **Two-hop success rate (%).** Same corpora, checkpoints, and scoring conventions as Table 4. L4 uses test worlds w_1–w_3; other levels use development world w_0. Caption→Qwen3 retains the query image as pixels. In block (b), the agent uses the learned per-step retriever; the other trained-agent blocks use the chain tool. Vague prompts are evaluated on L1–L5. Bold marks the best result within each training block on each side of the dashed rule.

		Retrieval methods													Agentic search		
		Embedding retrieval						Caption → embedding						VHOP-ROUTER	GE2	Qwen3 + VHOP-ROUTER	
		GE2			Qwen3			GE2			Qwen3						
Level	n	1-shot	Greedy	Hist.	1-shot	Greedy	Hist.	1-shot	Greedy	Hist.	1-shot	Greedy	Hist.				
(a) Precise prompt, 3-hop training																	
L1 (re-ID)	99	2.0	13.1	12.1	1.0	21.2	12.1	0.0	2.0	3.0	0.0	6.1	5.1	**81.8**	90.9	83.8	**96.0**
L2 (spatial)	98	2.0	14.3	9.2	4.1	20.4	12.2	0.0	1.0	3.1	4.1	5.1	9.2	**44.9**	**91.8**	88.8	72.4
L3 (color)	97	4.1	13.4	10.3	2.1	14.4	5.2	0.0	0.0	1.0	1.0	3.1	4.1	**46.4**	84.5	**85.6**	79.4
L4 (dead ends)	300	2.7	8.3	8.0	1.0	12.3	6.0	0.0	1.0	0.3	1.0	4.3	4.3	**45.0**	65.3	66.7	**75.0**
L5 (twins)	100	2.0	8.0	6.0	2.0	6.0	4.0	0.0	0.0	0.0	0.0	3.0	4.0	**38.0**	50.0	64.0	**70.0**
L6 (logo)	94	3.2	3.2	2.1	4.3	7.4	4.3	1.1	0.0	0.0	2.1	2.1	1.1	**9.6**	23.4	25.5	**28.7**
L7 (text)	95	5.3	**6.3**	4.2	1.1	2.1	0.0	1.1	0.0	0.0	1.1	1.1	2.1	2.1	36.8	**43.2**	29.5
(b) Precise prompt, mixed-hop training																	
L1 (re-ID)	99	2.0	13.1	12.1	1.0	21.2	12.1	0.0	2.0	3.0	0.0	6.1	5.1	**89.9**	**90.9**	83.8	89.9
L2 (spatial)	98	2.0	14.3	9.2	4.1	20.4	12.2	0.0	1.0	3.1	4.1	5.1	9.2	**92.9**	**91.8**	88.8	90.8
L3 (color)	97	4.1	13.4	10.3	2.1	14.4	5.2	0.0	0.0	1.0	1.0	3.1	4.1	**94.8**	84.5	85.6	**92.8**
L4 (dead ends)	300	2.7	8.3	8.0	1.0	12.3	6.0	0.0	1.0	0.3	1.0	4.3	4.3	**90.0**	65.3	66.7	**75.3**
L5 (twins)	100	2.0	8.0	6.0	2.0	6.0	4.0	0.0	0.0	0.0	0.0	3.0	4.0	**87.0**	50.0	64.0	**77.0**
L6 (logo)	94	3.2	3.2	2.1	4.3	7.4	4.3	1.1	0.0	0.0	2.1	2.1	1.1	**23.4**	23.4	25.5	**26.6**
L7 (text)	95	5.3	**6.3**	4.2	1.1	2.1	0.0	1.1	0.0	0.0	1.1	1.1	2.1	**6.3**	36.8	43.2	**46.3**
(c) Vague prompt, 3-hop training																	
L1 (re-ID)	99	1.0	8.1	6.1	2.0	11.1	3.0	0.0	2.0	1.0	0.0	3.0	1.0	**37.4**	23.2	28.3	**79.8**
L2 (spatial)	49	0.0	4.1	4.1	0.0	14.3	0.0	2.0	8.2	4.1	0.0	4.1	2.0	**40.8**	24.5	30.6	**81.6**
L3 (color)	48	0.0	6.2	8.3	0.0	14.6	6.2	0.0	0.0	0.0	0.0	0.0	0.0	**25.0**	6.2	6.2	**72.9**
L4 (dead ends)	150	0.0	2.7	1.3	0.7	7.3	4.0	0.0	0.0	0.0	0.0	1.3	0.0	**20.0**	7.3	6.7	**59.3**
L5 (twins)	50	0.0	4.0	2.0	0.0	10.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	**22.0**	2.0	10.0	**58.0**
(d) Vague prompt, mixed-hop training																	
L1 (re-ID)	99	1.0	8.1	6.1	2.0	11.1	3.0	0.0	2.0	1.0	0.0	3.0	1.0	**55.6**	23.2	28.3	**80.8**
L2 (spatial)	49	0.0	4.1	4.1	0.0	14.3	0.0	2.0	8.2	4.1	0.0	4.1	2.0	**59.2**	24.5	30.6	**93.9**
L3 (color)	48	0.0	6.2	8.3	0.0	14.6	6.2	0.0	0.0	0.0	0.0	0.0	0.0	**54.2**	6.2	6.2	**83.3**
L4 (dead ends)	150	0.0	2.7	1.3	0.7	7.3	4.0	0.0	0.0	0.0	0.0	1.3	0.0	**53.3**	7.3	6.7	**84.7**
L5 (twins)	50	0.0	4.0	2.0	0.0	10.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	**48.0**	2.0	10.0	**78.0**

E.2 AGENT MODEL AND RETRIEVAL TOOL

Settings for Figure 1. Success rates use the same 300 precise three-hop L4 test queries. Panel (a) compares Qwen3 and VHOP-ROUTER, each standalone and as the Flash agent's retrieval tool. VHOP-ROUTER uses precise mixed-hop training and checkpoint `mixh_rl_best`. Colors identify retrieval tools; circles denote standalone retrieval, and diamonds denote Flash agents. The green dashed line connects the Pareto frontier among the four plotted configurations.

