SELECTION-BASED STRUCTURED REASONING: TOWARD EFFICIENT MULTIMODAL SEARCH AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal agents commonly generate freeform reasoning before each action. For small models, limited model capacity can result in lengthy reasoning that provides little useful guidance for action generation while incurring substantial inference cost. To address this challenge, we introduce Selection-based Structured Reasoning (SSR), a framework that reformulates reasoning as selection instead of open-ended generation. SSR represents recurring high-level reasoning as pre-specified, reusable natural-language candidates. At each turn, the model selects from the reasoning candidates based on their likelihoods given the current context, without an auxiliary task head. Using pre-specified reasoning traces enables parallel scoring, where teacher-forced prefilling computes token likelihoods concurrently within and across candidates using a shared context KV cache. We evaluated SSR on seven multimodal search benchmarks using 2B and 4B models. Across multiple reinforcement learning objectives and supervised fine-tuning, SSR delivers significant efficiency gains without sacrificing task performance. SSR achieves an average success rate competitive with leading search agents of the same scale, while reducing per-turn reasoning latency by over 90% and total per-question model inference latency by 28–54%.

1 INTRODUCTION

Multimodal agents turn textual and visual observations into actions: they analyze multimodal inputs, gather information, and use tools to resolve tasks that require multi-turn interactions (He et al., 2024; Chng et al., 2025). A common agent design interleaves observations, freeform reasoning, and actions (Yao et al., 2022). The reasoning step lets the model assess the available evidence and decide what to do next, but adds sequential token generation cost before every action. For small, on-device models with limited reasoning capacity, lengthy texts and repeated deliberations may waste the generation cost without providing useful guidance for generating better actions. An efficient agent needs reasoning that guides it to choose the desired action with minimal repeating cost of generating reasoning traces.

Existing approaches reduce this overhead by prompting concise traces or rewarding shorter reasoning (Xu et al., 2025; Arora & Zanette, 2025). These approaches reduce the amount of text generated, yet keep the sequential dependency between its tokens during autoregressive decoding. We investigate an orthogonal direction: changing the representation of reasoning itself. We observed that in multimodal search, entities, queries, and observations vary across questions, but the high-level reasoning for taking the next action often recurs. For example, finding the location of an upcoming AI conference and the year LEGO was founded can share the same rationale: the entity is known but the requested information is missing, so search for the entity name and the attribute of interest. Previous work on thought templates (Yang et al., 2024; Ouyang et al., 2026) reported a similar finding, showing the value of reusing reasoning knowledge across problems. Therefore, the required reasoning space for search agents does not span the entire natural language space. Instead, the model can achieve a decent success rate with a few reasoning candidates to choose from.

Motivated by this, we introduce **Selection-based Structured Reasoning (SSR)**, a framework that formulates reasoning as *selection* among reusable natural-language candidates instead of open-ended freeform generation. At each turn, the agent selects a reasoning candidate conditioned on its interaction history, then uses the selected reasoning to generate the next action and its arguments

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

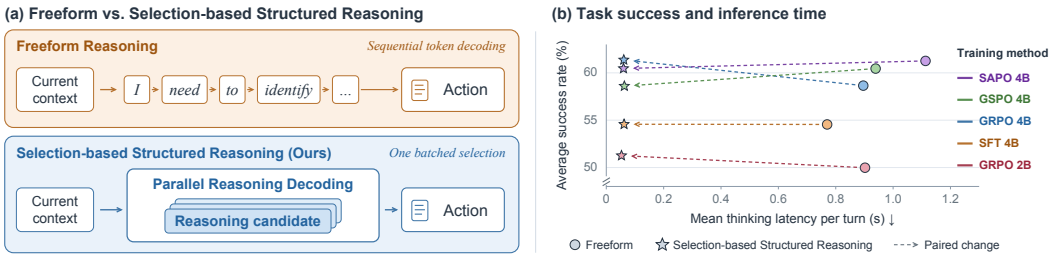

Figure 1: **SSR significantly reduces inference latency. Left:** Parallel reasoning decoding replaces autoregressive reasoning generation. **Right:** Across four training objectives and two model sizes, SSR achieves comparable or higher success rates while reducing mean reasoning latency per turn by more than 90%. Mean total model inference latency per question is reduced by 28–54% (Table 2).

(e.g., a text search with a question-specific query). This design separates reusable reasoning patterns from instance-specific execution: the selected candidate provides the rationale for the next step, while action generation supplies the details needed to realize it. The agent then adapts its actions to the current context without generating freeform reasoning traces at any stage.

This representation enables **parallel reasoning decoding** (Figure 1). We use the language model directly as a candidate classifier, without any additional head, to score each reasoning candidate by its length-normalized likelihood conditioned on the current context. This leverages the language understanding capability of the model to match the context to the most suitable reasoning. Since all candidate tokens are pre-specified, teacher forcing computes token likelihoods in parallel within each reasoning candidate, and candidates are also scored concurrently sharing the same KV cache. The model then samples a reasoning candidate based on its score, and the harness inserts the selected reasoning into the model context as natural-language guidance for the next action.

We evaluate SSR across seven multimodal search benchmarks. Our 4B SSR with GRPO model achieves an average success rate of 61.37%, matching the performance of leading trained 4B search agents (61.25%). Across GRPO (Shao et al., 2024), GSPO (Zheng et al., 2025b), SAPO (Gao et al., 2025), and SFT, the SSR agents achieve comparable task performance while significantly reducing model inference latency compared to the freeform reasoning counterpart. Against four representative efficient reasoning baselines, SSR achieves a higher success rate, lower model inference time, and more consistent latency.

In summary, our contributions are three-fold:

- **Reasoning by selection rather than generation.** We introduce Selection-based Structured Reasoning (SSR), which replaces freeform reasoning with selection among reusable natural language candidates conditioned on the current context.
- **Parallel reasoning decoding.** We use the language model itself to score reasoning candidates by their length-normalized likelihoods. This approach requires neither an additional classification head nor an SFT warm start. Teacher forcing enables parallel scoring across all tokens within and between candidates, eliminating autoregressive generation of reasoning traces.
- **Competitive performance with substantial efficiency gains.** Across GRPO, GSPO, SAPO, and SFT, our 4B agents maintain competitive performance while reducing mean reasoning latency per turn by over 90% and mean model latency per question by 28–54%.

2 Related Work

2.1 Multimodal Search Agents

Prior work trained multimodal agents to interleave perception, retrieval, and tool use. In multimodal search, MMSearch-R1 (Wu et al., 2025) and DeepMMSearch-R1 (Narayan et al., 2025) trained models to use text and image search, while Visual-ARFT (Liu et al., 2025) and DeepEyes (Zheng et al., 2026) added image operations such as cropping and zooming. DeepEyesV2 (Hong et al., 2026) combined image operations with code execution and web search. SenseNova-MARS (Chng et al., 2025), TAPO (Dong et al., 2026), and WebWatcher (Geng et al., 2025) investigated multi-turn rein-

forcement learning, tool-aware credit assignment, and deep-research behavior. More recent systems extended the scale and supervision of this recipe: VSearcher trained long-horizon search over text, images, and webpages (Zhang et al., 2026), while ProMMSearchAgent introduced process-oriented rewards in a static sandbox before transfer to live search (Yan et al., 2026). These approaches focused on training-data mixtures, gradient routing, and credit assignment for performance improvements, while we explore an orthogonal direction that improves inference efficiency.

2.2 Efficient Reasoning for MLLM Agents

Efficient-reasoning methods controlled model deliberation through token budgets, early stopping, compressed chains of thought, or adaptive test-time computation (Han et al., 2025; Ma et al., 2025; Sui et al., 2025). Agentic settings add another resource axis: the model must decide not only how long to think, but also whether to spend the next unit of budget on retrieval, interaction, or answer commitment. Inference-time budget control for search agents explicitly modeled this trade-off (Fang et al., 2026). Closest to our efficiency goal, HyperEyes (Li et al., 2026) treated inference efficiency as a first-class training objective: it dispatched multiple grounded queries concurrently within a round and rewarded trajectories that avoided superfluous tool calls, reducing the *number* of interaction rounds, whereas we reduce the *cost of the reasoning span* within each round. Grammar-constrained decoding enforced admissible output forms (Geng et al., 2023), with engines such as XGrammar reducing the cost of constraint checking (Dong et al., 2025). SSR instead scores complete candidates in parallel and directly supplies the selected text as the reasoning trace.

2.3 Reusable Reasoning and Agent Skills

Prior work explored reusable knowledge through thought templates (Yang et al., 2024), context-aware guidelines (Fu et al., 2024), reasoning memories (Ouyang et al., 2026), and task workflows (Wang et al., 2025). Given a library of reasoning candidates, SSR directly uses the acting MLLM's internal semantic knowledge to select among them through context-conditioned likelihoods, without any auxiliary knowledge-management or retrieval system. Trajectory-level approaches such as StraTA (Xue et al., 2026) generated a strategy from the initial task state and kept that strategy fixed throughout the interaction. SSR instead selects a reasoning candidate at every turn based on the updated interaction history, allowing new observations to change the guidance for subsequent actions. Programmatic approaches (Wang et al., 2023; Zheng et al., 2025a) encoded reusable skills as fixed, executable programs. An SSR candidate instead supplies natural-language guidance that biases action generation without prescribing executable control flow. This leaves concrete queries, tool arguments, and answers flexible, allowing the same reasoning candidate to support different actions across contexts without modifying the candidate itself.

