

# SOAR: Supervision from Observation for Agentic Reinforcement Learning

Meng Li<sup>1\*</sup>, Lei Li<sup>2</sup>, Xiting Wang<sup>1†‡</sup>, Yi Yuan<sup>2</sup>, Zheng Wei<sup>2†</sup>, Jiang Bian<sup>2</sup>, Zang Li<sup>2</sup>

<sup>1</sup>Gaoling School of Artificial Intelligence, Renmin University of China,

<sup>2</sup>Platform and Content Group, Tencent

{mengli.24,xitingwang}@ruc.edu.cn, hemingwei@tencent.com

## Abstract

Agentic reinforcement learning enables large language models to solve long-horizon tasks by interacting with the environment and internalizing tool-use behavior into their reasoning. Prior work assigns supervision primarily based on outcome rewards or external reward models, but largely ignores environment observations, a critical source of learning. Consequently, agents may identify successful actions without understanding how the environment responds, producing suboptimal policies. To address this, we propose SOAR (Supervision from Observation for Agentic Reinforcement Learning), which assigns positive advantages to observation tokens proportional to the negative entropy of preceding actions. This encourages the agent to learn from outcomes of confident actions, grounding policy updates in environment dynamics and improving anticipation of tool-call consequences. Empirical results across three domains and 13 benchmarks show that SOAR consistently improves performance, yielding gains of up to 7.0% on general reasoning tasks and 16.9% on deep research tasks, while reducing erroneous and inefficient tool usage.<sup>1</sup>

## 1 Introduction

Recent advances in large language model (LLM)-based agents have enabled effective solutions to long-horizon tasks, including deep research (OpenAI, 2025), automated presentation generation (Zheng et al., 2025a), and autonomous large-scale code debugging (Cursor, 2025; Anthropic, 2025a). Unlike static LLMs, these agents operate within iterative action-observation loops, using multi-step

\*Work done during internship at Tencent.

†Corresponding Authors.

‡Work done at Beijing Key Laboratory of Research on Large Models and Intelligent Governance, and Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE.

<sup>1</sup>Code is available at <https://github.com/Mimasss2/SOAR>.

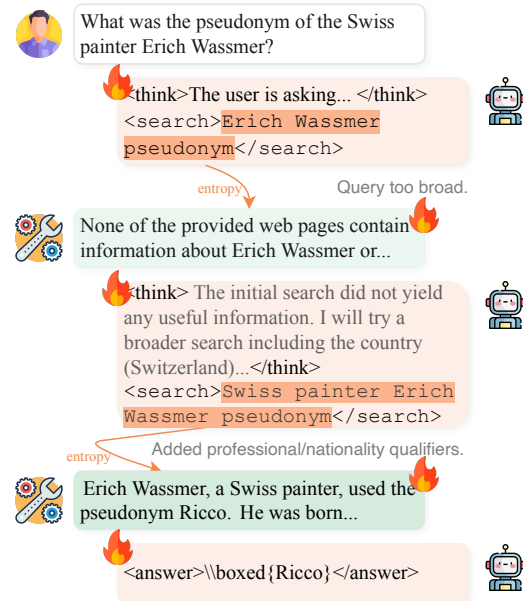


Figure 1: SOAR treats observations as learning signals, not passive context: instead of assigning them zero advantage, it rewards them positively in proportion to the negative entropy of the preceding action.

tool use to adapt to dynamic and partially observable environments (Zhang et al., 2025b; Cheng et al., 2025b). To move beyond fixed, heuristic-based workflows, Reinforcement Learning (RL) has emerged as the primary paradigm for training agentic foundation models (Anthropic, 2025b; Liu et al., 2025), especially in the era of experience (Silver and Sutton, 2025). In the agentic setting, classical RL is extended by treating language-based reasoning and tool-use actions as part of the policy, while modeling environment feedback as observations, enabling agents to learn adaptive, multi-step decision-making strategies through interaction (Jin et al., 2025b; Wang et al., 2025b).

Despite its promise, agentic RL faces unique efficiency bottlenecks. Agentic rollouts are long and computationally expensive to generate (Nguyen et al., 2025; Team et al., 2025a), while supervision

remains largely sparse and outcome-based (Wang et al., 2025b), leading to low per-token learning efficiency. Existing credit assignment methods attempt to alleviate this by redistributing sparse rewards across intermediate steps (Dong et al., 2025b; Wang et al., 2025a) or by introducing external reward models (Zheng et al., 2025b; Rahman et al., 2025). However, these approaches overlook a critical source of supervision: environment observations themselves. Observation tokens, which constitute a considerable fraction of each rollout, are typically assigned zero advantage. As a result, the policy may learn which actions lead to success without sufficiently modeling how the environment responds (Silver and Sutton, 2025). This contrasts with human learning, where unexpected observations provide strong feedback that refines one’s understanding of underlying dynamics.

Leveraging observations during RL is non-trivial. During exploration, a partially trained agent may take noisy or sub-optimal actions, resulting in observations that do not contribute to a coherent policy. In light of this, we propose **SOAR** (Supervision from Observation for Agentic Reinforcement Learning). SOAR explicitly incorporates environment observations into the policy optimization objective by treating them as active learning signals rather than passive context. The core mechanism of SOAR is an entropy-aware optimization strategy: instead of assigning zero advantages to observations, we assign them positive advantages proportional to the negative entropy of the preceding action, as shown in Fig. 1. The intuition is that low action entropy reflects higher policy confidence and consistency in action selection; observations following such actions are therefore more likely to reflect meaningful and predictable environment responses, making them more reliable learning signals for internalizing environment dynamics. By optimizing the likelihood of these observations, SOAR encourages the policy to internalize environment dynamics and anticipate consequences. This dense supervision alleviates reward sparsity and reduces unproductive tool calls by grounding the agent’s thoughts in environment dynamics. Extensive experiments demonstrate SOAR’s improvement across diverse agentic settings, including reasoning and deep search tasks.

Our contributions are threefold:

- We provide a diagnostic analysis of policy uncertainty in agentic RL, where the policy model

obtains more deterministic actions but reveals limited internalization of environment dynamics.

- We propose **Supervision from Observation for Agentic Reinforcement Learning (SOAR)**, which directly optimizes observation tokens to enhance the agent’s awareness of action consequences.
- Extensive experiments across 3 domains and 13 benchmarks demonstrate the effectiveness of SOAR for agentic RL, yielding performance gains of up to 16.9% alongside reduced observation uncertainty and more productive tool usage.

## 2 Related Work

**Agentic Reinforcement Learning.** Agentic reinforcement learning extends classical RL to multi-step interactive settings, where large language models interleave reasoning, tool use, and environment feedback to solve long-horizon tasks. Recent agentic foundation models, including Kimi-K2 (Team et al., 2025a), GLM-4.6 (Z.ai, 2025), and DeepSeek-V3.2 (Liu et al., 2025), leverage RL to improve tool invocation and decision-making in dynamic environments, and agent orchestration as well (Ruan et al., 2026). Representative frameworks such as Search-R1 (Jin et al., 2025b), RAGEN (Wang et al., 2025b), and ReTool (Feng et al., 2025) rely primarily on rule-based or outcome-driven rewards, resulting in sparse and delayed supervision over long trajectories. To mitigate this, prior work either synthesizes more challenging interaction data (Wu et al., 2025a; Qian and Liu, 2025; Tang et al., 2025; Xu et al., 2026) or redistributes outcome-based rewards across intermediate steps and introduces rubric-based process supervision (Wang et al., 2025a; Dong et al., 2025b; Zheng et al., 2025b; Rahman et al., 2025). Yet these methods still derive supervision indirectly from outcomes or heuristics. In contrast, SOAR integrates environment observations optimization in addition to policy-generated token, enabling more explicit learning of action consequences.

**Learning from experience.** Despite substantial progress from training on large-scale human-generated or curated corpora, such approaches primarily enable models to approximate human behavior and are unlikely to yield superhuman intelligence in isolation (Silver and Sutton, 2025). Progress beyond this regime requires training signals that evolve with the agent’s competence, motivating learning from an agent’s own experience through interaction with environments. Experience-

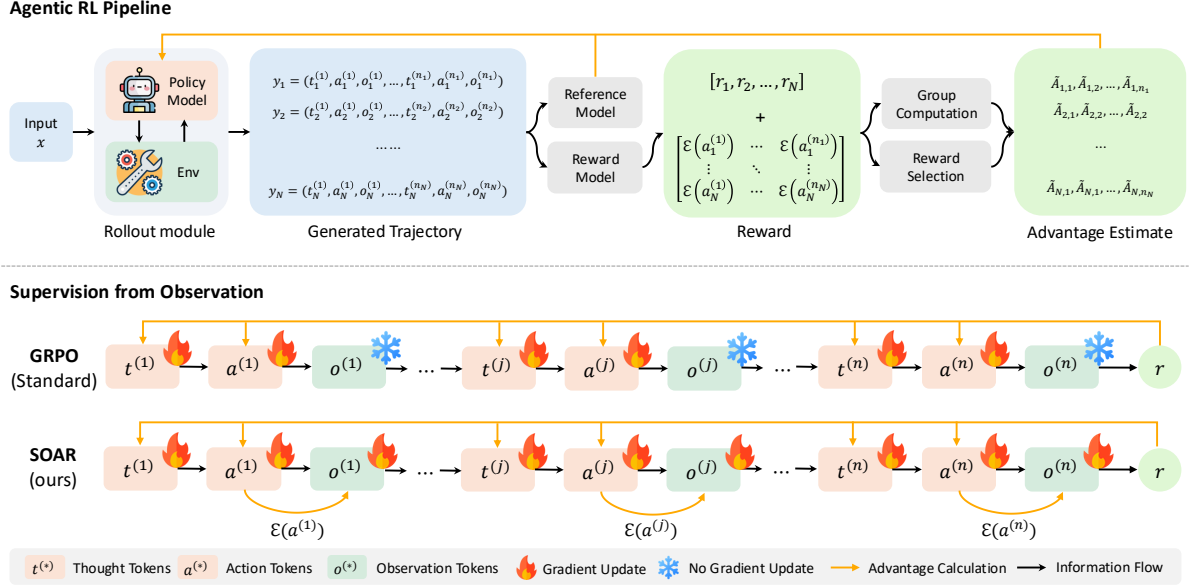


Figure 2: Overall framework of SOAR. (Top) Action entropy modulates rewards as an observation-aware advantage, with batch-wise relative reward normalization and sequence-wise relative entropy normalization. (Bottom) Observation tokens receive an auxiliary advantage proportional to the preceding action’s negative entropy, improving credit assignment for informative environment interactions.

based learning exposes agents to novel observations, state transitions, and feedback induced by their actions (Sutton et al., 1999). Representative successes include self-play systems such as AlphaGo and AlphaZero (Silver et al., 2016, 2017), iterative refinement in AlphaFold (Jumper et al., 2021), and interactive GUI agents (Zhang et al., 2025a). More recent work explores using LLM-based agents to rewrite or synthesize experiential observations to model environment dynamics (Zhang et al., 2025c; Chen et al., 2025) or experience selection (Gong et al., 2026; Xu et al., 2025). Motivated by these limitations, SOAR directly leverages agent observations as supervision to improve anticipation of action consequences.