The horizontal axis uses a linear scale. It adds generated tokens, including thinking tokens, to retrieval hops, counting each hop as one token. For agents, we sum tokens over all model calls and use the length of the recorded retrieval trajectory as the hop count. These means come from a separate pilot on 10 matched queries in w_1. Standalone Qwen3 uses five retrieval steps. The standalone Router count is estimated as $4 + 2b$, where b is the mean number of backtracks, giving 10.11 for the mixed-hop policy. The corresponding means for Flash with Qwen3 and VHOP-ROUTER are 1,885.8 and 727.9. This is a token-count convention, not a measure of latency or total computation.

Untrained retrievers count an answer found anywhere in the returned images; VHOP-ROUTER allows 16 policy actions and requires a correct answer at STOP. Agents are scored on their final answers. Agents with GE2 or Qwen3 allow 16 generation rounds; agents with VHOP-ROUTER allow 8. The generation limit is 8,000 tokens per model call. Panel (b) uses the same mixed-hop Router and Flash + Qwen3 baseline: upgrading Flash to Pro adds 3.7 percentage points, while replacing Qwen3 with VHOP-ROUTER adds 52.7 (Tables 4 and 13).

Figure 11 extends Figure 1b to all seven levels and both tool training settings. Table 13 lists all model–tool combinations. On L4, Flash + Qwen3 achieves 32.7% success. Upgrading Flash to Pro

raises this to 36.3%; replacing Qwen3 with the precise three-hop or mixed-hop VHOP-ROUTER raises it to 59.7% or 85.4%, respectively. The gains are 3.7, 27.0, and 52.7 percentage points, computed before rounding. With the mixed-hop tool, Flash and Pro reach 85.4% and 84.7%.

Table 13: **Agent model and retrieval tool.** Three-hop answer accuracy (%) with Gemini 3.5 Flash or Gemini 3.1 Pro, up to 16 tool calls, and the same answer-scoring convention. Both Router tools use precise training prompts. L4 uses the 300-query test set; other levels use development world w_0. The mean averages the seven levels. Bold marks the highest value in each column.

Retrieval tool	Agent	L1	L2	L3	L4	L5	L6	L7	mean
GE2 (untrained)	Flash	60.6	58.2	56.7	19.3	15.0	6.4	12.6	32.7
	Pro	**93.9**	**89.8**	**90.7**	37.0	40.0	11.7	11.6	53.5
Qwen3 (untrained)	Flash	73.7	69.4	77.3	32.7	28.0	9.6	14.7	43.6
	Pro	78.8	77.6	**90.7**	36.3	34.0	9.6	**18.9**	49.4
VHOP-ROUTER, 3-hop	Flash	85.9	62.2	58.8	59.7	57.0	12.8	6.3	49.0
	Pro	82.8	64.3	72.2	64.7	61.0	**16.0**	6.3	52.5
VHOP-ROUTER, mixed-hop	Flash	80.8	80.6	89.7	**85.4**	**84.0**	10.6	5.3	**62.3**
	Pro	78.8	81.6	88.7	84.7	83.0	11.7	4.2	61.8

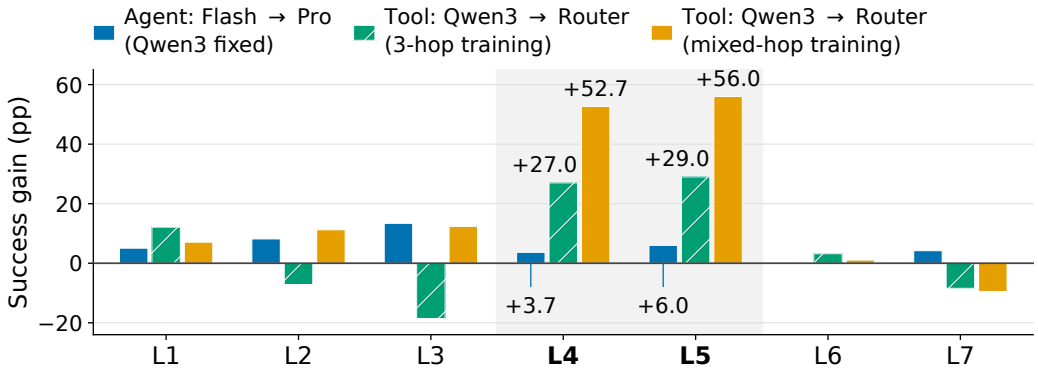


Figure 11: **Agent and tool upgrades across all seven levels.** Full version of Figure 1b, showing gains over Flash + Qwen3 on precise three-hop queries, including both Router training configurations. Router labels indicate training hops. L4 uses the 300-query test set; other levels use development world w_0. Gains are computed before rounding.