3 Selection-Based Structured Reasoning

3.1 Overview

We introduce **Selection-based Structured Reasoning (SSR)**, which replaces freeform reasoning in ReAct agents (Yao et al., 2022) with *selection* among reusable reasoning candidates.

At turn t, the interaction history of an agent is $h_t = [x, z_0, a_0, o_0, \ldots, z_{t-1}, a_{t-1}, o_{t-1}]$, where x is the initial user request, z_i is a reasoning trace, a_i is an action, and o_i is the observation returned by an external tool. A tool call extends the history with a new observation, while a final answer terminates the trajectory. Conventionally, an MLLM with distribution p_θ generates a freeform reasoning trace $z_t \sim p_\theta(\cdot \mid h_t)$ before generating the action $a_t \sim p_\theta(\cdot \mid h_t, z_t)$. Since z_t is unconstrained natural-language text, its tokens must be decoded autoregressively, resulting in substantial inference latency.

To eliminate this sequential decoding cost, SSR selects z_t from reusable natural-language candidates conditioned on h_t, instead of generating it token by token (Figure 2). SSR contains a *reasoning library* $\mathcal{R} = \{r_1, \ldots, r_N\}$, which consists of short natural-language candidates. Each candidate is a reusable reasoning trace that describes what information is available or missing in the current context and what step to take next. *Parallel reasoning decoding* uses the language model itself to score the full text of each candidate conditioned on h_t and select one as z_t. Since all candidate texts are specified in advance, their token likelihoods can be computed with teacher-forced prefill and

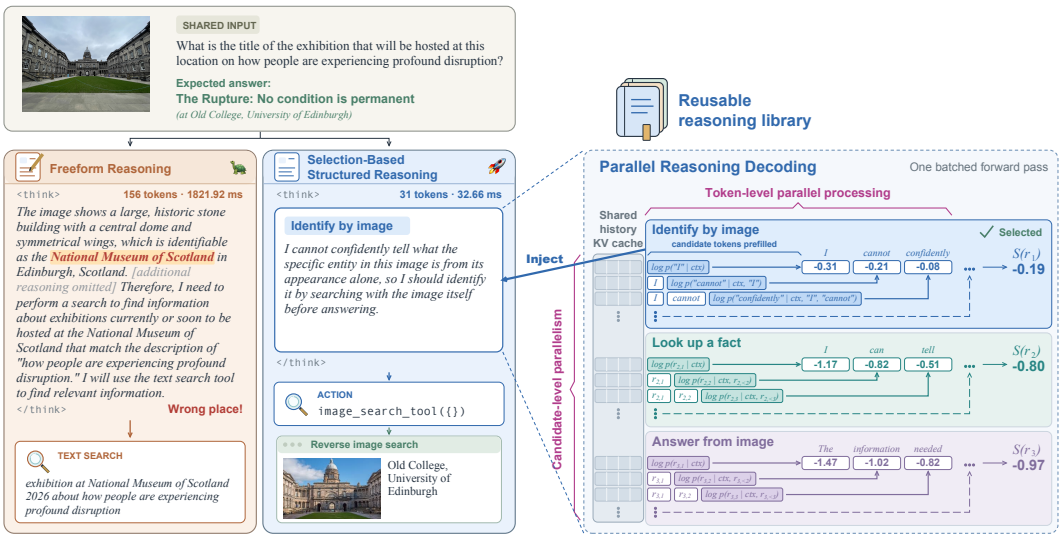

Figure 2: **Overview of Selection-based Structured Reasoning.** Freeform agents generate token-by-token reasoning before each action. SSR instead selects from a library of reusable natural-language reasoning candidates. Parallel reasoning decoding scores every token in each candidate using teacher-forced prefill, and the selected candidate guides context-specific action generation.

different candidates can be scored in parallel. The selected reasoning is then inserted into the context to guide instance-specific action generation, such as a search query, crop location, or final answer. This formulation leverages the semantics prior in the language decoder for reasoning selection, and makes use of the natural-language guidance for the action generation.

3.2 Parallel Reasoning Decoding

Scoring and selection. We define a scoring function to measure how suitable each reasoning candidate is for the current context. Specifically, for a candidate r_i that contains K_i tokens $(r_{i,1}, r_{i,2}, \ldots, r_{i,K_i})$, we compute its length-normalized log-likelihood given context h_t as:

$$S_\theta(r_i \mid h_t) = \frac{1}{K_i^\alpha} \sum_{j=1}^{K_i} \log p_\theta(r_{i,j} \mid h_t, r_{i,<j}) \tag{1}$$

where $\alpha \in [0, 1]$ controls length normalization. We use $\alpha = 1$ to compare candidates by their mean token log-likelihood, reducing the length bias from summing negative log-probabilities, which otherwise favors the shortest candidate. Applying a softmax gives the reasoning-selection policy $q_\theta(\cdot \mid h_t)$, a categorical distribution over $\mathcal{R}$:

$$q_\theta(r_i \mid h_t) = \frac{\exp(S_\theta(r_i \mid h_t)/\tau)}{\sum_{k=1}^{N} \exp(S_\theta(r_k \mid h_t)/\tau)} \tag{2}$$

where $\tau > 0$ is the selection temperature. This represents a context-dependent preference over the reasoning candidates. The model samples $r_t \sim q_\theta(\cdot \mid h_t)$, inserts the selected text as the reasoning trace, and generates the action from $p_\theta(\cdot \mid h_t, r_t)$. Scoring and action generation use the same model parameters and system prompt, without the need for any extra components. This mechanism also constrains the reasoning of the model to be in the library, regardless of its initial distribution.

Latency improvement through parallelism. Since all reasoning candidates are pre-specified, we exploit both candidate-level and token-level parallelism to reduce reasoning latency. At the token level, since the prefix $r_{i,<j}$ (Eq. 1) is part of the given candidate, we can prefill all of the tokens in one reasoning candidate through teacher forcing and score the log-probabilities of all tokens in one batched forward pass. All tokens in the reasoning text will be processed concurrently parallel within each attention layer. At the candidate level, each $S_\theta(r_i \mid h_t)$ can be computed concurrently given the same history KV cache for h_t. Throughout this reasoning process, the model does not need to generate any new token.

Next, we characterize the latency advantage of parallel reasoning decoding in terms of computational depth. Consider a model with fixed number of layers, token dimensions, and vocabulary size. Let H be the number of tokens in the history context; N be the number of candidates in the library; $K_{\max} = \max_i K_i$ be the maximum sequence length of the candidates; and L be the number of tokens in a freeform reasoning trace generated by autoregressive decoding. Then we can compare the length of the critical path of freeform and SSR:

$$\begin{aligned} \mathcal{D}_{\text{freeform}} &= O\Big(\sum_{\ell=0}^{L-1} \log(H+\ell)\Big) = O(L\log(H+L)) \\ \mathcal{D}_{\text{SSR}} &= O\Big(\underbrace{\log(H+K_{\max})}_{\text{scoring the longest candidate}} + \underbrace{\log N}_{\text{selecting among the candidates}}\Big) \end{aligned} \tag{3}$$

where applying attention over m tokens has $O(\log m)$ parallel depth due to the softmax and weighted sum operations. The difference in time complexity highlights the latency advantage of the proposed selection method, which removes the factor of L through token-level parallelism.

3.3 SSR Training

SSR provides a general alternative to freeform reasoning under both supervised fine-tuning (SFT) and reinforcement learning (RL) objectives. At each turn, the policy makes a categorical reasoning selection and then autoregressively generates the action and its arguments. The reasoning-selection and action-generation objectives share the same model parameters and can be optimized jointly in an end-to-end manner.

Supervised fine-tuning. Given context h_t, ground truth reasoning candidate z_t from the library, and ground truth action a_t (tool call or answer), we define the SFT loss as a weighted sum of the categorical selection loss and the token-level action loss (modulated by weighting factor $\lambda^{\text{SFT}}_{\text{reason}}$):

$$\mathcal{L}_{\text{SFT}} = -\sum_t \left[\lambda^{\text{SFT}}_{\text{reason}} \log q_\theta(z_t \mid h_t) + \frac{1}{|a_t|} \sum_{k=1}^{|a_t|} \log p_\theta(a_{t,k} \mid h_t, z_t, a_{t,<k}) \right] \tag{4}$$

GRPO-style reinforcement learning. Unlike many prior methods (Hong et al., 2026) that require an SFT warm-up to teach the base model a specific way of reasoning, SSR can be trained directly with RL. This is attributable to reasoning selection restricting the model to candidates in the library. Compared with freeform GRPO (Shao et al., 2024), the main difference is to replace the token-level importance ratios for generated reasoning with a single categorical ratio for reasoning selection. The action tokens retain their usual importance ratios:

$$\rho_t^{\text{reason}} = \frac{q_\theta(z_t \mid h_t)}{q_{\text{old}}(z_t \mid h_t)} \qquad \rho_{t,k}^{\text{action}} = \frac{p_\theta(a_{t,k} \mid h_t, z_t, a_{t,<k})}{p_{\text{old}}(a_{t,k} \mid h_t, z_t, a_{t,<k})} \tag{5}$$

where q_{old} and p_{old} use the rollout parameters θ_{old}. The inserted reasoning text contributes to its *single* categorical term. After a candidate is selected, the harness inserts its entire token sequence into the model context as a single block.