### 3 Method

We first introduce the key notations and preliminary for agentic reinforcement learning (Sec. 3.1). Then, we provide a diagnostic analysis on policy uncertainty during the reinforce learning phase (Sec. 3.2). Finally, we introduce Supervision from Observation for Agentic Reinforcement Learning, internalizing environment dynamics into the policy model (Sec. 3.3). Fig. 2 shows the overall framework.

#### 3.1 Notations and Preliminaries

We consider an agentic reinforcement learning setting in which a policy model interacts with an en-

vironment through tool calls. Given an input  $x$  from dataset  $D$  and a set of available tools, the policy produces a trajectory  $y$  following the ReAct paradigm (Yao et al., 2023). Specifically, a trajectory consists of a sequence of interaction rounds indexed by  $k = 0, 1, \dots, n$ :

$$y = \{(t^{(k)}, a^{(k)}, o^{(k)})\}_{k=0}^n, \quad (1)$$

where  $t^{(k)}$  denotes the reasoning trace generated by the policy,  $a^{(k)}$  denotes the action taken by the agent (either a tool invocation or a final answer), and  $o^{(k)}$  denotes the observation returned by the environment in response to  $a^{(k)}$ . Importantly, observations  $o^{(k)}$  are environment-generated tokens and are not sampled from the policy distribution. At the token level, we denote by  $y_j$  the  $j$ -th token in the flattened trajectory  $y$ , and by  $|y|$  the total number of tokens in the trajectory.

The goal of agentic reinforcement learning is to optimize a policy  $\pi_\theta$  that generates trajectories conditioned on the input  $x$ , that is:

$$\max_{\theta} \mathbb{E}_{x \sim D, y \sim \pi_\theta(\cdot|x)} [r(x, y) - \beta \mathbb{D}_{\text{KL}}(\pi_\theta(y|x) \parallel \pi_{\text{old}}(y|x))]. \quad (2)$$

Here  $r(x, y)$  denotes the task-level reward, and  $\mathbb{D}_{\text{KL}}(\cdot \parallel \cdot)$  regularizes the learned policy against a reference policy  $\pi_{\text{old}}$  to prevent excessive devia-

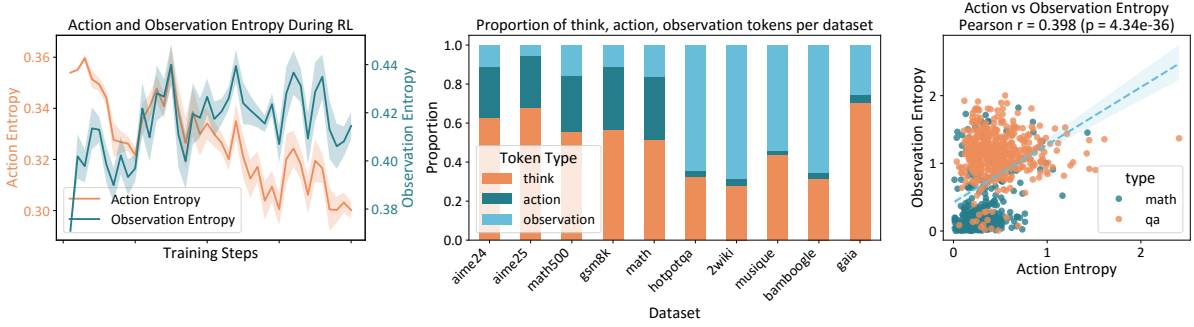


Figure 3: Diagnostic analysis of observation tokens on Qwen2.5-3B-Instruct. (Left) The entropy trend of actions and observations during agentic RL training. (Middle) Token type statistics across various datasets. (Right) Correlation analysis between action and observation tokens across tasks.

tion. The expectation is taken over both inputs and sampled trajectories.

To avoid reliance on explicit value-function approximation and to improve training efficiency for long-horizon agentic trajectories, Group Relative Policy Optimization (GRPO) is usually used to optimize the objective in Eq. (2) (Shao et al., 2024; Zheng et al., 2025c). For each input  $x$ , we sample a group of  $N$  trajectories  $\{y_1, \dots, y_N\}$  from the current policy and compute their corresponding rewards  $\{R_i\}_{i=1}^N$ , where  $R_i = r(x, y_i)$ . The relative advantage for trajectory  $y_i$  is estimated as:

$$\hat{A}_{i,j} = \frac{R_i - \text{mean}(\{R_k\}_{k=1}^N)}{\text{std}(\{R_k\}_{k=1}^N)} \mathbb{I}(y_{i,j}), \quad (3)$$

$$\text{where } \mathbb{I}(y_{i,j}) = \mathbf{1} \left\{ y_{i,j} \notin \bigcup_{k=0}^n \{o_i^{(k)}\} \right\}. \quad (4)$$

The indicator function  $\mathbb{I}(y_{i,j})$  ensures 0 advantage is assigned to observation tokens from the environment to exclude them from optimization.

Based on the advantage estimates, the policy is optimized using a clipped importance-sampling objective at the token level. The resulting loss function is given by:

$$\mathcal{L}_{\text{GRPO}} = \frac{1}{N} \sum_{i=1}^N \left[ \frac{1}{\sum_{j=1}^{|y_i|} \mathbb{I}(y_{i,j})} \sum_{j=1}^{|y_i|} \min \left( \rho_{i,j} \hat{A}_{i,j}, \text{clip}(\rho_{i,j}, 1 - \epsilon, 1 + \epsilon) \hat{A}_{i,j} \right) - \beta \mathbb{D}_{\text{KL}}(\pi_{\theta}(y_i | x) \| \pi_{\text{old}}(y_i | x)) \right], \quad (5)$$

where

$$\rho_{i,j} = \frac{\pi_{\theta}(y_{i,j} | x, y_{i,<j})}{\pi_{\text{old}}(y_{i,j} | x, y_{i,<j})} \quad (6)$$

is the importance sampling ratio for token  $y_{i,j}$ .

### 3.2 Diagnostic Analysis on Policy Uncertainty

In this section, we provide a diagnostic analysis of policy uncertainty during agentic reinforcement learning. We examine entropy dynamics of both action tokens and their resulting environment feedback, assessing whether on-policy training improves internalization of environment dynamics.

Prior work highlights the importance of entropy in reinforcement learning, where maintaining entropy is crucial for effective exploration and performance (Cheng et al., 2025a). As training progresses, policy optimization typically reduces action entropy, reflecting increasingly confident and deterministic behavior (Yue et al., 2025). However, whether lower action entropy corresponds to improved understanding of action consequences, i.e., environment feedback, remains an open question.

To address this, we analyze entropy dynamics of both action and observation tokens during agentic RL training. Let  $\mathcal{E}(\cdot)$  denote the entropy of a categorical distribution. For token position  $j$ , the policy entropy is defined as:

$$\mathcal{E}(y_j) = - \sum_v \pi_{\theta}(v | x, y_{<j}) \log \pi_{\theta}(v | x, y_{<j}), \quad (7)$$

where the summation is taken over the vocabulary. For an action  $a^{(k)}$ , represented as a contiguous token subsequence, we define its entropy as the average token entropy:

$$\mathcal{E}(a^{(k)}) = \frac{1}{|a^{(k)}|} \sum_{y_j \in a^{(k)}} \mathcal{E}(y_j). \quad (8)$$

Similarly, we compute the entropy of an observation  $o^{(k)}$  by averaging token-level entropies over the observation span.

The left panel of Fig. 3 shows the entropy dynamics of actions and observations during train-

ing. Action entropy  $\mathcal{E}(a^{(k)})$  decreases gradually, indicating increasing policy confidence in action selection. In contrast, observation entropy  $\mathcal{E}(o^{(k)})$  rises over time, revealing a growing uncertainty in environment responses despite on-policy optimization. This divergence suggests that the policy becomes more decisive in its actions without a commensurate improvement in anticipating their consequences, pointing to limited internalization of environment dynamics.

We further examine the composition of agent trajectories across datasets, as shown in the middle panel in Fig. 3. Observation tokens constitute a substantial fraction of total tokens, exceeding 50% in search-intensive benchmarks such as HotpotQA and Bamboogle. Given the high cost of on-policy rollouts, this indicates a major inefficiency in standard agentic RL: a large volume of tokens encoding rich environment feedback does not contribute to the learning signal.

Upon a closer examination of the entropy distributions of action and observation tokens in Fig. 3 (right), we found that the entropy scale varies across tasks. Thus a more fine-grained control of entropy is needed. Together, these findings suggest that existing agentic RL paradigms insufficiently leverage environment feedback, motivating the need for explicitly incorporating observation-level learning signals to improve environmental awareness.