E.3 NUMBER OF IMAGES RETURNED PER CALL

Table 14 reports accuracy and context usage for the retrieval-width experiment. Larger candidate sets improve the untrained tools, with GE2 reaching 81.3% on L4 at $k = 50$. The precise mixed-hop VHOP-ROUTER reaches 85.4% while returning fewer images. The number of image occurrences sent to the model includes repeated conversation history; it is distinct from the number of retrieved images and from measured latency or monetary cost.

Table 14: **Full retrieval-width comparison.** Three-hop answer accuracy (%) with Flash and up to 16 tool calls. L4 uses test worlds w_1–w_3; other levels use development world w_0. A chain call returns the policy's active image chain. L4 context statistics are per-query means: *calls* counts tool calls, *images* counts returned images including repeats, and *sent* also counts repeated images in the conversation history. At $k = 50$, three GE2 and four Qwen3 conversations reached the request-size limit and used the last top-1 candidate as the fallback answer. Dashes indicate settings not evaluated.

Retrieval tool	k	L1	L2	L3	L4	L5	L6	L7	calls	images	sent
GE2 (untrained)	1	60.6	58.2	56.7	19.3	15.0	6.4	12.6	14.5	15	120
	3	82.8	81.6	78.4	46.7	40.0	10.6	24.2	11.8	35	249
	5	86.9	86.7	84.5	61.0	60.0	19.1	36.8	9.6	48	300
	10	92.9	93.9	88.7	78.0	72.0	31.9	42.1	6.9	69	344
	50	–	–	–	81.3	–	–	–	5.4	270	1084
Qwen3 (untrained)	1	73.7	69.4	77.3	32.7	28.0	9.6	14.7	13.0	13	104
	3	66.7	64.3	71.1	36.7	26.0	7.4	22.1	11.9	36	255
	5	75.8	78.6	82.5	58.0	54.0	12.8	32.6	9.3	47	295
	10	88.9	85.7	94.8	69.3	65.0	25.5	43.2	7.3	73	392
	50	–	–	–	78.7	–	–	–	5.5	275	1163
Flash + VHOP-ROUTER (3-hop)	chain	85.9	62.2	58.8	59.7	57.0	12.8	6.3	2.6	14	42
Flash + VHOP-ROUTER (mixed-hop)	chain	80.8	80.6	89.7	85.4	84.0	10.6	5.3	2.1	12	33

E.4 DECISION PENALTY

Figure 12 adds training reward and stopping behavior to the rollout length and test accuracy shown in the main text. Table 15 gives the checkpoint scores and prompt-transfer results. At iteration 200, the $\lambda = 0$ and $\lambda = 0.02$ runs reach 48.0% and 47.7% on L4. On the vague prompts, the mean accuracy rises from 26.0% to 32.3%, although L4 decreases from 20.0% to 18.0%. The mean stop rate increases and the mean number of backtracks decreases. Each setting has one run, so these results do not establish statistical equivalence or seed-level reliability. Appendix D.4 records the optimization settings, including the gradient-memory adjustments.

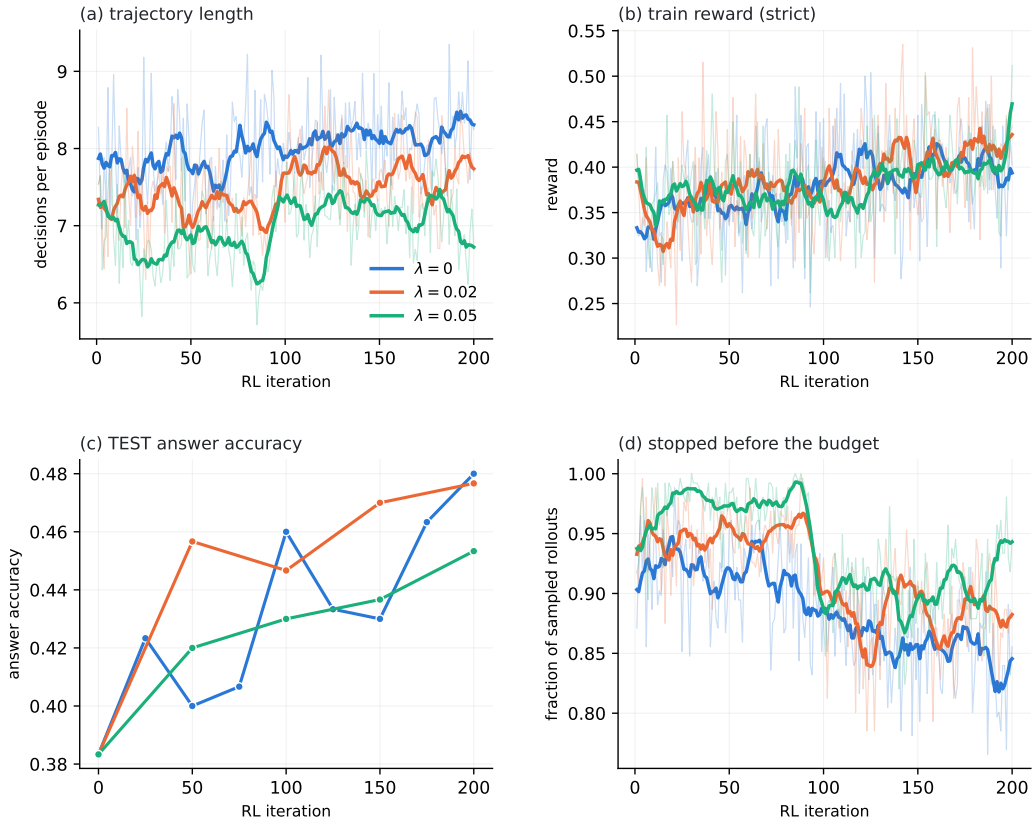


Figure 12: **Full training dynamics with a decision penalty.** Runs use $\lambda \in \{0, 0.02, 0.05\}$ with the same 16-decision budget. **(a)** Mean decisions per rollout. **(b)** Unpenalized terminal reward r, defined identically across runs. **(c)** Success rate on the preregistered 300-question L4 test set. **(d)** Fraction of sampled rollouts that issue STOP before exhausting the budget. Thin curves show per-iteration values and thick curves show moving averages in (a), (b), and (d).