Specifically, for each question x, we sample multiple trajectories to form group G by selecting $z_t \sim q_{\text{old}}(\cdot \mid h_t)$ and generating the action autoregressively from $p_{\text{old}}(\cdot \mid h_t, z_t)$ at each turn. For trajectory g in the group, its group-normalized advantage is $\widehat{A}_g = (R_g - \overline{R})/\sigma_R$, where R_g is its reward, $\overline{R}$ and σ_R are the reward mean and standard deviation within the group. After computing the ratios in Eq. 5, they enter the clipped surrogate $\ell_\epsilon(\rho, A) = \min\{\rho A, \text{clip}(\rho, 1-\epsilon, 1+\epsilon)A\}$. And the final objective combines the reasoning-selection and action-token terms:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{g\in G}\left[\sum_{t=0}^{T_g-1} \left[\lambda^{\text{RL}}_{\text{reason}} \ell_\epsilon(\rho^{\text{reason}}_{g,t}, \widehat{A}_g) + \sum_k \ell_\epsilon(\rho^{\text{action}}_{g,t,k}, \widehat{A}_g) \right] \right] - \beta \mathcal{K}_{\text{ref}}(\theta) \tag{6}$$

where T_g is the number of turns, and $\mathcal{K}_{\text{ref}}$ is the reference-policy KL regularizer on generated action-token distributions, weighted by β. The same reasoning-selection and action-generation components can also be trained with other GRPO variants, such as GSPO and SAPO. Appendix B gives the variant-specific objectives and practical selector-gradient approximation.

4 EXPERIMENTS

4.1 SETUP

Datasets. Following prior work (Chng et al., 2025; Dong et al., 2026), we use VisualProbe (Lai et al., 2026), DeepEyes (Zheng et al., 2026), and FVQA-train (Wu et al., 2025) for training, and use MMSearch (Jiang et al., 2024), HR-MMSearch (Chng et al., 2025), FVQA-test (Wu et al., 2025), InfoSeek (Chen et al., 2023), SimpleVQA (Cheng et al., 2025), LiveVQA (Fu et al., 2025), and MAT-Search (Liu et al., 2025) for evaluation. Appendix D details the datasets used.

Implementation details. We use the 2B/4B Qwen3-VL (Qwen Team, 2025) models for our experiments. Following the aforementioned prior work, we use the same tool harness with reverse-image search, text search, and image cropping, as well as the same LLM-as-judge for evaluation. Our reasoning library contains six manually written reasoning candidates shared across benchmarks (details in Appendix E): (1) doing an image search to identify the entity when unsure, (2) looking up specific information through a text search, (3) zooming into a specific part of the image for targeted inspection, (4) answering the question from the gathered evidence, (5) answering the question based on the image, and (6) answering with general knowledge. Each of them covers a general case that the search agent may encounter. Unless otherwise noted, we train each RL model in a single stage for one epoch with a learning rate of 10^{-5} using outcome and format rewards. For simplicity, we set $\lambda_{\text{reason}}^{\text{SFT}} = \lambda_{\text{reason}}^{\text{RL}} = 1$ and $\beta = 10^{-3}$. Appendix F contains additional implementation details.

Evaluation metrics. We report success rate on all seven multimodal search benchmarks and inference efficiency. For latency profiling, we sample 100 questions per benchmark and measure per-turn reasoning latency (time to produce the reasoning trace) and per-question model latency (prefill, reasoning, action generation, and other model inference over the full trajectory). We report mean and 95th-percentile latency. Environment-side processing, including tool execution and LLM-as-judge evaluation, is excluded as it introduces excessive model-independent variations. We also report effective reasoning throughput, computed as the total number of reasoning tokens processed divided by the total reasoning time. For SSR, this counts all candidate tokens being scored in parallel. For autoregressive generation, it counts the tokens in the generated reasoning trace. We run the profiling on an NVIDIA H100 GPU with an offline SGLang (Zheng et al., 2024) inference engine.

4.2 SSR ACHIEVES COMPETITIVE TASK PERFORMANCE

We first compare SSR with the state-of-the-art multimodal search agents. Table 1 shows that the 4B SSR model reaches 61.37% average success rate, comparable to the best 4B SOTA model TAPO with GSPO at 61.25% and outperforms 4B MMSearch-R1 and SenseNova-MARS baselines. The 2B SSR agent reaches 51.26%, comparable to the zero-shot 8B model, showing the generalization of the approach to a smaller backbone. These results show that selecting reusable reasoning can support competitive success rates in multimodal agentic search tasks.

4.3 COMPARISON BETWEEN SSR AND FREEFORM REASONING

Table 2 compares freeform reasoning and SSR across GRPO, GSPO, and SAPO, three representative RL algorithms used in Dong et al. (2026), as well as SFT. For the RL algorithms, we directly train the model with outcome and format rewards without a separate SFT stage. For SFT training data construction, we use Gemini 3.1 Pro (Google DeepMind, 2026a) to generate the ground truth trajectory using the reasoning candidates from the library.

Competitive success rates across objectives and model sizes. SSR improves average success over freeform GRPO at both model sizes and matches freeform SFT. It shows modest decreases under GSPO and SAPO. We hypothesize this comes from the selector-gradient approximation (Appendix B.3), which becomes less accurate across multiple gradient updates per rollout.

Lower reasoning latency across training objectives. SSR brings more than 90% reduction to the per-turn freeform reasoning latency and reduces mean per-question model latency by 28-54%. Scoring complete candidates in parallel removes the sequential generation of the reasoning trace,

Table 1: **Success rate (%) on multimodal search benchmarks.** SSR achieves comparable or better performance than state-of-the-art multimodal search agents of the same size. **Bold** denotes the best result among 4B models; underlines mark the second best. (* reproduced results).

Model	Size	MM Search	HR-MM Search	FVQA test	Simple VQA	Live VQA	MAT Search	InfoSeek	AVG
Agentic models: zero-shot (proprietary)									
GPT-4o-mini (OpenAI, 2024b)	–	38.60	26.23	50.00	50.84	31.54	80.00	42.35	45.65
GPT-4o (OpenAI, 2024a)	–	49.12	30.16	66.34	63.67	40.09	76.67	59.55	55.09
GPT-5 (OpenAI, 2025a)	–	52.63	38.36	62.61	70.58	56.02	84.67	55.95	60.12
Gemini-3-Flash (Google DeepMind, 2025)	–	62.57	41.64	64.89	67.92	48.06	82.67	61.10	61.26
GPT-5.2 (OpenAI, 2025b)	–	66.08	48.20	68.78	78.18	65.99	80.67	65.55	67.64
Gemini-3-Pro (Google DeepMind, 2026b)	–	74.27	48.52	72.61	75.91	59.69	86.00	66.45	69.06
Agentic models: zero-shot (open-source)									
Qwen3-VL-32B-Instruct (Qwen Team, 2025)	32B	49.12	34.43	54.28	64.17	42.87	82.00	49.85	53.82
Qwen2.5-VL-32B-Instruct (Bai et al., 2025)	32B	49.71	33.44	52.22	65.15	42.17	81.33	50.10	53.45
Qwen3-VL-8B-Instruct (Qwen Team, 2025)	8B	47.37	27.87	53.61	62.29	39.37	82.67	46.15	51.33
Qwen2.5-VL-7B-Instruct (Bai et al., 2025)	7B	32.16	19.34	36.00	42.35	22.52	67.33	28.80	35.50
Qwen3-VL-4B-Instruct (Qwen Team, 2025)	4B	43.86	25.57	50.50	63.57	37.34	79.33	43.25	49.06
Qwen3-VL-2B-Instruct (Qwen Team, 2025)	2B	32.35	20.00	42.90	49.16	31.69	70.47	33.20	39.97
Agentic models: trained ($\geq$ 7B)									
Skywork-R1V4 (Zhang et al., 2025)	30B	66.10	–	67.20	–	–	–	–	–
SenseNova-MARS (Chng et al., 2025)	8B	66.67	40.33	67.11	70.19	56.22	83.33	61.70	63.65
TAPO + SAPO (Dong et al., 2026)	8B	67.21	40.66	69.89	73.64	59.71	82.67	62.70	65.21
Visual-ARFT (Liu et al., 2025)	7B	34.50	24.92	41.72	42.45	25.40	74.00	37.95	40.13
DeepMMSearch-R1 (Narayan et al., 2025)	7B	–	–	–	55.87	–	–	47.51	–
MMSearch-R1 (Wu et al., 2025)	7B	53.80	20.33	58.40	57.40	48.40	74.00	55.10	52.49
DeepEyesV2 (Hong et al., 2026)	7B	63.70	–	60.60	59.40	–	–	51.10	–
Agentic models: trained ($\leq$ 4B)									
MMSearch-R1* (Wu et al., 2025)	4B	53.80	25.50	67.20	68.50	50.60	78.00	60.00	57.66
SenseNova-MARS* (Chng et al., 2025)	4B	44.05	24.67	57.19	65.55	43.06	78.67	48.40	51.65
TAPO + GRPO (Dong et al., 2026)	4B	54.97	**33.11**	65.56	68.31	54.05	82.00	58.85	59.55
TAPO + GSPO (Dong et al., 2026)	4B	60.82	31.48	66.61	70.98	56.14	**83.33**	59.40	61.25
TAPO + SAPO (Dong et al., 2026)	4B	56.73	31.48	67.17	**71.27**	55.36	81.33	59.95	60.47
SSR (ours)	4B	**61.40**	30.16	**70.22**	71.08	**57.19**	78.67	**60.90**	**61.37**
SSR (ours)	2B	51.18	17.06	61.16	61.83	51.54	62.59	53.44	51.26

Table 2: **Success rate and inference efficiency across training objectives and model sizes.** SSR achieves success rates comparable to those of freeform reasoning while reducing reasoning latency per turn by over 90% and model latency per question by 28–54%. *Italic* percentages show relative changes from the paired freeform model.