### 3.3 Supervision from Observation for Agentic Reinforcement Learning

Motivated by the diagnostic analysis in Sec. 3.2, we observe that existing agentic RL paradigms primarily emphasize learning which actions are beneficial, while providing limited supervision over the consequences of those actions, i.e., the induced environment transitions. As a result, even under on-policy interaction, the learned policy may fail to internalize the underlying environment dynamics.

To address this, we propose *Supervision from Observation for Agentic Reinforcement Learning* (SOAR), which augments outcome-based RL with an auxiliary training signal on observation tokens. In addition to trajectory-level rewards for policy-generated tokens, SOAR assigns positive advantages to observation tokens, treating them as implicit feedback reflecting the consequences of preceding actions. This encourages the policy to anticipate environment responses, internalizing environment dynamics beyond optimizing final outcomes.

Naively assigning advantages to all observations can, however, may amplify noisy or uninformative feedback. To mitigate this, we scale observation-level advantages by an action-certainty signal, emphasizing observations that follow actions about which the policy is more confident. Since action certainty is only meaningful relative to other actions within a trajectory, as different tasks may have systematically different entropy scales (see Fig. 3 right), we normalize action entropy (following Eq. (8)) across interaction rounds within the same trajectory:

$$\Delta\mathcal{E}(a^{(k)}) = \frac{\mathcal{E}(a^{(k)}) - \text{mean}(\{\mathcal{E}(a^{(s)})\}_{s=1}^n)}{\text{std}(\{\mathcal{E}(a^{(s)})\}_{s=1}^n)}. \quad (9)$$

We now define the observation-aligned advantage at the token level. Outcome-based advantages are applied to policy-generated tokens as in GRPO, while entropy-weighted advantages are only assigned to observation tokens associated with preceding actions, providing denser credit assignment signals that help the model better interpret environmental feedback. Formally, the advantage for token  $y_{i,j}$  in trajectory  $y_i$  is given by:

$$\begin{aligned} \tilde{A}_{i,j} = & \underbrace{\frac{R_i - \text{mean}(\{R_k\}_{k=1}^N)}{\text{std}(\{R_k\}_{k=1}^N)} \mathbb{I}(y_{i,j})}_{\text{Outcome-based advantage}} \\ & + \underbrace{\lambda \sum_{k=1}^{n_i} \text{sg}[\sigma(-\Delta\mathcal{E}(a_i^{(k)}))] \mathbf{1}\{y_{i,j} \in o_i^{(k)}\}}_{\text{Observation-aligned advantage}}, \end{aligned} \quad (10)$$

where  $\mathbb{I}(y_{i,j})$  is the policy-token indicator defined in Eq. (4),  $\text{sg}[\cdot]$  denotes the stop-gradient operator.  $\sigma(z) = 1/(1 + e^{-x})$  is the sigmoid function, and  $\lambda$  is a hyperparameter controlling the strength of the observation-aligned signal.

Using the observation-aligned advantage estimate, we optimize the policy with a clipped importance-sampling objective:

$$\begin{aligned} \mathcal{L}_{\text{SOAR}} = & \frac{1}{N} \sum_{i=1}^N \left[ \frac{1}{|y_i|} \sum_{j=1}^{|y_i|} \min \right. \\ & \left. \left( \rho_{i,j} \tilde{A}_{i,j}, \text{clip}(\rho_{i,j}, 1 - \epsilon, 1 + \epsilon) \tilde{A}_{i,j} \right) \right. \\ & \left. - \beta \mathbb{D}_{\text{KL}}(\pi_{\theta}(y_i | x) \| \pi_{\text{old}}(y_i | x)) \right], \end{aligned} \quad (11)$$

here normalization is performed over all tokens, as both policy-generated tokens and observation tokens now receive non-zero advantages.

Method	Mathematical Reasoning					Knowledge Reasoning						Overall	
	AIME 24	AIME 25	MATH 500	GSM 8K	MATH AVG	HQA	2Wiki	Mus	Bamb	GAIA	AVG		
<i>Qwen2.5-3B-Instruct</i>													
DA	3.3	0.0	54.4	62.0	64.8	36.9	6.6	11.2	4.4	8.6	2.3	6.6	21.8
TIR	<u>10.0</u>	10.0	60.2	60.2	70.0	42.1	11.9	14.4	8.1	13.0	6.7	10.8	26.5
+GRPO	<u>10.0</u>	<u>13.3</u>	<u>71.2</u>	<u>84.0</u>	<u>80.0</u>	<u>51.7</u>	<b>61.0</b>	<b>66.4</b>	<u>27.2</u>	<u>77.5</u>	<u>25.0</u>	<u>51.4</u>	<u>51.6</u>
+SOAR	<b>26.7</b>	<b>16.7</b>	<b>71.4</b>	<b>84.8</b>	<b>82.4</b>	<b>56.4</b>	<u>59.9</u>	<u>65.9</u>	<b>30.8</b>	<b>80.8</b>	<b>32.5</b>	<b>54.0</b>	<b>55.2</b>
<i>Qwen2.5-7B-Instruct</i>													
DA	3.3	10.0	70.8	90.0	83.6	51.5	11.4	17.7	9.9	25.6	5.6	14.0	32.8
TIR	20.0	20.0	72.8	72.8	81.2	53.4	17.7	14.1	20.1	6.8	14.1	14.6	34.0
+GRPO	<u>30.0</u>	<u>23.0</u>	<u>79.4</u>	<u>90.2</u>	<u>88.0</u>	<u>62.1</u>	<u>54.6</u>	<u>71.2</u>	<b>31.6</b>	<u>74.8</u>	<u>34.0</u>	<u>53.2</u>	<u>57.7</u>
+SOAR	<b>33.3</b>	<b>33.3</b>	<b>84.2</b>	<b>92.0</b>	<b>89.4</b>	<b>66.4</b>	<b>59.9</b>	<b>73.8</b>	<u>30.5</u>	<b>74.9</b>	<b>39.6</b>	<b>55.7</b>	<b>61.1</b>

Table 1: Evaluation results on mathematical and knowledge reasoning benchmarks. The best and second-best performances are indicated by **bold** and underlined text, respectively. DA stands for direct answering without tool calling, while TIR stands for tool-intergrated reasoning. Dataset abbreviations correspond to HotpotQA (HQA), Musique (Mus), and Bamboogle (Bamb).

## 4 Experimental Setup

This section outlines the experimental setup for evaluating SOAR. We describe the datasets spanning reasoning and deep search tasks, the base models and baseline algorithms, and the training and evaluation protocols.

**Training and Evaluation Datasets.** We evaluate our method on three categories of tasks:

- *Mathematical Reasoning:* This category includes GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), as well as competition-level benchmarks such as MATH500 (Lightman et al., 2023), AIME24 (AIMO, 2024), and AIME25 (OpenCompass, 2025).
- *Knowledge Reasoning:* We consider multi-hop question answering datasets including HotpotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023).
- *Deep Search:* These tasks require complex information retrieval and multi-step reasoning, including GAIA (Mialon et al., 2023), WebWalkerQA (Wu et al., 2025b), SealQA (Thinh Pham et al., 2025), and FRAMES (Krishna et al., 2025).

We train with 10K agentic RL dataset from (Dong et al., 2025a) for reasoning tasks and a 1.1K mixed hard search dataset for deep search (Dong et al., 2025b), more details can found in App. C.

**Base Models and Baselines.** We conduct agentic RL experiments using the Qwen family of

models. Specifically, we use Qwen2.5-3B/7B-Instruct for mathematical and knowledge reasoning tasks (Team et al., 2025b). For deep search tasks, we adopt Qwen3-8B/14B (Yang et al., 2025), which provide stronger long-horizon reasoning capabilities. Our method is agnostic to the underlying RLVR algorithm, as it only augments the token-level advantage estimates by assigning non-zero advantages to environment-generated observation tokens, which are masked out in standard formulations. In the main experiments, we build upon Group Relative Policy Optimization (GRPO) and study the effect of incorporating observation-aligned advantages.

**Training Setting.** All models are trained using the verl framework (Sheng et al., 2025). For each input, we sample  $N = 16$  trajectories from the current policy and compute a trajectory-level outcome reward based on the prediction’s F1 score against ground-truth answers, augmented with format-based penalties. We employ a Python executor as the code tool. For search, we use a local dense retriever (Jin et al., 2025b) when training on reasoning tasks, and a search engine for deep search training. In both cases, retrieved content is summarized before being fed back to the policy. Training proceeds for 2 epochs on reasoning tasks and 5 epochs on deep search tasks due to limited data. Across all experiments, we use a batch size of 128 and a learning rate of  $1 \times 10^{-6}$ . Additional implementation details are provided in Appx. C.