Table 15: **Decision-penalty results.** Comparison of $\lambda = 0$ and $\lambda = 0.02$ with gated decoding. *In-domain* rows use the L4 test set. *Vague* rows measure stopping on a verified acceptable answer, using development world w_0 except for L4, which uses the test worlds. *Rephrased* rows use two natural-language versions of the L4 development questions; they are separate from the human-authored dataset. Stop rate and backtracks are means over the five vague-prompt levels. Accuracy is in percent; stop rate is a fraction. Bold marks the better value. Each setting uses one seed.

		$\lambda = 0$	$\lambda = 0.02$	Δ
in-dom.	L4 test, iter 50	41.0	**45.7**	$+4.7$
	L4 test, iter 100	**46.0**	44.7	-1.3
	L4 test, iter 200	**48.0**	47.7	-0.3
vague	L1	32.3	**37.4**	$+5.1$
	L2	34.7	**49.0**	$+14.3$
	L3	27.1	**31.2**	$+4.1$
	L4	**20.0**	18.0	-2.0
	L5	16.0	**26.0**	$+10.0$
	mean	26.0	**32.3**	$+6.3$
repr.	rephrased v1	19.0	19.0	± 0.0
	rephrased v2	26.0	**31.0**	$+5.0$
beh.	stop rate	0.43	**0.51**	$+0.08$
	backtracks	2.69	**2.22**	-0.47

F Generalization and Limitations

F.1 Three-Object Scenes

The three-object evaluation adds a distractor to each two-object hop scene while preserving the question and target. Query views and answer portraits are unchanged. The added objects come from a separate pool. The standalone VHop-Router rows in Figure 5 use the policy alone; Flash appears only in the rows explicitly labeled as agents. No agent with the trained chain tool is included in that comparison.

The precise three-hop and mixed-hop policies decrease from 48.0% to 30.4% and from 76.3% to 52.2%, respectively. The perturbed results average the three test worlds, with 99, 100, and 100 valid queries. The drop shows sensitivity to competing visual content, but this experiment alone does not separate failures of object recognition from failures of search control.

F.2 Four-Hop Extrapolation

The four-hop evaluation extends L4 beyond the maximum training depth. It uses 97 questions over a 600-image development corpus. All policies receive precise prompts. Table 16 reports the better of two decoding settings for each checkpoint: a question-specific minimum stop depth and stopping permitted after the first retrieval. Thus the comparison gives each policy the benefit of either stopping constraint.

Table 16: **Four-hop answer accuracy (%).** Standalone policies on L4 four-hop development queries, with a 16-decision budget. Values use the better of the two stopping constraints described above.

Training configuration	Accuracy
Precise, 3-hop	1.0
Precise, mixed-hop	1.0
Vague, 3-hop	8.2
Vague, mixed-hop	1.0

All four policies remain at or below 8.2%. For the precise three-hop policy, the depth floor gives 0.0% and removing it gives 1.0%; the precise mixed-hop policy obtains 1.0% either way. The depth

floor therefore does not resolve the failure. This test also increases the corpus from roughly 450 to 600 images and adds another layer of distractors. It measures transfer to longer, harder searches, rather than isolating hop count alone. Training on four-hop worlds is a possible next experiment; recovery from such training has not been established here.

F.3 Realistic Queries and Settings

Table 17 expands the results on realistic queries in Table 8. All methods use the same 50 questions and 386-image corpus, without further training. Annotators write the queries and specify everyday scenes; the images are re-rendered while preserving the intended content. These are human-authored tasks with rendered images, so the experiment measures transfer beyond procedural prompts and scene templates, rather than performance on unmodified personal photographs.

Table 17: **Full results on realistic queries.** Answer accuracy and set success (%) for standalone VHop-Router and Flash with each retrieval tool. The metric definitions are given below.

Method	Training configuration	Answer accuracy	Set success
VHop-Router	Precise, 3-hop	12.0	20.0
	Precise, mixed-hop	4.0	20.0
	Vague, 3-hop	4.0	4.0
	Vague, mixed-hop	14.0	14.0
Flash + VHop-Router	Precise, 3-hop	20.0	36.0
	Precise, mixed-hop	20.0	34.0
	Vague, 3-hop	8.0	18.0
	Vague, mixed-hop	16.0	30.0
Flash + GE2	Untrained	12.0	22.0
Flash + Qwen3	Untrained	8.0	26.0

Metrics. For standalone VHop-Router, answer accuracy (`answer_acc`) requires Stop with an acceptable image at the top of the stack. Set success (`e2e`) checks whether an acceptable image is anywhere in the final active stack; popped images do not count. For Flash, answer accuracy (`ans_acc`) checks the final returned answer. If there is no valid `FINAL` declaration, the evaluator uses the last answer candidate returned by VHop-Router, or the last retrieved image for GE2 and Qwen3. Set success (`final_gt`) checks the cumulative returned image set across tool calls. These set metrics therefore retain different histories for standalone policies and agents.