Training Objective	Size	Success Rate (%) ↑	Reasoning Latency per Turn (s) ↓	Model Latency per Question (s) ↓	Effective Reasoning Throughput (toks/s) ↑
GRPO-freeform	4B	58.65	0.896	5.456	72.7
GRPO-SSR	4B	61.37 *(+4.6%)*	0.061 *(-93.2%)*	2.514 *(-53.9%)*	3029.9 *(+4070.2%)*
GSPO-freeform	4B	60.45	0.939	5.195	71.9
GSPO-SSR	4B	58.60 *(-3.1%)*	0.063 *(-93.3%)*	3.706 *(-28.7%)*	2526.2 *(+3414.6%)*
SAPO-freeform	4B	61.26	1.113	6.145	69.4
SAPO-SSR	4B	60.46 *(-1.3%)*	0.060 *(-94.6%)*	2.862 *(-53.4%)*	3215.3 *(+4535.0%)*
SFT-freeform	4B	54.55	0.770	5.308	70.9
SFT-SSR	4B	54.57 *(+0.0%)*	0.062 *(-91.9%)*	3.318 *(-37.5%)*	2567.0 *(+3520.5%)*
GRPO-freeform	2B	49.98	0.902	4.153	87.2
GRPO-SSR	2B	51.26 *(+2.6%)*	0.053 *(-94.1%)*	2.125 *(-48.8%)*	3302.6 *(+3687.9%)*

yielding higher effective reasoning throughput that matches the token processing speed during prefill. These results show consistent latency benefits across different training methods.

Table 3: **Success rate and efficiency measurement on the profiling subset.** Our approach yields the highest success rate while being significantly faster than the baselines.

Method	Subset SR (%) ↑	Reasoning Latency/Turn (s)		Model Latency/Question (s)		Effective Reasoning Throughput (toks/s) ↑
		Mean ↓	p95 ↓	Mean ↓	p95 ↓	
MMSearch-R1-4B (Wu et al., 2025)	51.59	1.235	1.892	5.601	7.502	73.8
SenseNova-MARS-4B (Chng et al., 2025)	49.57	0.960	1.533	5.289	10.397	72.5
Chain-of-Draft (Xu et al., 2025)	54.06	0.416	0.798	3.657	6.616	69.9
Sketch-of-Thought (Aytes et al., 2025)	56.07	0.543	1.054	3.487	7.048	71.7
Efficiency-reward RL (Arora & Zanette, 2025)	56.28	0.756	1.273	4.638	8.675	68.9
Probe & Prefill (Sun et al., 2026)	55.49	0.890	1.564	5.219	11.369	65.8
SSR (Ours)	**58.36**	**0.061**	**0.071**	**2.514**	**4.620**	**3029.9**

Table 4: **Comparison between different reasoning methods.** Selecting only the index degrades performance, while autoregressive decoding without SFT fails to limit its reasoning to the reasoning candidates in the library.

Reasoning Method	MM Search	HR-MM Search	FVQA test	Simple VQA	Live VQA	MAT Search	InfoSeek	AVG
Parallel reasoning decoding								
Entire reasoning	**61.40**	30.16	**70.22**	**71.08**	**57.19**	78.67	60.90	**61.37**
Index only	60.82	22.30	67.56	69.99	55.56	77.33	59.30	58.98
Autoregressive decoding								
With SFT	**61.40**	29.84	69.22	69.10	56.97	72.00	**61.55**	60.01
Without SFT (*degraded to freeform)	60.23	**33.11**	67.28	68.41	54.67	**79.33**	58.60	60.23

4.4 Inference Latency and Reasoning Throughput

To compare SSR with existing efficient-reasoning methods, we compare four baselines spanning three approaches: prompt-based compression with Chain-of-Draft (Xu et al., 2025) and Sketch-of-Thought (Aytes et al., 2025), learning shorter reasoning through efficiency-reward RL (Arora & Zanette, 2025), and optional reasoning trace decoding with a lightweight classification head in Probe & Prefill (Sun et al., 2026). We implement the efficiency baselines on the GRPO-freeform pipeline for a fair comparison and report the results in Table 3.

Higher success rate with lower inference latency. SSR achieves the highest success rate and delivers approximately $7\times$ or greater speedup in reasoning on all baselines. The gains extend to question-level: mean model latency is around $2.5s$, approximately 30% lower than Sketch-of-Thought, the fastest baseline on this metric. These results show that parallel reasoning decoding is a more effective than reasoning-compression methods at reducing latency while maintaining high success rate.

The speed advantage is even larger at the 95th percentile. SSR has a p95 reasoning time of $0.071s$, 10 ms above its mean, whereas the p95 latency for the baselines range from 0.798 to $1.892s$. The tight mean-to-p95 gap indicates consistent reasoning latency in SSR, matching the predictable computation of scoring all candidates in parallel as opposed to freeform generation with varying lengths. At the question level, p95 latency is $4.6s$, about a 30% reduction from the $6.6s$ of the fastest baseline.

4.5 Comparison Between Different Reasoning Methods

We ablate the generation methods and compare different ways to select reasoning from the library. Table 4 compares full-reasoning-text selection, index-only selection, and autoregressive reasoning generation. These variants distinguish the effects of the reasoning selection mechanism in SSR.

Reasoning representation for candidate selection. The index-only variant applies Eq. 1 to the index of each candidate (e.g., `[1]`) rather than its full text, then relies on the harness to inject the selected candidate before action generation. Both variants supply the same structured reasoning to the action generation, but index-only scoring selects without access to the semantics of the reasoning tokens. This change lowers average success by 3.9%, affecting all seven benchmarks. This comparison highlights the value of reasoning-candidate semantics for better context-dependent selection.

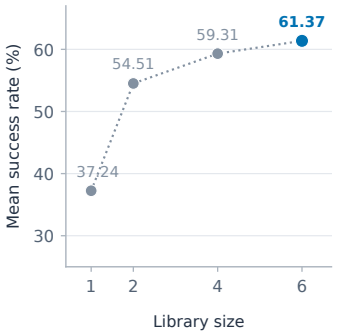

Figure 3: **Reasoning library size.** Average success rate increases as the library size grows.

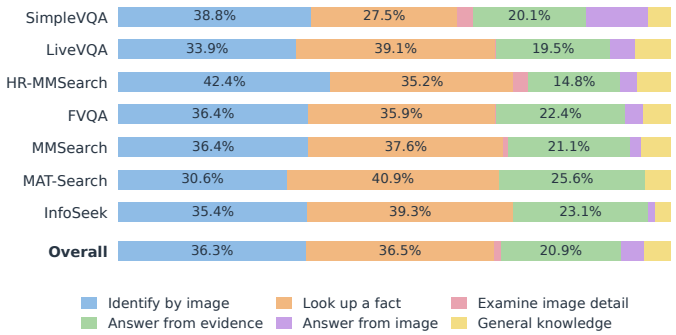

Figure 4: **Reasoning selection distribution across benchmarks.** The same checkpoint exhibits different reasoning-selection distributions across benchmarks.

Selection vs. autoregressive generation. Autoregressive decoding can also generate the specified reasoning candidate when the model is trained to use the library entries. With an SFT warm start followed by RL with a format reward, the 4B model reproduces reasoning-library entries and achieves average success comparable to parallel reasoning decoding. However, without SFT, specifying in the prompt and applying a format reward during RL cannot enforce using only the reasoning candidates from the library, and the model instead generates freeform reasoning. Thus, the autoregressive variant must learn to conform to the reasoning library, whereas our reasoning selection enforces this interface directly. Additionally, parallel full-text scoring further removes the sequential generation of the selected reasoning candidate, combining effective reasoning with lower decoding latency.

4.6 Reasoning Library Size and Selection Distribution

Effect of library size. Figure 3 compares libraries with one, two, four, and six entries, ranging from a single generic candidate, "I am a helpful assistant," to six more specific reasoning candidates. Success rate across seven benchmarks increases monotonically from 37.24% to 61.37%, showing that more detailed, granular reasoning candidates better guide learning for action generation. The 17.27 percentage point gain from one generic candidate to two candidates, answer or tool call, highlights the value of meaningful reasoning choices even in a small library.

Reasoning selection distribution. Figure 4 shows that the same model yields different distributions of reasoning selection across different benchmarks. Specifically, the simplest SimpleVQA has the highest selection rate for "Answer from image" and the lowest for "Look up a fact" and the high-resolution image centric HR-MMSearch has the highest selection rate for "Examine image detail". These distribution differences suggest that SSR adapts its choices to question-specific information needs despite using a shared library. These reasoning choices are reflected in action generation: in 99.9% of turns, the generated action type matches the expected action type of the selected candidate. The agent is indeed demonstrating behaviors that match the charasteristics of different benchmarks.

5 Conclusion

We presented **Selection-based Structured Reasoning (SSR)**, which reformulates reasoning in small multimodal search agents as selection rather than open-ended generation. SSR captures recurring high-level reasoning as reusable natural-language candidates and selects among them based on their likelihoods under the current context. Pre-specified candidates enable teacher-forced parallel scoring within and across candidates, eliminating token-by-token reasoning generation. Across seven multimodal search benchmarks, SSR achieves competitive task performance while reducing per-turn reasoning latency by over 90% and total per-question model inference latency by 28-54%. More broadly, our results suggest that small search agents may not need to reason over the full language space at every step, and that organizing recurring reasoning into a compact, reusable space can enable more efficient multimodal agents.