Method	GAIA			WebWalkerQA				SealQA			FRAMES	Overall	
	Lv.1	Lv.2	Lv.3	OA	Easy	Med.	High	OA	S.0	S.Hard			OA
<i>Qwen3-8B</i>													
DA	17.9	7.7	0.0	12.6	4.4	7.1	0.0	6.0	14.0	6.3	10.6	22.0	12.8
RAG	38.5	15.4	0.0	22.3	42.2	29.8	28.2	32.0	16.8	10.8	14.2	30.4	24.7
ReAct	35.9	11.5	<u>8.3</u>	20.4	17.8	19.0	15.5	17.5	19.6	4.5	13.0	38.0	22.2
Search-o1	38.5	15.4	0.0	22.3	26.7	23.8	22.5	24.0	28.7	9.9	20.5	41.4	27.1
+GRPO	<u>43.6</u>	<b>34.6</b>	<b>16.7</b>	<b>35.9</b>	<u>48.9</u>	<u>45.2</u>	<u>33.8</u>	<u>42.0</u>	<u>34.3</u>	<u>13.5</u>	<u>25.2</u>	<u>46.6</u>	<u>37.4</u>
+SOAR	<b>48.7</b>	<b>34.6</b>	<u>8.3</u>	<b>35.9</b>	<b>53.3</b>	<b>56.0</b>	<b>38.0</b>	<b>49.0</b>	<b>34.9</b>	<b>16.2</b>	<b>26.3</b>	<b>53.6</b>	<b>41.2</b>
<i>Qwen3-14B</i>													
DA	25.6	13.4	8.3	17.5	6.7	4.8	2.8	4.5	14.7	4.5	10.2	27.8	15.0
RAG	38.5	13.5	0.0	21.4	40.0	34.5	29.6	34.0	16.1	10.8	13.8	33.0	25.6
ReAct	30.8	21.2	<b>16.7</b>	24.3	24.4	27.4	19.7	24.0	17.5	9.0	13.8	<u>49.0</u>	27.8
Search-o1	<u>48.7</u>	21.2	<b>16.7</b>	31.1	42.2	41.7	28.2	37.0	27.3	<u>13.5</u>	21.3	45.4	33.7
+GRPO	43.6	<u>36.5</u>	<b>16.7</b>	<u>36.9</u>	<u>46.7</u>	<u>50.0</u>	<u>39.4</u>	<u>45.5</u>	<u>30.8</u>	11.7	<u>22.4</u>	46.6	<u>37.9</u>
+SOAR	<b>51.3</b>	<b>40.4</b>	8.3	<b>40.8</b>	<b>48.9</b>	<b>53.6</b>	<b>42.2</b>	<b>48.5</b>	<b>32.9</b>	<b>18.0</b>	<b>26.4</b>	<b>61.6</b>	<b>44.3</b>

Table 2: Evaluation results on deep search benchmarks. The best and second-best performances are indicated by **bold** and underlined text, respectively. DA stands for direct answering without calling tools. OA stands for Overall, S.0 and S.Hard represents the Seal-0 and Seal-Hard subset.

**Evaluation Setting.** During inference, we use a temperature of 0.6 and top- $p$  sampling with  $p = 0.95$  for all models. The python executor is used for code tool, and the Serper search engine<sup>2</sup> is employed for search tool. For retrieved web results, we additionally crawl the full webpages and summarize relevant content before returning observations to the agent. For knowledge reasoning tasks, we report F1 scores on the four QA datasets. For all other tasks, we adopt the LLM-as-a-Judge paradigm, using Qwen3-30B-A3B-Instruct-2507 as the judge model with the instruction in Fig. 13. Following prior work (Dong et al., 2025b), we extract final answers enclosed in `\box{ }` and report pass@1 across all benchmarks.

## 5 Experimental Results

In this section, we present the main experimental results across multiple domains (Sec. 5.1). We then conduct ablation studies to validate the contribution of individual components (Sec. 5.2). Finally, we provide quantitative analyses of training dynamics and tool invocation behavior (Sec. 5.3), with additional qualitative examples included in App. G.

### 5.1 Main Results

We report the main results in Tabs. 1 and 2, highlighting the consistent superiority of SOAR in

improving agentic performance across different model scales and backbone architectures.

**Overall performance improvement on reasoning tasks.** As shown in Tab. 1, SOAR achieves the best overall performance, improving absolute scores by 7.0% and 5.9% over GRPO. For the 7B model, it ranks first on 9 of 10 benchmarks, showing consistent gains across reasoning tasks. For 3B models, mathematical reasoning improves by 9.1%, indicating that leveraging environment observations enhances the use of the code interpreter and overall problem-solving.

**Performance enhancement on deep search tasks.** Results on deep search benchmarks are shown in Tab. 2. SOAR achieves the strongest overall performance for both 8B and 14B models, outperforming GRPO by 10.2% and 16.9%, respectively. On the FRAMES benchmark, gains reach 15.0% for 8B and 32.6% for 14B models. The observed 8.3% drop on GAIA Lv.3 is due to a single case in this small subset (12 samples), with minor differences between models arising from rounding and isolated rule application, and does not indicate a broader issue, with detailed analysis in App. G. Taken together, These results demonstrate that observation-level supervision from search tools enables the agent to better anticipate step-level outcomes, resulting in more effective search actions.

### Impact of agentic reasoning on deep search.

<sup>2</sup><https://serper.dev/>

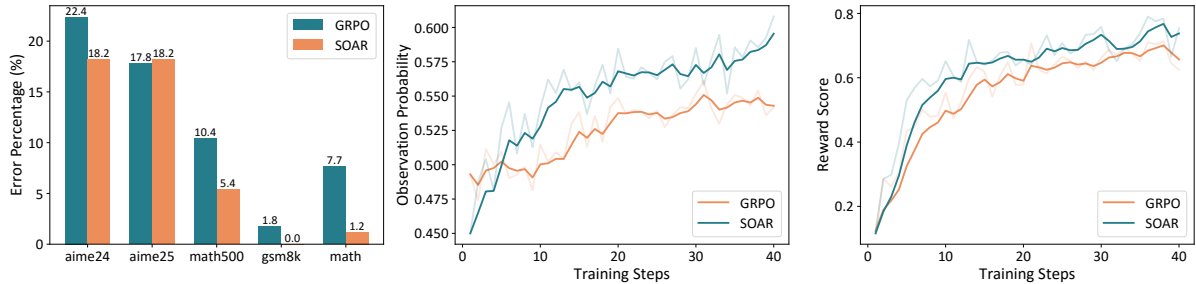


Figure 4: Quantitative Analysis. (Left) Comparison of Python tool call error rate. (Middle) Observation probability trend over training. (Right) Reward score trend over training.

Method	Math AVG	Know. AVG	Overall
SOAR	56.4	54.0	55.2
-Relative entropy	54.9	52.2	53.5
-Entropy weight	53.3	52.5	52.9
-Obs-aligned Adv	52.5	51.4	52.0

Table 3: Ablation results of SOAR on Qwen2.5-3B-Instruct on math and knowledge (Know.) reasoning tasks.

We analyze the impact of progressively introducing agentic reasoning into the inference pipeline. Compared to direct answering, RAG improves performance by  $1.7\times$  on 14B models. Incorporating ReAct increases it to  $1.87\times$ , and the Search-o1 pipeline, which enables additional reasoning over retrieved documents, further raises performance to  $2.3\times$ . These results validate the benefit of autonomous agentic pipelines over conventional direct-answering approaches for complex deep search tasks.

## 5.2 Ablation Study

We ablate the key design components of SOAR, including relative action entropy, entropy-modulated observation weighting, and observation-aligned advantage, and report the results in Tab. 3.

**Relative action entropy is critical for stable performance gains.** Replacing the relative action entropy term  $\Delta\mathcal{E}(a_i^{(k)})$  in Eq. (10) with the absolute action entropy  $\mathcal{E}(a_i^{(k)})$  (“Relative Entropy” row) degrades performance by 1.8% on mathematical reasoning and 3.3% on knowledge reasoning. This drop arises because entropy magnitudes vary across domains (see Fig. 3 right), leading to imbalanced gradient updates. Relative normalization mitigates this by enabling domain-agnostic scaling of advantages.

### Entropy-modulated observation weighting

**improves learning efficiency.** We remove the entropy modulation term  $\sigma(-\Delta\mathcal{E}(a_i^{(k)}))$  in Eq. (10) and assign a uniform observation advantage  $\lambda$  to all observation tokens (“Entropy Weight”). This leads to a larger performance drop of 4.7% on mathematical reasoning and 3.0% on knowledge reasoning tasks. The result indicates that treating all observations equally is sub-optimal, as it fails to prioritize informative environment feedback.

**Removing supervision from observation yields the largest degradation.** Finally, we eliminate the observation-aligned advantage term in Eq. (10), reducing the method to standard GRPO (“Obs-Aligned Adv”). This variant exhibits the worst performance, with average decreases of 6.1% on mathematical reasoning and 4.8% on knowledge reasoning, corresponding to a 5.5% overall drop. This highlights the importance of explicitly leveraging environment observations as learning signals in agentic reinforcement learning.

## 5.3 Quantitative Analysis

In this section, we compare SOAR with GRPO along 4 dimensions: (1) tool-call error rate, (2) observation likelihood under the policy, (3) reward progression, and (4) test-time scaling. Overall, we show that SOAR not only improves final reward but also promotes a more principled understanding of action–consequence relationships.

**Supervision from observations yields fewer erroneous tool calls.** The left panel of Fig. 4 shows the percentage of Python tool calls that result in compilation errors for the 7B model. Training with SOAR reduces errors by 18.8% on AIME 24, 48.1% on MATH500, and 83.3% on MATH. These results indicate that direct supervision from execution-time observations, i.e., code outputs and error traces, provides a valuable learning signal, encouraging the generation of syntactically and semantically valid code and reducing uninformative

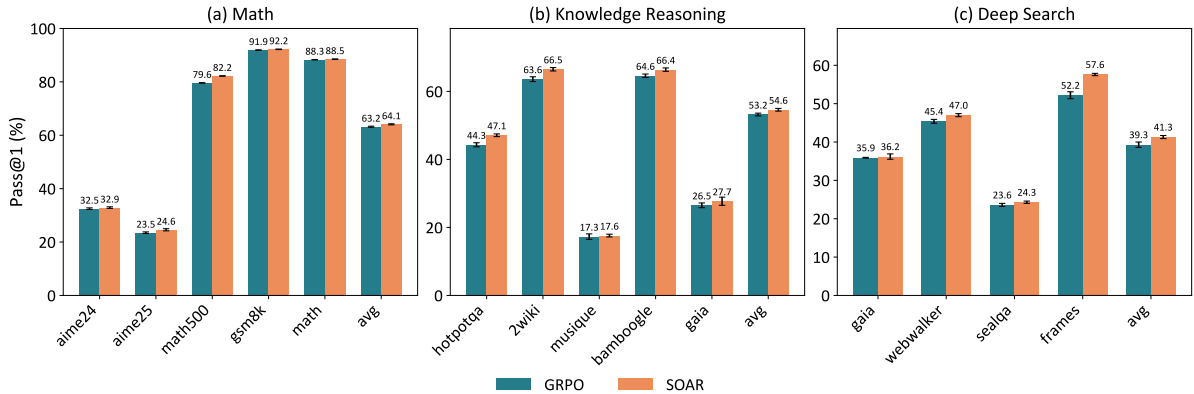


Figure 5: Benchmark performance of GRPO and SOAR under test-time scaling. We report pass@1 with standard deviation across benchmarks. Results are shown for Qwen2.5-7B on math and knowledge reasoning tasks, and for Qwen3-8B on deep search tasks.