Evaluation records and budgets. The precise three-hop and mixed-hop rows use `rloo10w10_200` and `mixh_rl_best`. The archived human-authored runs label `vgo_rl_best` as vague three-hop and `vgmix_rl_best` as vague mixed-hop. These are the checkpoints underlying the vague rows here; they differ from the `vg_rl_best`/`vgo_rl_best` pair used for the vague blocks in the procedural main table. The recorded Flash–VHop-Router runs allow up to eight chain calls, each with at most 16 policy decisions. GE2 and Qwen3 runs allow up to 16 single-step tool calls. Each chain call may return multiple images, so a chain call and a single-step call have different costs.

The main table reports the agent's set success: 36.0% and 34.0% for the two precise VHop-Router tools, compared with 22.0% and 26.0% for GE2 and Qwen3. Their answer accuracies are 20.0%, 20.0%, 12.0%, and 8.0%. The gap between finding an answer and returning it shows that answer selection remains a separate source of error. With only 50 queries, these results support a limited transfer claim, not reliable performance across arbitrary photo collections.

F.4 Extensions to Logo and Text Matching

The matching rule can extend beyond physical object identity. L6 adds logo matching, and L7 adds printed-text matching. The rule is chosen independently for each hop and specified in the instruction; the actual logo or text must be read from the image. These extensions retain the constraints

and distractors from L1–L5. Rendering checks also cover logos and printed text, and training and evaluation use disjoint logo and printed-string pools.

The L6 evaluation world contains 94 queries, 501 corpus images, and 50 query images; L7 contains 95 queries, 481 corpus images, and 50 query images. Table 18 reports the three-hop results with precise prompts. Tables 11 and 12 include the one- and two-hop results. The trained policies use the same L4 checkpoints as the main evaluation, without further training on either extension.

Table 18: **Three-hop success on logo and text matching (%).** These rows extend Table 4, with the same methods and scoring rules. Both levels use precise prompts and development world w_0. The strategy or training configuration is listed in the second column.

Method	Strategy / training	L6 (logo)	L7 (text)
GE2	Single-shot	2.1	3.2
GE2	Greedy	4.3	2.1
GE2	History	2.1	2.1
Qwen3	Single-shot	3.2	4.2
Qwen3	Greedy	4.3	3.2
Qwen3	History	4.3	3.2
Caption → GE2	Single-shot	2.1	0.0
Caption → GE2	Greedy	1.1	0.0
Caption → GE2	History	0.0	0.0
Caption → Qwen3	Single-shot	3.2	4.2
Caption → Qwen3	Greedy	3.2	2.1
Caption → Qwen3	History	2.1	2.1
VHOP-ROUTER	Precise, 3-hop	7.4	1.1
VHOP-ROUTER	Precise, mixed-hop	5.3	2.1
Flash + GE2	Untrained	6.4	12.6
Flash + Qwen3	Untrained	9.6	14.7
Flash + VHOP-ROUTER	Precise, 3-hop	12.8	6.3
Flash + VHOP-ROUTER	Precise, mixed-hop	10.6	5.3

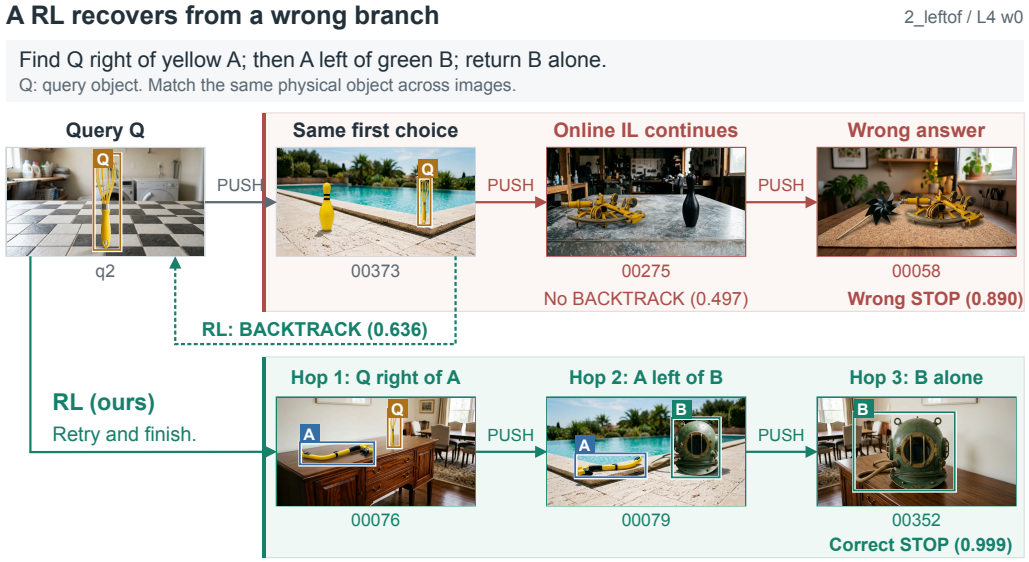
Training uses L4 identity links, so these results test transfer to matching rules absent from post-training. With precise mixed-hop training, standalone VHOP-ROUTER achieves 5.3% on L6 and 2.1% on L7; Flash with the same tool achieves 10.6% and 5.3%, respectively. The results do not establish whether the main cause is reading the marks, applying the new matching rule, or controlling the resulting search. Wider retrieval improves some L6/L7 agent results in Table 14, but it does not resolve these questions. Training with logo and text links, together with separate perception checks, would be needed to distinguish these possibilities.