AI USE STATEMENT

We used generative AI tools to assist with drafting and revising the manuscript, checking mathematical claims and derivations, and locating and formatting references. For the SFT experiments, Gemini 3.1 Pro generated reasoning traces and tool-call trajectories and mapped the retained traces to reasoning candidate labels, as described in Section 4. The experimental pipeline also uses Qwen models to summarize retrieved content and judge open-ended answers, as detailed in Appendix F. The authors take responsibility for the final text, mathematical claims, experimental results, and other AI-assisted content.

REPRODUCIBILITY STATEMENT

Appendix A lists the pseudocode for GRPO training and inference with SSR. Appendix D documents the training data and the evaluation benchmarks. Appendix E provides the reasoning library. Appendix F describes the training data, hyperparameters, interaction limits, action space, scoring protocol, and rewards.

REFERENCES

Daman Arora and Andrea Zanette. Training language models to reason efficiently. In *Advances in Neural Information Processing Systems*, volume 38, pp. 60770–60808, 2025.

Simon A Aytes, Jinheon Baek, and Sung Ju Hwang. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 24296–24320, 2025.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Y. Chen, H. Hu, Y. Luan, H. Sun, S. Changpinyo, A. Ritter, and M. Chang. Can pre-trained vision and language models answer visual information-seeking questions? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14948–14968, 2023.

X. Cheng, W. Zhang, S. Zhang, J. Yang, X. Guan, X. Wu, X. Li, G. Zhang, J. Liu, Y. Mai, et al. Simplevqa: Multimodal factuality evaluation for multimodal large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4637–4646, 2025.

Y. X. Chng, T. Hu, W. Tong, X. Li, J. Chen, H. Yu, J. Lu, H. Guo, H. Deng, C. Xie, et al. Sensenova-mars: Empowering multimodal agentic reasoning and search via reinforcement learning. *arXiv preprint arXiv:2512.24330*, 2025.

Chengqi Dong, Chuhuai Yue, Hang He, Yandong Liu, Fenghe Tang, S Kevin Zhou, Xiaohan Wang, Jiajun Chai, and Guojun Yin. Tapo: Tool-aware policy optimization via credit transfer for multimodal search agents. *arXiv preprint arXiv:2606.05784*, 2026.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. XGrammar: Flexible and efficient structured generation engine for large language models. In *Proceedings of Machine Learning and Systems*, 2025. URL https://arxiv.org/abs/2411.15100.

Zhengru Fang, Senkang Forest Hu, Zhonghao Chang, Yu Guo, Yihang Tao, Hongyao Liu, Mengzhe Ruan, Jun Huang, and Yuguang Fang. Inference-time budget control for LLM search agents. *arXiv preprint arXiv:2605.05701*, 2026.

Mingyang Fu, Yuyang Peng, Benlin Liu, Yao Wan, and Dongping Chen. Livevqa: Live visual knowledge seeking. *arXiv preprint arXiv:2504.05288v1*, 2025. URL https://arxiv.org/abs/2504.05288v1. Version 1.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. AutoGuide: Automated generation and selection of context-aware guidelines for large language model agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://papers.nips.cc/paper/2024/hash/d8efbb5dd415974eb095c3f06bff1f48-Abstract-Conference.html.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*, 2025. URL https://arxiv.org/abs/2511.20347.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured nlp tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10932–10952, 2023.

Xinyu Geng, Peng Xia, Zhen Zhang, Xinyu Wang, Qiuchen Wang, Ruixue Ding, Chenxi Wang, Jialong Wu, Yida Zhao, Kuan Li, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webwatcher: Breaking new frontier of vision-language deep research agent. *arXiv preprint arXiv:2508.05748*, 2025.

Google DeepMind. Gemini 3 Flash model card. https://deepmind.google/models/model-cards/gemini-3-flash/, December 2025.

Google DeepMind. Gemini 3.1 Pro model card. https://deepmind.google/models/model-cards/gemini-3-1-pro/, February 2026a.

Google DeepMind. Gemini 3 Pro model card. https://deepmind.google/models/model-cards/gemini-3-pro/, May 2026b.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025. doi: 10.18653/v1/2025.findings-acl.1274. URL https://aclanthology.org/2025.findings-acl.1274/.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. DeepEyesV2: Toward agentic multimodal model. In *International Conference on Learning Representations*, 2026.

D. Jiang, R. Zhang, Z. Guo, Y. Wu, J. Lei, P. Qiu, P. Lu, Z. Chen, C. Fu, G. Song, et al. Mmsearch: Benchmarking the potential of large models as multi-modal search engines. *arXiv preprint arXiv:2409.12959*, 2024.

Xin Lai, Junyi Li, Wei Li, Tao Liu, Tianjian Li, and Hengshuang Zhao. Mini-o3: Scaling up reasoning patterns and interaction turns for visual search. In *International Conference on Learning Representations*, volume 2026, pp. 76722–76746, 2026.

Guankai Li, Jiabin Chen, Yi Xu, Xichen Zhang, and Yuan Lu. Hypereyes: Dual-grained efficiency-aware reinforcement learning for parallel multimodal search agents. *arXiv preprint arXiv:2605.07177*, 2026.

Z. Liu, Y. Zang, Y. Zou, Z. Liang, X. Dong, Y. Cao, H. Duan, D. Lin, and J. Wang. Visual agentic reinforcement fine-tuning. *arXiv preprint arXiv:2505.14246*, 2025.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*, 2025.

K. Narayan, Y. Xu, T. Cao, K. Nerella, V. M. Patel, N. Shiee, P. Grasch, C. Jia, Y. Yang, and Z. Gan. Deepmmsearch-r1: Empowering multimodal llms in multimodal web search. *arXiv preprint arXiv:2510.12801*, 2025.

OpenAI. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/, May 2024a.

OpenAI. GPT-4o mini: Advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/, July 2024b.

OpenAI. Introducing GPT-5. https://openai.com/index/introducing-gpt-5/, August 2025a.

OpenAI. Introducing GPT-5.2, December 2025b. URL https://openai.com/index/introducing-gpt-5-2/.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Qwen Team. Qwen3-VL: Sharper vision, deeper thought, broader action. https://qwen.ai/blog?from=research.latest-advancements-list&id=99f0335c4ad9ff6153e517418d48535ab6d8afef, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

Chung-En Sun, Linbo Liu, Ge Yan, Zimo Wang, and Tsui-Wei Weng. Llm agents already know when to call tools–even without reasoning. *arXiv preprint arXiv:2605.09252*, 2026.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 63897–63911, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

J. Wu, Z. Deng, W. Li, Y. Liu, B. You, B. Li, Z. Ma, and Z. Liu. Mmsearch-r1: Incentivizing lmms to search. *arXiv preprint arXiv:2506.20670*, 2025.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of Draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

Xiangyuan Xue, Yifan Zhou, Zidong Wang, Shengji Tang, Philip Torr, Wanli Ouyang, Lei Bai, and Zhenfei Yin. StraTA: Incentivizing agentic reinforcement learning with strategic trajectory abstraction. *arXiv preprint arXiv:2605.06642*, 2026.

Wentao Yan, Shengqin Wang, Huichi Zhou, Yihang Chen, Kun Shao, Yuan Xie, and Zhizhong Zhang. ProMMSearchAgent: A generalizable multimodal search agent trained with process-oriented rewards. *arXiv preprint arXiv:2604.20486*, 2026.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. *arXiv preprint arXiv:2406.04271*, 2024.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Ruiyang Zhang, Qianguo Sun, Chao Song, Yiyan Qi, and Zhedong Zheng. VSearcher: Long-horizon multimodal search agent via reinforcement learning. *arXiv preprint arXiv:2603.02795*, 2026.

Y. Zhang, L. Hu, H. Sun, P. Wang, Y. Wei, S. Yin, J. Pei, W. Shen, P. Xia, Y. Peng, T. Xie, E. Li, Y. Liu, X. Song, and Y. Zhou. Skywork-r1v4: Toward agentic multimodal intelligence through interleaved thinking with images and deepresearch. *arXiv preprint arXiv:2512.02395*, 2025.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025a.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025b. URL https://arxiv.org/abs/2507.18071.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. DeepEyes: Incentivizing "thinking with images" via reinforcement learning. In *International Conference on Learning Representations*, 2026.

APPENDIX

A PSEUDOCODE FOR SSR INFERENCE AND GRPO TRAINING

A.1 INFERENCE

Algorithm 1 implements SSR inference. The library $\mathcal{R} = \{r_1, \ldots, r_N\}$ contains N reasoning candidates, with $r_i = (r_{i,1}, \ldots, r_{i,K_i})$ denoting the K_i tokens of a candidate, including its reasoning delimiters. At turn t, h_t is the interaction history, z_t the selected candidate, a_t the generated action or final answer, and o_t the tool observation. The algorithm assumes mean token log-likelihoods ($\alpha = 1$) and greedy selection.

Constructing the reasoning library. A straightforward procedure to construct the reasoning library is to sample training questions and successful freeform reasoning traces, group recurring information needs, and summarize them into a small set of reusable reasoning candidates. Each candidate describes how to make progress without retaining question-specific entities or answers. Ideally, the set of candidates should cover all states that the agent will encounter. Future work could explore skill-evolution methods such as SkillOpt (Yang et al., 2026) to construct and refine the reasoning library automatically from trajectory feedback.