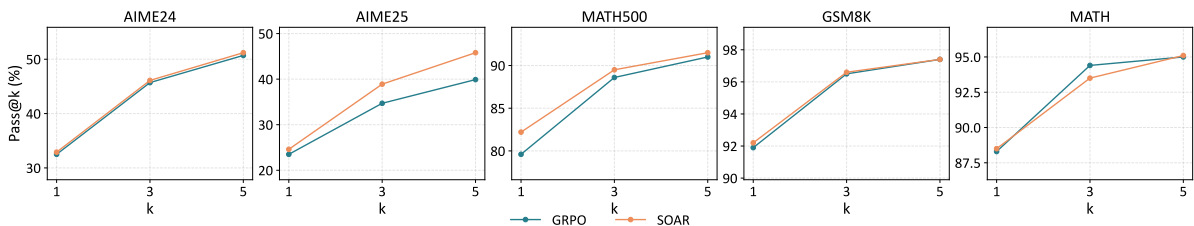


Figure 6: Math performance under different sampling budgets  $k$ . We report pass@ $k$  for  $k \in \{1, 3, 5\}$ . SOAR consistently improves performance across sampling budgets, with especially large gains on more challenging benchmarks such as AIME25, suggesting better sample efficiency and sample quality.

tool calls during inference.

**SOAR enhances environment awareness.** The middle panel in Fig. 4 shows that the likelihood of observed environment feedback under the policy increases more substantially with SOAR than with GRPO for the 8B model. This trend indicates that direct supervision from observations helps the model better anticipate environment responses, leading to more confident and informed action selection grounded in expected consequences.

**GRPO underperforms SOAR in reward optimization.** The right panel in Fig. 4 compares reward trajectories during training for the 8B model. SOAR consistently achieves higher reward scores than GRPO across training iterations, demonstrating improved learning efficiency. These results suggest that incorporating observation-level supervision yields not only better environment modeling but also stronger reward optimization, ultimately leading to more capable agent foundation models.

**Consistent gains under test-time scaling.** To evaluate SOAR under test-time scaling, we report pass@ $k$  and standard deviation in Figs. 5 and 6. For math benchmarks, we sample 32 outputs per problem and estimate pass@1/3/5. For knowledge reasoning and deep search benchmarks, where agentic inference is substantially more expensive, we

follow Team et al. (2025a) and sample 4 outputs per problem, reporting pass@1 and standard deviation. These results provide two key findings. First, SOAR consistently outperforms GRPO across sampling budgets while maintaining low standard deviations, indicating that the improvements are statistically stable rather than due to sampling noise. Second, SOAR improves both pass@1 and pass@ $k$ , suggesting that it enhances sample quality without sacrificing output diversity.

## 6 Conclusion

In this paper, we propose Supervision from Observation for Agentic Reinforcement Learning (SOAR), which incorporates environment observations as an explicit supervision signal. SOAR assigns positive advantages to observation tokens proportional to the negative entropy of preceding actions, encouraging the agent to learn from the consequences of confident decisions and ground actions in environment dynamics. This enables better anticipation of tool-call outcomes and stronger action–effect associations. We view SOAR as a step toward more efficient, environment-aware agentic RL and hope it motivates future work on richer observation-level supervision and credit assignment for language-based agents.

## Limitations and Future Work

Our experiments on reasoning tasks relied on a local dense retriever as the search tool. While more advanced retrieval methods, such as web search engines (Jin et al., 2025a), may further improve performance, we limited our setup due to the high cost of large-scale API calls. Exploring richer retrieval mechanisms is a promising direction for future work to enhance the model’s generalization capability.

We conducted experiments on relatively small models. Although these models provide meaningful insights, evaluating SOAR on larger-scale models (e.g., hundreds of billions of parameters) would help assess its scalability and generalizability. Extending to larger models requires substantial computational resources but could further validate the robustness of our approach.

For evaluating answer correctness, we relied on Qwen3-30B-A3B-Instruct as the judge model due to the large volume of evaluation. While this enables automated evaluation at scale, it may introduce subtle biases compared to human judgments. Future work could explore more robust evaluation protocols, including unbiased model-based judges or human-in-the-loop assessments, to further strengthen reliability.

## Ethical Considerations

We conducted basic checks to ensure the datasets used in our experiments were appropriate for training agentic foundation models. While efforts were made to minimize unsuitable content, a more comprehensive dataset curation and filtering process was beyond the scope of this work. Our experiments primarily utilized English-language datasets, and the generalizability of our approach to non-English or culturally diverse contexts has not been fully evaluated, highlighting an important direction for future research. We also used an AI assistant to check for grammatical errors and to polish sentence clarity during manuscript preparation. The main objective of this study was to enhance agentic capabilities in aligned large language models; ongoing evaluation and interventions remain essential to maintain safety and responsible deployment.

## Acknowledgements

The authors would like to thank the colleagues at Tencent for their helpful feedback and discussions, and the anonymous reviewers for their valuable

suggestions. Xiting Wang and Meng Li were supported by the National Natural Science Foundation of China (NSFC) (NO. 62476279, NO. 92470205, NO.U2436209), Major Innovation & Planning Interdisciplinary Platform for the “Double-First Class” Initiative, Renmin University of China, the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China No. 24XNKJ18. Supported by fund for building world-class universities (disciplines) of Renmin University of China and Public Computing Cloud, Renmin University of China.

## References

- AI-MO. 2024. Aime 2024. <https://huggingface.co/datasets/AI-MO/aime-validation-aime>.
- Anthropic. 2025a. Claude code. <https://www.claude.com/product/claude-code>.
- Anthropic. 2025b. Introducing claude sonnet 4.5. <https://www.anthropic.com/news/claude-sonnet-4-5>.
- Zhaorun Chen, Zhuokai Zhao, Kai Zhang, Bo Liu, Qi Qi, Yifan Wu, Tarun Kalluri, Sara Cao, Yuanhao Xiong, Haibo Tong, and 1 others. 2025. Scaling agent learning via experience synthesis. *arXiv preprint arXiv:2511.03773*.
- Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025a. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*.
- Zichen Cheng, Ziyue Lin, Yihang Yang, Zhongyu Wei, and Siming Chen. 2025b. Interactive simulation and visual analysis of social media event dynamics with llm-based multi-agent modeling. *Visual Informatics*, page 100260.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Cursor. 2025. Cursor: The ai code editor. <https://cursor.com/>.
- Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*.
- Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. 2025a. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*.

- Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, and 1 others. 2025b. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*.
- Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*.
- Yifan Gong, Jing Yao, Xiting Wang, Xunlong Wang, Xiaoyuan Yi, and Xie Xing. 2026. Influence-based online experience selection for effective rlhf. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. [Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625. International Committee on Computational Linguistics.
- Jian Hu. 2025. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*.
- Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan O Arik, and Jiawei Han. 2025a. An empirical study on reinforcement learning for reasoning-search interleaved llm agents. *arXiv preprint arXiv:2505.15117*.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025b. [Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning](#). In *Second Conference on Language Modeling*.
- John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Židek, Anna Potapenko, and 1 others. 2021. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589.
- Satyapriya Krishna, Kalpesh Krishna, Anhad Mohanane, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. 2025. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4745–4759.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.
- Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, and 1 others. 2025a. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*.
- Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025b. [Search-o1s: Agentic search-enhanced large reasoning models](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438, Suzhou, China. Association for Computational Linguistics.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.
- Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.
- Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.
- Xuan-Phi Nguyen, Shrey Pandit, Revanth Gangi Reddy, Austin Xu, Silvio Savarese, Caiming Xiong, and Shafiq Joty. 2025. Sfr-deepresearch: Towards effective reinforcement learning for autonomously reasoning single agents. *arXiv preprint arXiv:2509.06283*.
- OpenAI. 2025. Deep research system card. <https://openai.com/index/deep-research-system-card/>.
- OpenCompass. 2025. Aime 2025. <https://huggingface.co/datasets/opencompass/AIME2025>.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.

- Hongjin Qian and Zheng Liu. 2025. [Scent of knowledge: Optimizing search-enhanced reasoning with information foraging](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Salman Rahman, Sruthi Gorantla, Arpit Gupta, Swastik Roy, Nanyun Peng, and Yang Liu. 2025. Spark: Stepwise process-aware rewards for reference-free reinforcement learning. *arXiv preprint arXiv:2512.03244*.
- Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.
- Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. Zero-offload: Democratizing billion-scale model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564.
- Jianhao Ruan, Zhihao Xu, Yiran Peng, Fashen Ren, Zhaoyang Yu, Xinbing Liang, Jinyu Xiang, Yongru Chen, Bang Liu, Chenglin Wu, and 1 others. 2026. Aorchestra: Automating sub-agent creation for agentic orchestration. *arXiv preprint arXiv:2602.03786*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. [Hybridflow: A flexible and efficient rlhf framework](#). In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys '25*, page 1279–1297, New York, NY, USA. Association for Computing Machinery.
- David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and 1 others. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.
- David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, and 1 others. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.
- David Silver and Richard S Sutton. 2025. Welcome to the era of experience. *Google AI*, 1.
- Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, Lei Fang, Zhongyuan Wang, and Ji-Rong Wen. 2025. [SimpleDeepSearcher: Deep information seeking via web-powered reasoning trajectory synthesis](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 13705–13720, Suzhou, China. Association for Computational Linguistics.
- Richard S Sutton, Andrew G Barto, and 1 others. 1999. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134.
- Qiaoyu Tang, Hao Xiang, Le Yu, Bowen Yu, Yaojie Lu, Xianpei Han, Le Sun, WenJuan Zhang, Pengbo Wang, Shixuan Liu, and 1 others. 2025. Beyond turn limits: Training deep search agents with dynamic context window. *arXiv preprint arXiv:2510.08276*.
- Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025a. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.
- Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025b. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Nguyen Nguyen Thinh Pham, Pratibha Zunjare, Weiyuan Chen, Yu-Min Tseng, and Tu Vu. 2025. Sealqa: Raising the bar for reasoning in search-augmented language models. *arXiv preprint arXiv:2506.01062*, 2.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.
- Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. 2025a. Information gain-based policy optimization: A simple and effective approach for multi-turn llm agents. *arXiv preprint arXiv:2510.14967*.
- Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, and 1 others. 2025b. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.
- Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, and 1 others. 2025a. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*.
- Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He,