G TRAJECTORY CASE STUDIES

These examples show the retrieval decisions behind the results. In the figures, PUSH denotes SELECT and POP denotes BACKTRACK; image identifiers refer to entries in the same world.

G.1 RECOVERING FROM A WRONG BRANCH

We compare checkpoints from precise three-hop training on L4 development query `2_leftof`. The solver must find object Q to the right of a yellow object A, then A to the left of a green object B, and finally B alone. The gold chain is `00076→00079→00352`. Figure 13 contrasts Online IL and RLVR after the same initial mistake; the following figures show the other methods on this query.



Other methods on the same question

Untrained (greedy)	00141 → 00041 → 00203 → 00302 → 00027	Wrong
SFT	00076 → 00281 → 00058 → 00300 → 00278	Wrong
SFT + hard negatives	00076 → 00382 → 00183 → 00197 → 00376	Wrong
Flash + GE2	16 tool calls; gold answer not retrieved	Wrong
Flash + Qwen3-VL	16 tool calls; gold answer not retrieved	Wrong

Figure 13: **Recovery after the same incorrect first choice.** Online IL and RLVR both first select `00373`. Online IL continues along that branch and stops on a wrong answer. RLVR backtracks with $p_{\text{bt}} = 0.636$, retrieves the three gold images, and stops with $p_{\text{stop}} = 0.999$. Solid arrows show selections and the dashed return shows backtracking. The lower rows summarize the other methods; the untrained and SFT trajectories use five fixed retrieval steps.



Figure 14: **Untrained retrieval misses the first hop.** The complete five-step greedy trajectory for `2_leftof` ends at `00027`. Every retrieved image lies outside the gold chain (red frames). This decoder takes five steps without learned backtracking or stopping.

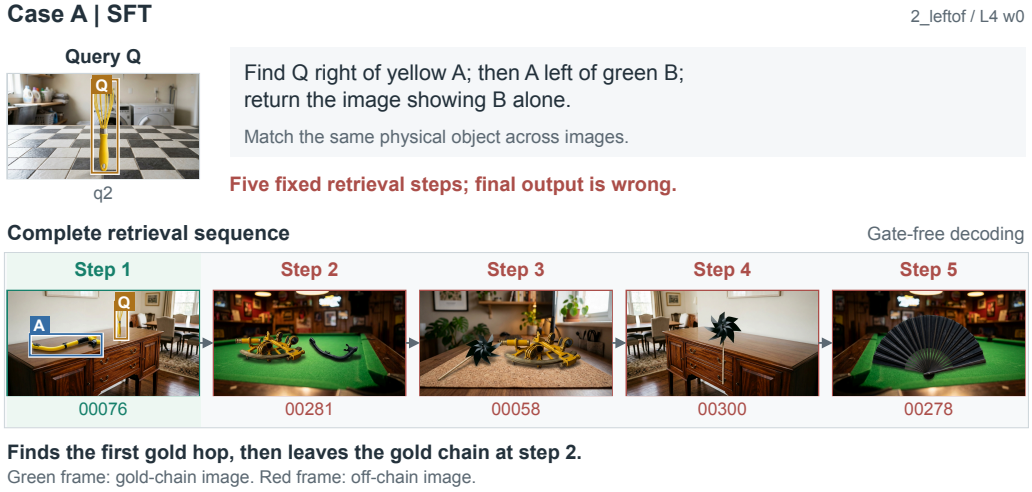


Figure 15: **SFT retrieves the first hop, then leaves the chain.** The state-encoder warm-up retrieves `00076` (green frame), then selects `00281` and eventually ends at `00278`. The full five-step trajectory is shown with the same decoder as the untrained model.

Case A | Flash + gemini-embedding-2

2_leftof / L4 w0

Query Q



q2

Find Q right of yellow A; then A left of green B;
return the image showing B alone.

Match the same physical object across images.

16 tool calls; the correct answer is never retrieved.

Calls 1-8

	Query and image input	Tool image
1	yellow whisk Image input: none	 00389
2	yellow whisk next to yellow object Image input: none	 00076
3	yellow snorkel to the left of green object Image input: 00076	 00281
4	yellow snorkel to the left of a green object Image input: 00076	 00223
5	yellow snorkel Image input: 00076	 00373
6	yellow bowling pin to the left of green object OR yellow snorkel to the left of green object Image input: 00373, 00076	 00158
7	yellow bowling pin Image input: 00373	 00349
8	yellow bowling pin next to green Image input: 00373	 00331

Calls 9-16

	Query and image input	Tool image
9	yellow bowling pin on beach or pool or indoors Image input: 00373	 00407
10	yellow bowling pin Image input: none	 00316
11	yellow bowling pin with other objects Image input: 00373	 00251
12	yellow bowling pin by the pool Image input: 00373	 00021
13	yellow bowling pin to the left of Image input: 00373	 00160
14	yellow bowling pin next to green ladle OR green hanger OR green whisk OR green microphone OR green tambourine Image input: 00373	 00194
15	yellow bowling pin to the left of a green object Image input: 00373	 00197
16	yellow bowling pin to the left of green colander / strainer Image input: 00373, 00197	 00231

Wrong after 16 calls. Gold hop 1 appears at call 2; hops 2 and 3 never appear.