Algorithm 1 SSR agent inference with parallel reasoning decoding

Require: Model p_θ; tokenized library $\mathcal{R}$; initial history h_0; tool executor $\mathcal{T}$; turn limit $T_{\max}$
Ensure: Final answer or a trajectory with an explicit stopping status

1: Initialize prefix cache $\mathcal{C}_{\mathrm{KV}}$ and media cache $\mathcal{C}_{\mathrm{media}}$
2: **for** $t = 0, \ldots, T_{\max} - 1$ **do**
3: $\quad M_t \leftarrow \textsc{ResolveMedia}(h_t, \mathcal{C}_{\mathrm{media}})$
4: $\quad \mathrm{KV}_t \leftarrow \textsc{PrefillHistory}(p_\theta, h_t, M_t, \mathcal{C}_{\mathrm{KV}})$
5: $\quad \{S_\theta(r_i \mid h_t)\}_{i=1}^{N} \leftarrow \textsc{ScoreCandidates}(p_\theta, \mathrm{KV}_t, \mathcal{R}, M_t)$
6: $\quad$ *Score all reasoning candidates in parallel using Eq. (1)*
7: $\quad z_t \leftarrow \arg\max_{r_i \in \mathcal{R}} S_\theta(r_i \mid h_t)$
8: $\quad a_t \leftarrow \textsc{GenerateAction}(p_\theta, h_t \parallel z_t, M_t, \mathcal{C}_{\mathrm{KV}})$
9: $\quad$ **if** a_t is a final answer **then**
10: $\quad\quad$ **return** a_t
11: $\quad$ **end if**
12: $\quad o_t \leftarrow \mathcal{T}(a_t)$
13: $\quad h_{t+1} \leftarrow \textsc{AppendTurn}(h_t, z_t, a_t, o_t)$
14: **end for**
15: **return** Recorded trajectory with status `turn-limit`

Parallel scoring and cache reuse. Here M_t contains the prepared media for h_t, KV_t contains its cached key–value states, and $\parallel$ denotes concatenation. The caches $\mathcal{C}_{\mathrm{KV}}$ and $\mathcal{C}_{\mathrm{media}}$ retain language-model states and reusable media features, respectively. PREFILLHISTORY evaluates the uncached history; SCORECANDIDATES directly returns the mean token log-likelihoods in Eq. (1). Each candidate attends to the history and its own preceding tokens, with its known positions evaluated in parallel within each layer. GENERATEACTION reuses the selected history-plus-candidate prefix without generating the reasoning text again. The executor $\mathcal{T}$ runs tool calls, and APPENDTURN preserves the model's chat template and media references.

A.2 GROUPED GRPO WITH DETACHED COMPETITORS

Algorithm 2 implements grouped GRPO with the detached-competitor approximation in Eq. (12). The rollout parameters θ_{old} remain fixed while collecting a batch and throughout its optimization epochs; the reference model p_{ref} is frozen throughout training.

Grouped rollouts. For each question x, SAMPLESSRROLLOUTS collects a group G_x of $n \geq 2$ live trajectories with independent sampling. It follows Algorithm 1 with gradients disabled, replacing greedy selection (line 7) by $z_t \sim q_{\mathrm{old}}(\cdot \mid h_t)$ and sampling action tokens from $p_{\mathrm{old}}(\cdot \mid h_t, z_t)$.

Algorithm 2 Grouped GRPO for SSR with detached competitor scores

Require: Initial parameters θ; frozen reference p_{ref}; question distribution $\mathcal{D}$; library $\mathcal{R}$; tools $\mathcal{T}$; reward rule; group size n; τ, $T_{\max}$, update epochs E, clipping threshold ϵ, weights $\lambda^{\mathrm{RL}}_{\mathrm{reason}}$, β, numerical floor δ

Ensure: Trained parameters θ

while the training budget remains **do**
  $\theta_{\mathrm{old}} \leftarrow \textsc{Snapshot}(\theta)$; synchronize and freeze the rollout model
  Sample a question batch $X \sim \mathcal{D}$; initialize rollout batch $\mathcal{B} \leftarrow \emptyset$
  for each $x \in X$ **do**
    $G_x \leftarrow \textsc{SampleSSRRollouts}(x, p_{\mathrm{old}}, \mathcal{R}, \mathcal{T}, n, \tau, T_{\max})$
    Evaluate $R_g \leftarrow \textsc{Reward}(x, D_g)$ for every $D_g \in G_x$
    Compute $\overline{R}_x$ and σ_x from $\{R_g : D_g \in G_x\}$
    Store $\widehat{A}_g \leftarrow \mathrm{stopgrad}((R_g - \overline{R}_x)/\max(\sigma_x, \delta))$ for each D_g
    $\mathcal{B} \leftarrow \mathcal{B} \cup G_x$
  end for
  for $e = 1, \ldots, E$ **do**
    for each trajectory minibatch $\mathcal{M}$ from $\mathcal{B}$ **do**
      for each recorded turn $(h_t, z_t, a_t, \ldots)$ in $\mathcal{M}$ **do**
        Recompute $S_\theta(z_t \mid h_t)$ with gradients using Eq. (1)
        Compute $\widetilde{b}^{\mathrm{reason}}_t$ from Eq. (7)
        $\widetilde{\rho}^{\mathrm{reason}}_t \leftarrow \exp(\widetilde{b}^{\mathrm{reason}}_t - b^{\mathrm{reason}}_t)$
        Teacher-force the recorded a_t to obtain $b^{\theta}_{t,k} \leftarrow \log p_\theta(a_{t,k} \mid h_t, z_t, a_{t,<k})$
        $\rho^{\mathrm{action}}_{t,k} \leftarrow \exp(b^{\theta}_{t,k} - b^{\mathrm{action}}_{t,k})$ at generated action positions
      end for
      Evaluate $\widehat{\mathcal{K}}_{\mathrm{ref},\mathcal{M}}$ using p_θ and p_{ref} at generated action positions
      Compute the minibatch objective $\widehat{\mathcal{J}}_{\mathcal{M}}$ in Eq. (8)
      Clear accumulated gradients; backpropagate $-\widehat{\mathcal{J}}_{\mathcal{M}}$; take an optimizer step
    end for
  end for
end while
return θ

The selector distribution is $q_{\mathrm{old}}(\cdot \mid h_t) = \mathrm{softmax}(S^{\mathrm{old}}_t/\tau)$, where $S^{\mathrm{old}}_{t,i} = S_{\theta_{\mathrm{old}}}(r_i \mid h_t)$ scores each candidate. Each record D_g retains the histories, media, selected candidates, actions, tool observations and stopping status. For every turn, it also stores S^{old}_t, $b^{\mathrm{reason}}_t = \log q_{\mathrm{old}}(z_t \mid h_t)$ and $b^{\mathrm{action}}_{t,k} = \log p_{\mathrm{old}}(a_{t,k} \mid h_t, z_t, a_{t,<k})$.

Group-relative advantages. After collection, evaluate the reward $R_g = \textsc{Reward}(x, D_g)$ for every trajectory, including those that reach the turn limit, under the task's reward rule. Compute the reward mean $\overline{R}_x$ and standard deviation σ_x within G_x, rather than across different questions. We use $\widehat{A}_g = (R_g - \overline{R}_x)/\max(\sigma_x, \delta)$ with a small numerical floor $\delta > 0$. This implements the normalization in Section 3.3 while assigning zero policy-gradient advantage to a group with identical rewards; the KL term can still contribute. Rewards, advantages, sampled trajectories and rollout log-probabilities are detached and held fixed for all updates on the collected batch.

Policy optimization. Algorithm 2 uses the detached-competitor approximation in Eq. (12). At each optimizer step, it recomputes the selected candidate score and action-token log-probabilities using the current parameters. Each update recomputes the selected candidate score while reusing the competitor scores recorded at rollout collection. Define $\mathrm{LSE}(v) = \log \sum_i \exp(v_i)$ and let j_t satisfy $z_t = r_{j_t}$. For the selected candidate, the stable log-probability calculation is

$$u_{t,i} = \begin{cases} S_\theta(z_t \mid h_t)/\tau & i = j_t \\ \mathrm{stopgrad}(S^{\mathrm{old}}_{t,i}/\tau) & i \neq j_t \end{cases} \qquad \widetilde{b}^{\mathrm{reason}}_t = u_{t,j_t} - \mathrm{LSE}(u_t) \tag{7}$$

The selected score remains differentiable in both terms of $\widetilde{b}^{\mathrm{reason}}_t$.

Using the clipped surrogate ℓ_ϵ from Section 3.3, the minibatch objective is

$$\widehat{\mathcal{J}}_{\mathcal{M}} = \frac{1}{|\mathcal{M}|} \sum_{g \in \mathcal{M}} \sum_{t=0}^{T_g - 1} \left[\lambda_{\text{reason}}^{\text{RL}} \ell_\epsilon(\widetilde{\rho}_{g,t}^{\text{reason}}, \widehat{A}_g) + \sum_{k=1}^{|a_{g,t}|} \ell_\epsilon(\rho_{g,t,k}^{\text{action}}, \widehat{A}_g) \right] - \beta \widehat{\mathcal{K}}_{\text{ref},\mathcal{M}}(\theta) \tag{8}$$

Here $\widehat{\mathcal{K}}_{\text{ref},\mathcal{M}}$ evaluates the reference-policy KL regularizer on generated action-token positions only. As in Eq. (6), the surrogate terms are summed within each trajectory and averaged across trajectories, without an additional trajectory-length normalization.