- Deyu Zhou, Pengjun Xie, and 1 others. 2025b. Web-walker: Benchmarking llms in web traversal. *arXiv preprint arXiv:2501.07572*.
- Zhihao Xu, Rumei Li, Jiahuan Li, Rongxiang Weng, Jingang Wang, Xunliang Cai, and Xiting Wang. 2026. Unlocking implicit experience: Synthesizing tool-use trajectories from text. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- Zhihao Xu, Yongqi Tong, Xin Zhang, Jun Zhou, and Xiting Wang. 2025. Reward consistency: Improving multi-objective alignment from a data-centric perspective. *arXiv preprint arXiv:2504.11337*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.
- Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Z.ai. 2025. Glm-4.6: Advanced agentic, reasoning and coding capabilities. <https://z.ai/blog/glm-4.6>.
- Chaoyun Zhang, Shilin He, Jiayu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2025a. Large language model-brained GUI agents: A survey. *Transactions on Machine Learning Research*.
- Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, and 1 others. 2025b. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*.
- Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, and 1 others. 2025c. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*.
- Hao Zheng, Xinyan Guan, Hao Kong, Wenkai Zhang, Jia Zheng, Weixiang Zhou, Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2025a. PPTAgent: Generating and evaluating presentations beyond text-to-slides. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14413–14429, Suzhou, China. Association for Computational Linguistics.
- Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. 2025b. StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21816–21841, Suzhou, China. Association for Computational Linguistics.
- Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyang Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.
- Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025c. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 414–431, Suzhou, China. Association for Computational Linguistics.

## A Baselines

In this section, we introduce the three categories of baselines used for comparison: 1) direct model reasoning, 2) tool-enhanced methods, and 3) RL-based methods.

**Direct Reasoning.** For direct reasoning baselines, we provide each question directly to the language model and prompt it to reason thoroughly before producing a final answer. No external tools are made available, so the model must rely solely on its internal knowledge. The instruction used is shown in Fig. 9.

**Tool-enhanced Methods.** We consider three representative tool-enhanced approaches: standard retrieval-augmented generation (RAG), ReAct, and Search-o1. In standard RAG, the question is used as a query to retrieve relevant documents from a search engine, after which the model generates an answer conditioned on both the question and the retrieved content (Lewis et al., 2020), with the instruction in. ReAct extends standard RAG by enabling iterative reasoning, action, and observation, allowing the model to perform multi-round

searches and refine its information gathering (Yao et al., 2023). Both standard RAG and ReAct use a direct search API that returns the top-10 results, each consisting of a title, URL, and a short snippet. Beyond the ReAct paradigm, Search-o1 further crawls the retrieved webpages and employs an LLM-based summarizer to extract query-relevant information, yielding more detailed and structured evidence (Li et al., 2025b). The prompts used for these methods are provided in Figs. 10 and 11.

**RL-based Methods.** For RL-based baselines, we primarily consider GRPO in our main experiments due to its wide adoption in agentic RL (Wang et al., 2025b; Zheng et al., 2025c; Jin et al., 2025b) and because our method is agnostic to the underlying RL algorithm. In addition to GRPO, we evaluate compatibility with other representative RL-based methods, including REINFORCE++ (Hu, 2025) and ARPO (Dong et al., 2025b). REINFORCE++ extends the classic REINFORCE algorithm with variance-reduction techniques to yield more stable and efficient policy gradients. ARPO further extends GRPO to the agentic setting by introducing an entropy-based adaptive rollout mechanism that promotes exploration at high-uncertainty decision points following tool usage. Results for REINFORCE++ and ARPO and their combination with SOAR are reported in App. E.

## B Evaluation Benchmarks

For mathematical reasoning evaluation, we use 5 challenging dataset as follows:

- AIME24 (AI-MO, 2024): A competition-level math benchmark from the American Invitational Mathematics Examination. It contains 30 challenging problems spanning diverse domains including algebra, geometry.
- AIME25 (OpenCompass, 2025): A competition-level math benchmark consisting of 30 newly released problems from American Invitational Mathematics Examination in February 2025, including problems spanning algebra, geometry, combinatorics and so on.
- GSM8K (Cobbe et al., 2021): An elementary school level math benchmark consisting of 1.32k multi-step calculation problems that require 2-8 reasoning steps.
- MATH (Hendrycks et al., 2021): A comprehensive academic benchmark for mathematical reasoning, spanning fields such as calculus, algebra, and discrete mathematics.
- MATH500 (Lightman et al., 2023): A selected subset from the MATH dataset containing 500 high-difficulty math problems, with difficulty exceeding college-level math.

For knowledge reasoning evaluation, we use 4 multi-hop question-answering benchmarks:

- HotpotQA (Yang et al., 2018): A multi-hop question-answering benchmark built from Wikipedia, targeting evaluation of complex information retrieval and reasoning across documents.
- 2WikiMultihopQA (Ho et al., 2020): Another multi-hop question-answering benchmark based on structured and unstructured data, and additionally provides evidence information for a comprehensive evaluation.
- MuSiQue (Trivedi et al., 2022): A challenging multi-hop question-answering benchmark, with problems progressing from simple retrieval to deeper understanding of documents and logical reasoning.
- Bamboogle (Press et al., 2023): A question-answering benchmark requiring composition of multiple facts to arrive at a solution, evaluating both information retrieval and reasoning abilities.

For deep search evaluation, we adopt 4 challenging search-intensive benchmarks as follows:

- General AI Assistant (GAIA) (Mialon et al., 2023): A benchmark of 466 carefully designed real-world tasks that evaluate LLMs’ core capabilities, including reasoning, web browsing, and tool use, providing a framework for assessing general AI assistants. We use only the text-subset with 103 examples, with 39, 52, 12 samples for Lv.1, Lv.2, Lv.3 separately.
- WebWalkerQA (Wu et al., 2025b): A benchmark of 680 question-answer pairs designed to evaluate LLMs’ web traversal abilities, focusing on multi-hop reasoning over complex and dynamic web content. We use a subset with 200 examples following (Dong et al., 2025b), with 45, 84, 71 samples for Easy, Medium, Hard level problems separately. Among these questions, 125 are chinese questions, while the rest are english questions.
- FRAMES (Krishna et al., 2025): A comprehensive evaluation dataset for Retrieval-Augmented Generation systems with 824 samples, testing capabilities across factuality, retrieval accuracy, and reasoning.
- SealQA (Thinh Pham et al., 2025): This bench-

mark presents a challenge for evaluating search-augmented LLMs on fact-seeking questions, where web search often returns conflicting, noisy, or uninformative results. For our experiments, we use the Seal-0 subset, which contains 111 samples, and the Seal-Hard subset, which contains 143 samples excluding those in Seal-0. We omit the LongSeal subset, as our model has not been optimized for long-context scenarios.

To facilitate evaluation, we select the first 500 examples from datasets with a large volume of data.

## C Implementation Details

### C.1 Tool Configuration

We employ two categories of tools: coding tools and search tools. For coding, we use a Python interpreter with a timeout of 120 seconds. For search during training on reasoning tasks, we adopt a local dense retriever following (Jin et al., 2025b). The retriever is based on the 2018 Wikipedia corpus, which provides a large-scale and stable document collection. We use `intfloat/e5-base-v2` as the retriever backbone, retrieve the top-5 documents per query, and truncate each document to a maximum length of 800 tokens.

For deep search training and for evaluation across all datasets, we instead use a Google search engine. This design choice is motivated by the fact that many questions require access to more recent or domain-diverse documents that are not covered by Wikipedia alone, even with optimal query formulation. We use the Serper API to obtain Google search results, returning the top-10 results, each consisting of a title, URL, and snippet. Following retrieval, we crawl the webpages associated with each URL and apply a filtering step based on word-overlap F1 score between the webpage content and the snippet. The filtered content is treated as the final search result. Regardless of whether it originates from the local dense retriever or the web search engine, this search result is provided to an LLM, which extracts and summarizes query-relevant information and returns the condensed evidence to the agent for subsequent decision making following the instruction in Fig. 12.

### C.2 Inference Settings

We use `vLLM` (Kwon et al., 2023) under the Apache License 2.0 as the inference framework. During generation, we set the temperature to 0.6, top- $p$  to 0.95, and the repetition penalty to 1.1. We allow

at most 5 tool calls to the code interpreter and 10 tool calls to the search tool per sample. The overall time limit for processing a single sample is set to 1500 seconds.

### C.3 Training Configuration

We first conduct cold-start supervised fine-tuning (SFT) to empower the model with basic agentic capabilities, then we conduct reinforcement learning to further enhance and internalize its agentic capabilities.

In the SFT stage, we train the model on the ToolStar-54K dataset, which contains both mathematical problems and question-answering tasks (Dong et al., 2025a). Trajectories are sampled from Qwen2.5-3B-Instruct based on answer accuracy. We use LLaMA-Factory (Zheng et al., 2024) under the Apache License 2.0 as the training framework and train for 3 epochs with a learning rate of  $7 \times 10^{-6}$ , a batch size of 128, and a cosine learning rate scheduler with a warmup ratio of 0.1. Weight decay is set to 0.1.