Figure 16: **Flash with GE2 on the same query.** All 16 tool calls are shown, read down the left column and then down the right. Each row gives the recorded query, image inputs, and retrieved image. Call 2 retrieves the first gold hop, but neither the second hop nor the answer is retrieved. The recorded outcome is incorrect.

Case A | Flash + Qwen3-VL-Embedding-2B 2_leftof / L4 w0

Query Q



q2

Find Q right of yellow A; then A left of green B; return the image showing B alone.

Match the same physical object across images.

16 tool calls; the correct answer is never retrieved.

Calls 1-8

	Query and image input	Tool image
1	yellow whisk Image input: none	 00373
2	yellow bowling pin Image input: none	 00316
3	yellow bowling pin green Image input: none	 00419
4	bowling pin Image input: none	 00349
5	yellow bowling pin green Image input: none	 00248
6	yellow bowling pin Image input: none	 00181
7	yellow bowling Image input: none	 00275
8	yellow bowling pin Image input: 00373	 00158

Calls 9-16

	Query and image input	Tool image
9	green hanger Image input: none	 00262
10	hanger Image input: none	 00105
11	green plastic hanger Image input: none	 00194
12	green plastic hanger alone Image input: none	 00153
13	hanger Image input: 00158	 00331
14	green hanger alone Image input: none	 00427
15	green roller skate Image input: none	 00243
16	green roller skate alone Image input: none	 00221

Wrong after 16 calls. None of the three gold-chain images appears.

Figure 17: **Flash with Qwen3 on the same query.** The complete 16-call trace uses the same layout as Figure 16. None of the three gold images is retrieved. The last thumbnail shows the final tool result; the figure does not supply a separate answer declaration.

G.2 Revising the Query and Selecting an Answer

An agent can help at two different points. If a returned chain lacks the answer, it can call the policy again with revised text. If the answer is already present, it can select that image even when the policy continues searching. Figures 18 and 19 show these behaviors on two separate queries.

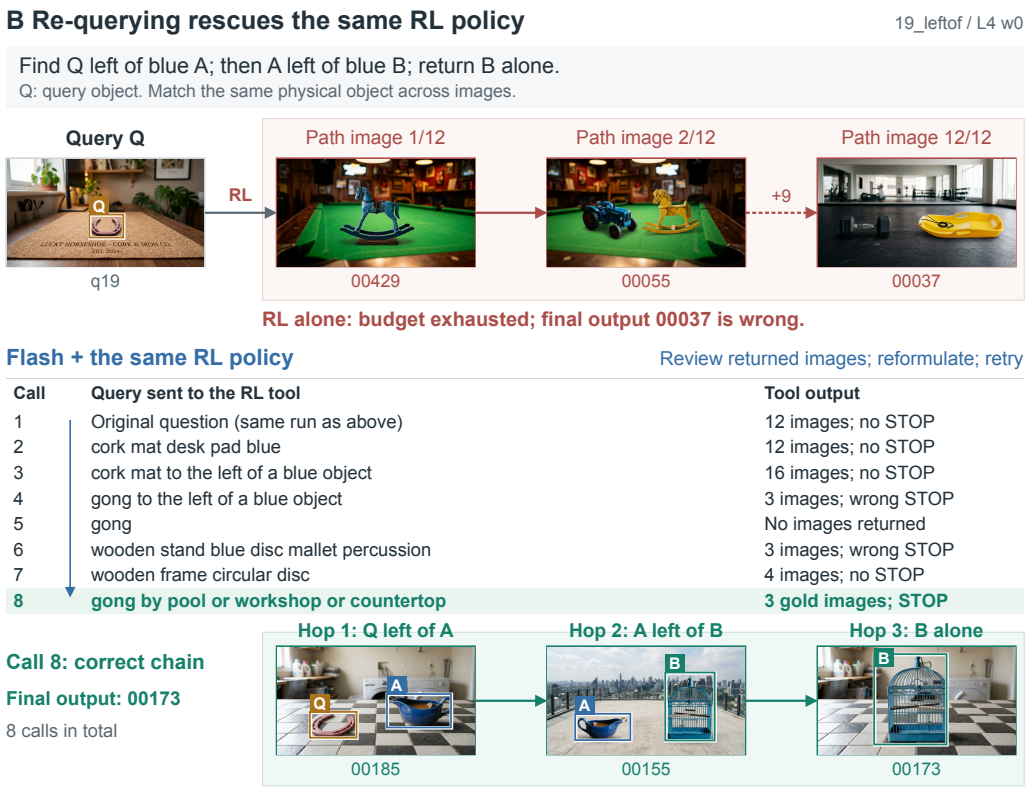


Figure 18: **Re-querying the same policy recovers a valid chain.** On L4 development query `19_leftof`, the initial policy call exhausts its budget and returns a chain without the answer. Flash retries the same checkpoint and query image with revised text. The eighth call retrieves `00185`→`00155`→`00173` and stops correctly. Query strings are copied from the trace. The recorded system answer is `00173`; the trace does not contain a separate `FINAL` declaration.