Gradient and state handling. Each reasoning selection contributes exactly one categorical surrogate term. The selected candidate is trained through its categorical selection term. The action-token surrogate and token-level KL regularizer apply only to generated action tokens; inserted reasoning tokens, prompts, observations, and padding are masked from these token-level losses. They remain visible in the model's context. Optimization recomputes differentiable activations using the current parameters. Weight-dependent caches are rebuilt or invalidated when the rollout model is synchronized. Caches that depend on model weights must be invalidated or rebuilt when the rollout model is synchronized for the next collection.

B Selector-Gradient Approximation

B.1 Exact and Approximate Gradients

Exact gradient. For turn t with history h_t and a selected candidate z_t from the library $\mathcal{R} = \{r_1, \ldots, r_N\}$ using selection temperature $\tau > 0$, the normalized policy in Eq. (2) satisfies

$$\log q_\theta(z_t \mid h_t) = \frac{S_\theta(z_t \mid h_t)}{\tau} - \log \sum_{i=1}^{N} \exp\left(\frac{S_\theta(r_i \mid h_t)}{\tau}\right) \tag{9}$$

Differentiating gives

$$\nabla_\theta \log q_\theta(z_t \mid h_t) = \frac{1}{\tau} \left[\nabla_\theta S_\theta(z_t \mid h_t) - \sum_{i=1}^{N} q_\theta(r_i \mid h_t) \nabla_\theta S_\theta(r_i \mid h_t) \right] \tag{10}$$

With length-normalization exponent $\alpha = 1$,

$$\nabla_\theta S_\theta(r_i \mid h_t) = \frac{1}{K_i} \sum_{k=1}^{K_i} \nabla_\theta \log p_\theta(r_{i,k} \mid h_t, r_{i,<k}) \tag{11}$$

where $r_{i,<k}$ is the candidate prefix before token k. The exact gradient requires both forward and backward computation for **all** candidates, which can be very expensive.

Detached competitors. Let θ_{old} be the rollout parameters. We recompute the selected score at the current parameters and hold the competing scores at their rollout values, giving the approximation

$$\widetilde{q}_\theta(z_t \mid h_t) = \frac{\exp(S_\theta(z_t \mid h_t)/\tau)}{\exp(S_\theta(z_t \mid h_t)/\tau) + \sum_{r_i \neq z_t} \exp(S_{\theta_{\text{old}}}(r_i \mid h_t)/\tau)} \tag{12}$$

This avoids backward computation through competing candidates:

$$\nabla_\theta \log \widetilde{q}_\theta(z_t \mid h_t) = \frac{1 - \widetilde{q}_\theta(z_t \mid h_t)}{\tau} \nabla_\theta S_\theta(z_t \mid h_t) \tag{13}$$

At $\theta = \theta_{\text{old}}$ during the first gradient update after the rollouts are collected, the selected probability is exact. After an update, the stored competitor scores can differ from their current values.

B.2 SSR Rollouts Are On-Policy in GRPO

During rollout collection, parallel reasoning decoding scores the reasoning candidates under the rollout parameters θ_{old} and samples $z_t \sim q_{\text{old}}(\cdot \mid h_t)$. The harness inserts the selected candidate into the context, after which the model generates action tokens autoregressively from $p_{\text{old}}(\cdot \mid h_t, z_t)$. The retained behavior probabilities therefore correspond to the process that generated the trajectory: one categorical probability for the reasoning selection and token-level probabilities for the subsequent action. Selecting from the reasoning library defines the support of the reasoning policy, while inserting the selected text is deterministic conditional on that selection. Thus, rollout collection is on-policy with respect to the composite SSR behavior policy.

When a rollout batch is reused after parameter updates, the current policy differs from the behavior policy, as in standard GRPO. The reasoning- and action-level importance ratios in Eq. (5) compare the current probabilities with the fixed probabilities recorded during rollout collection. The detached-competitor approximation in Eq. (12) concerns the current selector probability and its gradient during optimization; it does not change the policy used to collect the rollouts. Each inserted reasoning span is consequently represented by one categorical selection term, while token-level action terms apply only to the autoregressively generated action tokens.

B.3 GSPO and SAPO with Reused Rollouts

Update rules. For trajectory g in rollout group G, let $\mathcal{U}_g$ index its categorical reasoning selections and generated action tokens. With T_g turns and action-token sequence $a_{g,t}$ at turn t, define $Z_g = |\mathcal{U}_g| = \sum_{t=0}^{T_g-1}(1 + |a_{g,t}|)$. For $u \in \mathcal{U}_g$, $\rho_{g,u}$ is the corresponding ratio in Eq. (5), and $\widehat{A}_g$ is the group-normalized trajectory advantage defined in Section 3.3. GSPO (Zheng et al., 2025b) forms the length-normalized sequence ratio

$$\overline{\rho}_g = \exp\left(\frac{1}{Z_g}\sum_{u\in\mathcal{U}_g}\log\rho_{g,u}\right) \tag{14}$$

and applies the clipped surrogate $\ell_\epsilon(\overline{\rho}_g, \widehat{A}_g)$ from Section 3.3 at the sequence level, with clipping threshold ϵ. SAPO (Gao et al., 2025) instead applies a smooth surrogate $f_\kappa(\rho)\widehat{A}_g$ to each ratio $\rho > 0$, with slope $\kappa > 0$ chosen according to the sign of $\widehat{A}_g$:

$$f_\kappa(\rho) = \frac{4}{\kappa[1+\exp(-\kappa(\rho-1))]} \qquad f'_\kappa(\rho) = \frac{4\exp(-\kappa(\rho-1))}{[1+\exp(-\kappa(\rho-1))]^2} \tag{15}$$

The derivative attenuates updates as the ratio moves away from one. Both comparisons normalize by trajectory length and average across trajectories, whereas Eq. (6) sums the within-trajectory terms. Inserted reasoning tokens, prompts, observations, and padding do not add to Z_g.

Second and subsequent updates. As discussed for Eq 12, from the second update onward, the rollouts are stale relative to the current policy. During repeated updates on a collected batch, competitor scores remain fixed at their rollout values. Differences from current competitor scores affect the reasoning ratios: these enter the GSPO sequence ratio in Eq. (14) and the SAPO attenuation in Eq. (15). The selector gradient uses the approximation in Eq. (13).

C Detailed Inference Cost

Let H be the cached history length, L the freeform reasoning length, N the library size, and K_i the length of candidate r_i. All lengths count tokens, including scored formatting tokens, and $K_{\text{tot}} = \sum_{i=1}^{N} K_i$ is the total number of candidate tokens scored. Equation 3 compares the parallel computational depth of freeform reasoning and SSR. We distinguish this quantity from the number of FLOPs, KV-cache memory usage, and inference latency.

FLOPs and KV-cache memory usage. For a fixed Transformer with causal self-attention, let γ_{tok} denote the approximate FLOPs per token for linear projections, feed-forward networks, and the

output projection, summed across layers. Let γ_{att} denote the approximate FLOPs per query–key pair for the attention computation, also summed across layers and heads. We count a multiplication and an addition as two FLOPs and retain the leading terms in the operation count. Let W_{free} and W_{SSR} denote the approximate FLOP counts for freeform generation and SSR candidate evaluation, respectively. With the shared history already stored in the KV cache, we have

$$W_{\text{free}} \simeq \gamma_{\text{tok}} L + \gamma_{\text{att}} \left(HL + \frac{L(L-1)}{2} \right) \tag{16}$$

$$W_{\text{SSR}} \simeq \gamma_{\text{tok}} K_{\text{tot}} + \gamma_{\text{att}} \left(HK_{\text{tot}} + \frac{1}{2} \sum_{i=1}^{N} K_i(K_i - 1) \right) \tag{17}$$

This represents self-attention where each candidate attends to the shared history and its own preceding tokens. Reusing the KV cache of the common prefix avoids recomputing history representations (Zheng et al., 2024). For fixed reasoning and candidate lengths, the ratio of attention FLOPs approaches K_{tot}/L as H increases.

Let m_{KV} denote KV-cache storage in bytes per token across all layers. If the common prefix is stored once and all candidate suffixes are retained, the KV-cache memory usage is approximately $m_{\text{KV}}(H + K_{\text{tot}})$ bytes. Storing a separate copy of the prefix for each candidate instead requires approximately $m_{\text{KV}}(NH + K_{\text{tot}})$ bytes. With the history stored once, candidate-suffix KV storage grows linearly with K_{tot}; for fixed candidate length, it grows linearly with the number of candidates.

Inference latency. For an execution stage s, let Δ_s denote its duration in seconds, F_s its FLOP count, and B_s the total bytes read from and written to device memory. Let Π be the peak floating-point performance in FLOPs/s at the chosen precision, and $\mathcal{B}$ the peak device-memory bandwidth in bytes/s. These resource limits imply

$$\Delta_s \geq \max \left\{ \frac{F_s}{\Pi}, \frac{B_s}{\mathcal{B}} \right\} \tag{18}$$

Here, arithmetic intensity is F_s/B_s, measured in FLOPs per byte.