After obtaining the SFT checkpoints, we perform reinforcement learning based on `verl` framework under the Apache License 2.0. For reasoning tasks, we train on 10k ToolStar examples (5k math and 5k question-answering) for 2 epochs with a learning rate of  $1 \times 10^{-6}$ . The training batch size is 128, with a maximum prompt length of 1536 and a maximum response length of 4096. For each prompt, we sample 16 rollout trajectories and compute relative rewards to estimate advantages.

For deep search tasks, we use a dataset of 1.1k hard search questions, containing 871 questions from SimpleDeepResearcher (Sun et al., 2025) and 200 questions from WebSailor (Li et al., 2025a). The model is trained for 5 epochs using the same learning rate of  $1 \times 10^{-6}$ . The maximum prompt length is increased to 2000 and the maximum response length to 10,000. We generate 16 rollout trajectories for 8B models and 12 trajectories for 14B models due to memory constraints.

Across all training runs, we set the hyperparameter  $\lambda$  in Eq. (10) to 0.2, which encourages learning from observations while preventing the observation loss term from dominating gradient updates. We set  $\beta = 0$  in both Eq. (11) and Eq. (5) to promote policy exploration. All experiments are conducted on 8 NVIDIA H20 GPUs, and we employ FlashAttention-2 (Dao, 2024) and DeepSpeed ZeRO-3 (Rasley et al., 2020; Ren et al., 2021) to accelerate training.

Judge Model	AIME24	AIME25	MATH500	GSM8K	MATH	Avg
Qwen3-30B-A3B	33.3	33.3	84.2	92.0	89.4	66.4
Claude-4.5-Sonnet	33.3	33.3	84.0	92.2	89.4	66.4
Gemini-2.5-Flash	33.3	33.3	83.2	92.0	89.4	66.2
GPT-5.2	33.3	33.3	84.0	92.0	89.2	66.4
<b>Max Diff</b>	<b>0.0</b>	<b>0.0</b>	<b>1.0</b>	<b>0.2</b>	<b>0.2</b>	<b>0.2</b>

Table 4: Robustness to judge model choice. We re-evaluate the same predictions using four different judge models. The performance is highly stable across judges, with at most 1.0 percentage point deviation on any benchmark.

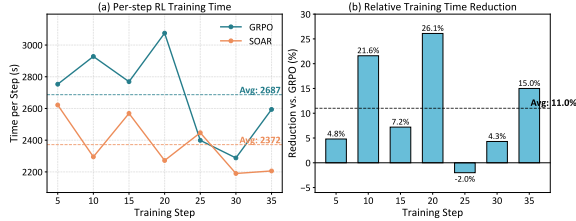


Figure 7: Per-step RL training time during Deep Search training on Qwen3-8B. Left: absolute training time measured at regular training checkpoints. Right: relative reduction in training time of SOAR compared with GRPO.

#### C.4 Reward Design

Rewards are computed based on both final answer accuracy and format correctness, following (Dong et al., 2025b). Final answer accuracy is measured using word-level F1 score, while format correctness is determined by whether the answer is wrapped in `\box{}`. Responses that achieve both a positive word-level F1 score and correct formatting are assigned higher rewards. The reward formula is as follows:

$$r(s, G) = (\mathbb{I}_{\text{fm}}(s) - 1) + \mathbb{I}_{\text{fm}}(s) \cdot \left[ \mathbb{I}(F_1 > 0)(F_1 + 0.1 \mathbb{I}_{\text{tc}}(s)) \right]. \quad (12)$$

Here,  $s$  is the answer extracted from `\box{}`,  $G$  is the ground truth answer, and  $F_1$  is the word-level F1 score between the two strings.  $\mathbb{I}_{\text{fm}}(s)$  represents whether the trajectory fulfills the format requirements, while  $\mathbb{I}_{\text{tc}}(s)$  represents whether the trajectory contains both tool call for search and code interpreter.

### D Further Quantitative Analysis

In this section, we conduct further quantitative analysis on the training time of SOAR, and the robustness to judge model choice for evaluation.

#### D.1 Training Time Comparison

To investigate the additional computation cost for SOAR compared to other baselines, we measure

the per-step RL training time in Fig. 7. We observe reduced overall training time because SOAR improves agent efficiency, leading to shorter and more effective trajectories with fewer redundant tool calls. Since rollout generation dominates agentic RL training cost, improved rollout efficiency reduces end-to-end training time. The additional entropy computation itself introduces negligible overhead, since, entropy is computed directly from token log-probabilities already produced during the forward pass, requiring only lightweight additional reductions per token. No additional model parameters, forward passes, or auxiliary networks are introduced. Thus, SOAR introduces negligible computational overhead.

#### D.2 Robustness to Judge Model Choice

In our setting, the judge is only used to verify whether the predicted answer matches the reference answer, which is a largely objective matching task with minimal subjective interpretation. To assess robustness, we re-evaluated results using multiple alternative judge models, including Claude-4.5-Sonnet, Gemini-2.5-Flash, and GPT-5.2, in addition to Qwen3-30B-A3B. As shown in Table 4, the reported performance is highly stable across judges. The maximum deviation across judge models is at most 1.0 percentage point on every benchmark, with an average maximum deviation of only 0.28 percentage points across tasks. Most differences are within 0.2 points. These results indicate that our conclusions are not sensitive to the choice of judge model. This evaluation protocol is also consistent with prior agentic RL work such as (Li et al., 2025b; Dong et al., 2025b).

### E Generalizability Analysis

In this section, we evaluate SOAR on more advanced RL algorithms, including Reinforce++ (Hu, 2025) and ARPO (Dong et al., 2025b), as reported in Tab. 5. When combined with SOAR, Reinforce++ and ARPO achieve consistent performance

<b>Method</b>	<b>Math AVG</b>	<b>Know. AVG</b>	<b>Overall</b>
ARPO	54.0	49.0	51.5
+SOAR	55.0	49.6	52.3
Reinforce++	52.0	48.4	50.2
+SOAR	55.4	50.3	52.9

Table 5: Generalizability to other RL algorithms.

improvements of 5.4% and 1.6%, respectively. For the baseline algorithms, we observed gradient explosion around 50–60% of the total training steps, so we selected the checkpoint immediately preceding this instability for evaluation. To ensure a fair comparison, we also selected the SOAR checkpoints at the corresponding training steps, even though training with SOAR does not exhibit gradient explosion or exhibit at later steps. Extending training with ARPO + SOAR to the full 2 epoch further boosts overall performance to 53.0. We expect that with additional RL stabilization techniques could yield even greater gains, which we leave for future work.

## F Prompt Templates

In this section, we demonstrate the instructions used for our experiments, including system prompts for training and inference in Figs. 8 to 11, instructions for information extraction model after a search tool call in Fig. 12, and the instruction used for LLM-as-Judge Fig. 13.

## G Case Study

We present examples of outputs from the optimized model on MATH, Bamboogle, and GAIA. We observe that models trained with SOAR exhibits more reasoning-consistent tool calls and more accurate tool call arguments.

In addition, we examined the GAIA Lv.3 subset and found that the observed performance drop (8.3%) is largely driven by a single case, which is more noticeable due to the small size of this subset (12 samples). In this instance, the 8B model shows a minor inconsistency in applying decision rules, while the 14B model produces a slightly different output because it used a rounded approximation for a quantitative value in the retrieved evidence. These differences are limited to this individual case and are not observed across other subsets. Overall, this variation reflects the inherent sensitivity of

small-sample evaluations rather than a substantive issue with model behavior or methodology.

### Training Template

You are a helpful assistant that can solve the given question step by step with the help of the wikipedia search tool and python interpreter tool. Given a question, you need to first think about the reasoning process in the mind and then provide the answer. During thinking, you can invoke the wikipedia search tool to search and python interpreter tool to calculate the math problem for fact information about specific topics if needed. The reasoning process and answer are enclosed within `<think>` `</think>` and `<answer>` `</answer>` tags respectively, and the search query and result are enclosed within `<search>` `</search>` and `<result>` `</result>` tags respectively. For example, `<think>` This is the reasoning process. `</think>` `<search>` search query here `</search>` `<result>` search result here `</result>` `<think>` This is the reasoning process. `<python>` python code here `</python>` `<result>` python interpreter result here `</result>` `<think>` This is the reasoning process. `</think>` `<answer>` The final answer is `\box{answer here}` `</answer>`. In the last part of the answer, the final exact answer is enclosed within `\box{ }` with latex format.

Figure 8: Instruction used for training.

### Direct Reasoning Instruction

You are a helpful assistant that can solve the given question step by step. Given a question, you need to first think about the reasoning process in the mind and then provide the answer. The reasoning process and answer are enclosed within `<think>` `</think>` and `<answer>` `</answer>` tags respectively. For example, `<think>` This is the reasoning process. `</think>` `<answer>` The final answer is `\box{answer here}` `</answer>`. In the last part of the answer, the final exact answer is enclosed within `\box{ }` with latex format.

Figure 9: Instruction used for testing standard reasoning.

### Standard RAG Instruction

You are a knowledgeable assistant that utilizes the provided documents to answer the user's question accurately.

Guidelines:

- Analyze the provided documents to extract relevant information. Synthesize the information to formulate a coherent and accurate answer.
- Ensure that your response directly addresses the user's question using the information from the documents.

Given a question and the provided reference documents, you need to first think about the reasoning process in the mind and then provide the answer. The reasoning process and answer are enclosed within `<think>` `</think>` and `<answer>` `</answer>` tags respectively. For example, `<think>` This is the reasoning process. `</think>` `<answer>` The final answer is `\box{answer here}` `</answer>`. In the last part of the answer, the final exact answer is enclosed within `\box{ }` with latex format.

Figure 10: Instruction used for testing standard RAG.