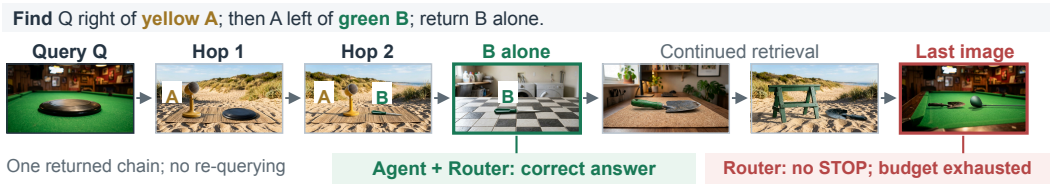


Figure 19: **Answer selection from the same returned chain.** On L4 development query `33_leftof`, precise mixed-hop VHop-Router retrieves the gold chain but continues searching until its 16-action budget runs out. Its returned stack contains six images and ends at the incorrect image `00435`. Without re-querying, Flash selects the correct third image, `00247`.

G.3 A Vague Query with Two Acceptable Answers

The vague question in Figure 20 omits the relation that distinguishes two precise chains. Both share the first hop, and both endpoints are verified answers. The demonstration record specifies one branch; the outcome reward accepts either endpoint. In this trace, RLVR first explores the demonstrated branch, backtracks, and stops at the other acceptable endpoint. Online IL repeatedly backtracks and exhausts its budget without returning an answer.

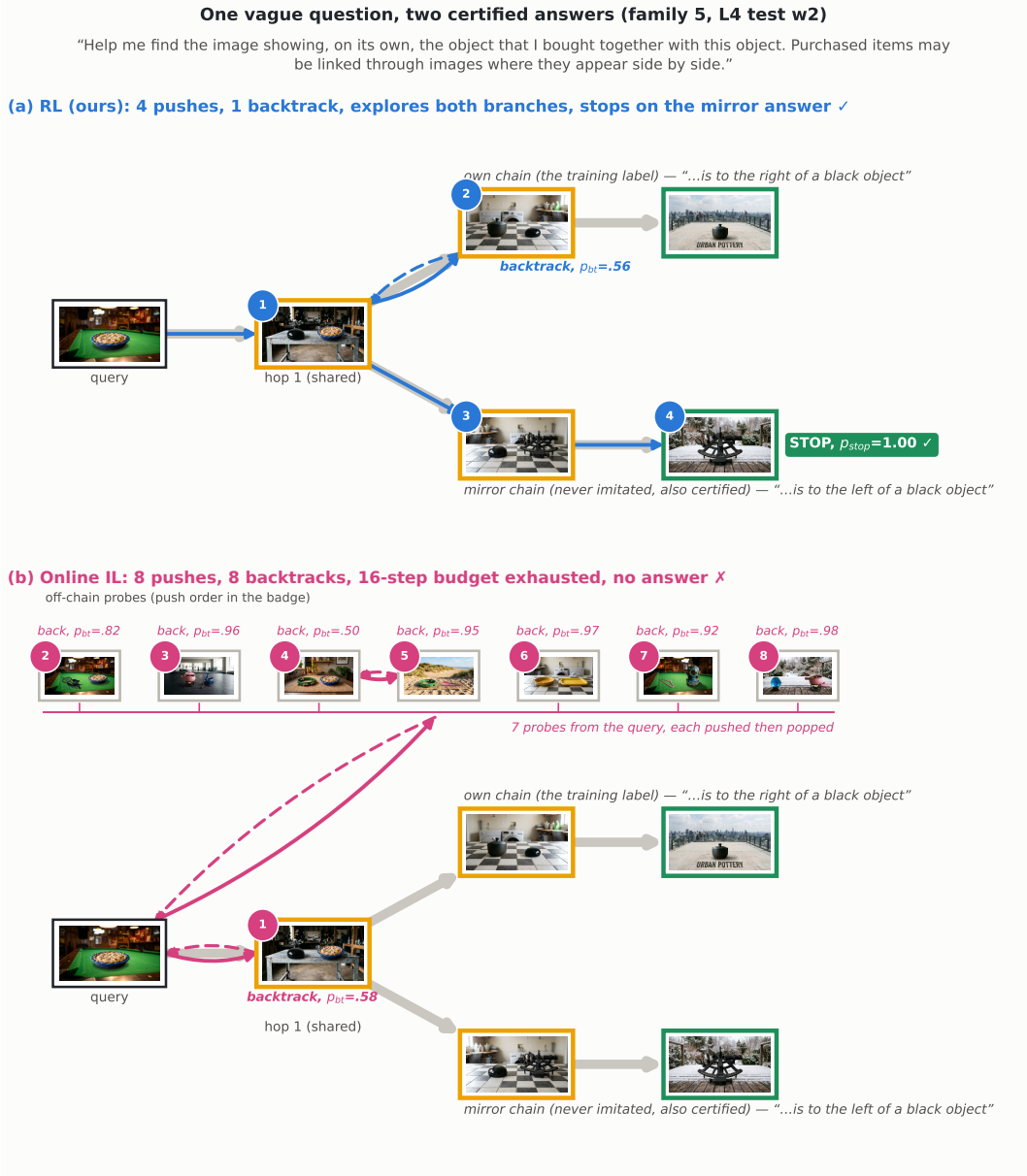


Figure 20: **Two valid endpoints under one vague instruction.** Family 5 from L4 test world w_2. Gray paths show the two verified chains; colored arrows show the recorded policy decisions. Solid arrows are selections and dashed arrows are backtracks. RLVR makes four selections, backtracks once, and stops at the alternative valid answer. Online IL makes eight selections and eight backtracks, exhausting its 16-decision budget. Both are standalone policies.