Let Δ_{free} and Δ_{SSR} denote inference latency for generating and selecting the reasoning text, respectively. Define v_{dec} as the number of generated tokens per second and v_{score} as the number of candidate tokens scored per second during model execution. Let δ_{free} and δ_{SSR} denote additional runtime overheads excluded from those rates, such as candidate selection on the CPU. Then

$$\Delta_{\text{free}} = \delta_{\text{free}} + L/v_{\text{dec}} \qquad \Delta_{\text{SSR}} = \delta_{\text{SSR}} + K_{\text{tot}}/v_{\text{score}} \tag{19}$$

Then, for $L > 0$ and positive rates, SSR reduces inference latency if and only if

$$\frac{K_{\text{tot}}}{L} < \frac{v_{\text{score}}}{v_{\text{dec}}} + \frac{v_{\text{score}}}{L} (\delta_{\text{free}} - \delta_{\text{SSR}}) \tag{20}$$

Assuming the overhead difference is negligible, on the same hardware platform, this condition reduces to

$$\frac{K_{\text{tot}}}{L} < \frac{v_{\text{score}}}{v_{\text{dec}}} \tag{21}$$

To put this into perspective, for an NVIDIA RTX PRO 6000 GPU, generating 100 freeform reasoning tokens autoregressively takes around the same time of processing a 15,200-token library.

D Training and Evaluation Datasets

D.1 Training Datasets

We use the 3,859-example training mixture from TAPO/SenseNova-MARS (Dong et al., 2026; Chng et al., 2025), with the subset sizes and accuracy rewards shown in Table 5. The reward implementation is described in Appendix F.

FVQA-train. FactualVQA (Wu et al., 2025) combines factual image–question–answer pairs requiring visual entity recognition or associated factual knowledge, with a mixture of search-required and search-free questions. It trains the agent to identify when image or text retrieval is needed.

DeepEyes-4K. DeepEyes (Zheng et al., 2026) provides visual questions spanning fine-grained perception, chart understanding, and challenging reasoning, supporting learning to inspect relevant image regions. Our training mixture includes both multiple-choice (MCQ) and non-MCQ subsets.

VisualProbe-train. VisualProbe (Lai et al., 2026) contains visual search questions about small targets in high-resolution images with distractor objects. These questions encourage repeated inspection and exploration of image regions.

Table 5: Multimodal-search RL training data.

Subset	Samples	Accuracy reward
FVQA-train	1,894	LLM judge
DeepEyes-4K (non-MCQ)	425	LLM judge
Visual-Probe-train	912	LLM judge
DeepEyes-4K (MCQ)	628	Exact match
Total	3,859	–

D.2 Evaluation Benchmarks

We evaluate on seven publicly available multimodal search benchmarks that collectively span a broad range of retrieval difficulty, temporal freshness, and image resolution requirements. We use the benchmark versions distributed in the SenseNova evaluation collection (Chng et al., 2025).

MMSearch (Jiang et al., 2024) contains 300 manually collected questions spanning 14 subfields, of which 171 are accompanied by an image; its News area covers events from May to August 2024, testing the model's ability to ground visual observations in up-to-date factual knowledge. We follow the standard split of 171 questions used for system-level evaluation.

HR-MMSearch (Chng et al., 2025) is a knowledge-intensive benchmark comprising 305 questions paired with high-resolution (4K) images sourced from international news agencies (2025 events), spanning eight topical domains. Approximately 60% of questions are classified as hard, requiring three or more tool invocations to answer correctly.

FVQA-test (Wu et al., 2025) is a curated 1,800-sample test set consisting of three equal-sized subsets: FVQA-auto-vc (600 automatically constructed samples, each manually checked for correctness and disjoint from the training set), InfoSeek Human Split (600 samples with manually corrected annotations), and an expert-annotated subset (600 samples).

InfoSeek (Chen et al., 2023) is a visual question answering dataset for information-seeking questions that cannot be answered through commonsense reasoning alone. It is built on visual entity recognition and grounded in Wikidata, targeting fine-grained factual attributes that models do not reliably surface without retrieval. We sample 2,000 instances from its test split.

SimpleVQA (Cheng et al., 2025) focuses on factual accuracy, combining post-2023 VQA instances with expert-curated image-question pairs across 1,013 English samples.

LiveVQA (Fu et al., 2025) aggregates 3,602 image-question pairs from major news outlets, including CNN and BBC, spanning 14 thematic categories, and emphasizes temporal reasoning over recently published content.

MAT-Search (Liu et al., 2025) is a manually crafted and human-verified benchmark of 150 examples designed to evaluate agentic multimodal reasoning, requiring models to handle composite multi-hop queries, retrieve external information, and invoke tools effectively across varying reasoning depths.

E Reasoning Candidate Library

The default library contains six manually written reasoning candidates (Table 6). Each entry is a single short natural-language statement describing *how to make progress from the current state*. The library is fixed throughout training and evaluation.

Table 6: The six-candidate multimodal-search reasoning library.

Name	Reasoning candidate
GENERAL-KNOWLEDGE	I recognize the main subject of this image and I already know the answer from general knowledge, so I can answer the question directly.
ANSWER-FROM-IMAGE	The information needed to answer is clearly visible in the image itself, so I can determine the answer by reading it directly from what I see.
EXAMINE-IMAGE-DETAIL	The part of the image that matters for this question is small or hard to make out, so I should examine that region more closely before I decide.
IDENTIFY-BY-IMAGE	I cannot confidently tell what the specific entity in this image is from its appearance alone, so I should identify it by searching with the image itself before answering.
LOOKUP-FACT	I can tell what the entity in the image is, but I do not know the particular fact the question is asking about, so I should look that information up.
ANSWER-FROM-EVIDENCE	The details I have gathered so far already contain what the question is asking for, so I can now give the final answer.

F Training Details

Reference recipe and our adaptation. We adapt the multimodal-search setup of TAPO (Dong et al., 2026). The agent is trained directly with RL, without an SFT warm-up, using VeRL, a global batch size of 64, learning rate 1×10^{-5}, and 8 rollouts per GRPO group. Each trajectory permits at most 10 interaction rounds during training, 8,192 generated tokens per round, and 32,768 tokens across the full trajectory. If it exceeds any of these, it will be marked as incorrect. Text retrieval during training uses a local Wikipedia retrieval index to reduce cost and evaluation uses web search through Serp API.

Action space. The agent can choose among three information-acquisition tools and a terminal answer: (i) `image_search_tool`, which performs reverse-image search on the question image and returns image captions; (ii) `text_search_tool(query)`, which performs query-specified search and returns a Qwen3-32B-generated summary; (iii) `image_zoom_in_tool(bbox)`, which crops a bounding box and returns the resulting image patch; and (iv) a terminal `<answer>` action. At every non-terminal turn, the base protocol expects a thinking block followed by exactly one `<tool_call>` conforming to the JSON schema; SSR supplies the thinking block through candidate selection while retaining the same tool-call schema.

Reward. For RL, we use a combination of accuracy reward and format reward.

$$r_{\text{final}} = r_{\text{accuracy}} + 0.5 r_{\text{format}} \tag{22}$$

Non-multiple-choice questions are judged by Qwen3-VL-32B-Instruct with greedy decoding and temperature zero given the image, question, ground truth, and the final answer generated by the model. Multiple-choice questions use exact match. Format reward is model-dependent: all conditions have to generate valid `<think>` delimiters and correct tool call format; SSR models (including the index-only variant) additionally have to use the expected tool of each reasoning that they select for full format reward, otherwise they only get partial format reward for the correct tool call format; the autoregressive structured reasoning model gets the format reward only if it generates reasoning in the library with exact match.

G Reasoning Comparison: No Thinking, Freeform, and SSR

We compare no thinking, freeform reasoning, and SSR (ours) under GRPO with the Qwen3-VL-4B backbone.

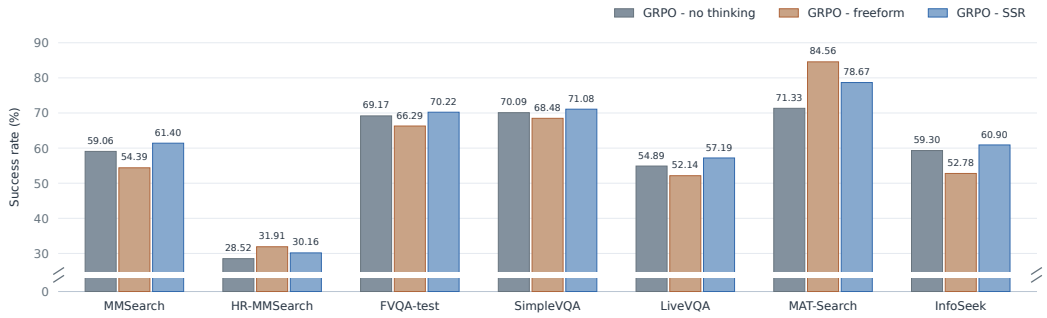

Figure 5: Success rate (%) of the three reasoning types on the multimodal search benchmarks.

Aggregate comparison. Figure 5 compares no thinking, freeform reasoning, and SSR under GRPO. The unweighted mean success rate across the seven benchmarks is 58.91%, 58.65%, and 61.37%, respectively. SSR therefore improves the benchmark mean by 2.47 percentage points over no thinking and 2.72 points over freeform reasoning. Freeform reasoning is 0.26 points below no thinking on this aggregate, so generating a freeform reasoning trace does not by itself consistently improve task performance in this comparison.

Consistent gains over no thinking. SSR outperforms no thinking on all seven benchmarks (7/7), with gains ranging from 0.99 to 7.34 percentage points. In contrast, freeform reasoning improves over no thinking on only two of seven benchmarks with the same Qwen3-VL-4B backbone under GRPO. SSR improves over no thinking across all seven benchmarks whereas freeform does not outperform no thinking on all of the benchmarks. SSR also exceeds freeform reasoning on five of seven benchmarks, with HR-MMSearch and MAT-Search favoring freeform reasoning.