### ReAct and Search-o1 Instruction

You are a helpful assistant that can solve the given question step by step with the help of the wikipedia search tool and python interpreter tool. Given a question, you need to first think about the reasoning process in the mind and then provide the answer. During thinking, you can invoke the wikipedia search tool to search and python interpreter tool to calculate the math problem for fact information about specific topics if needed. The reasoning process and answer are enclosed within `<think>` `</think>` and `<answer>` `</answer>` tags respectively, and the search query and result are enclosed within `<search>` `</search>` and `<result>` `</result>` tags respectively. For example, `<think>` This is the reasoning process. `</think>` `<search>` search query here `</search>` `<result>` search result here `</result>` `<think>` This is the reasoning process. `</think>` `<python>` python code here `</python>` `<result>` python interpreter result here `</result>` `<think>` This is the reasoning process. `</think>` `<answer>` The final answer is  $\boxed{\text{answer here}}$  `</answer>`. In the last part of the answer, the final exact answer is enclosed within  $\boxed{\text{ }}$  with latex format.

Figure 11: Instruction used for testing Search-o1.

### Search Result extraction Instruction

#### Task Instruction:

You are tasked with reading and analyzing web pages based on the following inputs: **Current Search Query**, and **Searched Web Pages**. Your objective is to extract relevant and helpful information for **Current Search Query** from the **Searched Web Pages**.

#### Guidelines:

##### 1. Analyze the Searched Web Pages:

- Carefully review the content of each searched web page.
- Identify factual information that is relevant to the **Current Search Query** and can aid in the reasoning process for the original question.
- Ensure that the extracted information is accurate and relevant.
- DO NOT add any new knowledge that does not exist in the searched web pages.

##### 2. Output Format:

- Present the helpful information for current search query: beginning with ‘**Final Information**’ as shown below.

#### Final Information

[Helpful information]

#### Inputs:

- **Current Search Query:**

{search\_query}

- **Searched Web Pages:**

{documents}

Now you should analyze each web page and find helpful information based on the current search query “{search\_query}” and previous reasoning steps.

Figure 12: Instruction used for extracting relevant information from search results.

### LLM-as-Judge Instruction

Given a Question and its Golden Answer, verify whether the Predicted Answer is correct. The prediction is correct if it fully aligns with the meaning and key information of the Golden Answer. Respond with “Correct” if the prediction is correct and “Incorrect” otherwise. Golden Answer may have multiple options, and matching any one of them is considered correct.

Question: {question}

Golden Answer: {labeled\_answer}

Predicted Answer: {pred\_answer}

Figure 13: Instruction used for comparing a prediction and a reference answer for LLM.

### Case 1: Reasoning-aligned code tool calling

**Question:** Round 6287215 to the nearest ten thousand.

**Answer:** 6290000

**GRPO:** `<think>` To round the number 6287215 to the nearest ten thousand, we need to follow these steps:...

Let's confirm this using Python code.

`<think><python>`

...

```
# Round the number to the nearest ten thousand
```

```
rounded_number = round(number, -5)
```

```
print(rounded_number)
```

```
</python>
```

```
<result>6300000</result>
```

`<think>`

It seems there was a slight misunderstanding in the initial explanation. When rounding to the nearest ten thousand, the correct approach is to look at the digit in the thousands place. Let's correct the code and explanation. The correct steps are: ...

Let's confirm this using Python code again.

`<think><python>`

...

```
# Round the number to the nearest ten thousand
```

```
rounded_number = round(number, -5)
```

```
print(rounded_number)
```

```
</python>
```

```
<result>6300000</result>
```

`<answer>`The output from the code confirms that the correct rounding of 6287215 to the nearest ten thousand is indeed 6300000. Therefore, the final answer is `6300000`.`</answer>`

**SOAR:**

`<think>` To round the number 6287215 to the nearest ten thousand, we need to look at the digit in the thousands place. If this digit is 5 or greater, we round up the ten thousands digit by one. If it is less than 5, we keep the ten thousands digit as it is.

Here's the step-by-step process:

...

Let's confirm this with Python code.

`<think><python>`

```
# Define the number
```

```
number = 6287215
```

```
# Round the number to the nearest ten thousand
```

```
rounded_number = round(number, -4)
```

```
print(rounded_number)
```

```
</python>
```

```
<result>6290000</result>
```

`<answer>`The number 6287215 rounded to the nearest ten thousand is `6290000`.`</answer>`

Figure 14: A case from the MATH dataset. The GRPO-trained model fails to produce Python code consistent with its reasoning, rounding a number incorrectly to a hundred thousand instead of to the nearest ten thousand. Even after recognizing the mistake, it does not correct the error. In contrast, the model trained with SOAR successfully aligns its code with its reasoning and solves the problem correctly, demonstrating improved reasoning-to-execution consistency.

## Case 2: Purpose-aligned tool calling

**Question:** As of August 1, 2024, what is the population of the writer of the “Culdcept Saga”’s birthplace? Write the answer to the nearest million, in characters.

**Answer:** Two million.

**GRPO:**

```
<think> Okay, let’s tackle this question step by step...</think>
<search>writer of Culdcept Saga</search>
<result>From the searched web pages, the writer (specifically the story and scenario writer) of
*Culdcept Saga* is **Tow Ubukata**...</result>
<think> Alright, so the writer is Tow Ubukata. Now, I need to find his birthplace...</think>
<python> ...
print("Searching for Tow Ubukata’s birthplace...")
birthplace = "Tokyo, Japan"
print(f"Tow Ubukata’s birthplace is birthplace.")
projected_population = 38
...</python>
<result>Searching for Tow Ubukata’s birthplace...</result>
<think> Wait, but I should verify the exact birthplace. Maybe the search results...</think>
<search>Tokyo population 2024 projection</search>
<result>..."The largest and most populated city on Earth is Tokyo, Japan, with a population of
approximately 37.1 million people in its urban area as of 2024."...</result>
<think> Okay, so after correcting myself, the accurate number based on the search results is 37.1
million, which rounds to 37 million...</think>
<answer>37</answer>
```

**SOAR:**

```
<think> Okay, let me try to figure this out step by step. ...
Let me do a quick search. Let me start by searching for the writer of Culdcept Saga.</think>
<search>writer of culdcept saga</search>
<result>Based on the searched web pages, particularly...the primary writer of the story and scenario
for *Culdcept Saga* is **Tow Ubukata**...</result>
<think> Okay, so according to the search results, the writer of Culdcept Saga is Tow Ubukata. Now
I need to find out where he was born...</think>
<search>Tow Ubukata birthplace</search>
<result>Based on the searched web pages, particularly...the birthplace of Tow Ubukata is confirmed
to be **Gifu Prefecture, Japan**. Specifically:...</result>
<think> Alright, so his birthplace is Gifu Prefecture, Japan. Now I need to find the population of
Gifu Prefecture as of August 1, 2024...</think>
<search>Gifu Prefecture population 2024</search>
<result>Based on the searched web pages, particularly **Web Page 1 (Wikipedia)** , the population
of Gifu Prefecture as of **April 1, 2025** , is **2,040,000** . This is the most recent official
population figure available ...</result>
<think> Hmm, the sources give conflicting numbers. Some say 2,040,000 as of April 2025, others
older figures like 1,931,212 in 2023....</think>
<answer>2</answer>
```

Figure 15: A case from the FRAMES dataset. The GRPO trained model cheat by using python code to print the internal knowledge instead of using search tool to acquire information related to the question, thereby produces the wrong answer. In contrast, the model trained with SOAR successfully searched all the relevant information required for solving the problem, demonstrating improved tool call productivity.

### Case 3: Search accuracy comparison

**Question:** In 2016, who was the host of the longest running talk show?

**Answer:** Jimmy Fallon

**GRPO:**

*<think>*... Let's break down my plan: 1. Search for lists of long-running talk shows, focusing on those active around 2016. 2. Identify the oldest among them. 3. Find the host(s) of that show.*</think>*

*<search>*list of long running talk shows started before 2016*</search>*

*<result>*From the searched web pages, the following long-running talk shows that began before 2016 are particularly relevant to identifying the "longest running talk show" as of 2016:...*</result>*

*<think>*The search results indicate that "The Tonight Show" is the longest-running talk show franchise. Under Johnny Carson, it ran from 1962 to 1992...Let's look for more recent lists or discussions specifically addressing the state of talk shows in 2016 regarding their historical standing.*</think>*

*<search>*which talk show was the longest running in 2016*</search>*

*<result>*...**"The Tonight Show"** is consistently identified across multiple sources (including Wikipedia, CNN, and academic analyses) as the **"longest-running talk show in American television history"**. It began on **"September 27, 1954"**...*</result>*

*<think>*The search results confirm that "The Tonight Show" is the longest-running talk show ever. ... That show was "The Tonight Show," and its primary host during its formative and longest-running periods was Johnny Carson.*</think>* **Johnny Carson** *</answer>*

**SOAR:**

*<think>* To determine who hosted the longest-running talk show in 2016, I will start by identifying which talk shows have been around for an extended period before looking into their hosts during that year.*</think>*

*<search>* search query: Which talk show has the longest history?*</search>*

*<result>*The Tonight Show is the longest-running talk show in television history...This establishes that The Tonight Show has the longest history among all talk shows, making it the definitive answer to the query.*</result>*

*<think>* After finding out that "The Tonight Show" holds the record for being the longest-running talk show due to its inception back in 1954, my next task would be determining whose role as host contributed towards maintaining such longevity specifically up until 2016.*</think>*

*<search>* search query: Who was hosting The Tonight Show in 2016?*</search>*

*<result>*Based on the searched web pages, specifically ... it is confirmed that **"Jimmy Fallon"** was hosting **"The Tonight Show"** in 2016...*</result>*

*<think>* Now having identified both pieces of necessary information—the longest-running talk show being "The Tonight Show", and the person who hosted it in 2016 as Jimmy Fallon—I am ready to formulate my response to the original question.*</think>*

*<answer>* **Jimmy Fallon** *</answer>*

Figure 16: A case from the Bamboogle dataset. The GRPO trained model only searched the show correctly, but extracted the host required wrongly, while the SOAR trained model identifies the host for 2016 correctly.