ARISE: ADAPTING TO EVOLVING CAPABILITY GAPS IN AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

As a long-horizon agent improves through experience, previously observed weaknesses may recede while new limitations emerge, continually changing what it still needs to learn. Yet the learning process often remains tied to a static view of these needs: fixed behavioral criteria and training priorities can become misaligned with evolving agent capabilities, while sparse task-level feedback makes such misalignment more difficult to detect. Even when capability gaps are identified, rollouts from the current policy may repeatedly reproduce the same failures rather than explore better alternatives. To address this, we introduce **A**daptive **R**ubr**i**c–**S**kill Co-**E**volution (ARISE), a reinforcement learning framework that uses rollout evidence to continually adapt evaluation criteria, exploration guidance, and training priorities. Rubrics evolve to reward partial behavioral progress, while their paired skills are refined and selectively activated to guide exploration toward unresolved weaknesses. Alongside this co-evolution, capability-based adaptive sampling prioritizes tasks that target behaviors needing further improvement. Experiments on two challenging long-horizon agent benchmarks, SkillsBench and Terminal-Bench, demonstrate that ARISE successfully enhances both overall task performance and training efficiency.

1 INTRODUCTION

Large Language Models (LLMs) increasingly act as agents that reason, invoke tools, and interact with environments to complete multi-step tasks (Yao et al., 2023; Yang et al., 2024; Feng et al., 2026). These long-horizon tasks require the coordination of capabilities such as planning, execution, and verification (Yao et al., 2023; Shinn et al., 2023). Reinforcement learning refines these behaviors through interactive experience (Ivison et al., 2026). Although recent methods enrich this process with behavioral feedback and reusable guidance (Chen et al., 2026b; Shao et al., 2026; Xia et al., 2026), structuring training for sustained capability development remains an open question.

Two coupled challenges hinder aligning training with evolving agent capabilities: (i) *Tracking evolving capability gaps.* Correcting an intermediate error may leave the task reward unchanged if subsequent steps prevent completion, masking shifts in behavioral bottlenecks (Lightman et al., 2024). Furthermore, even detailed criteria become uninformative once consistently satisfied, while out-of-scope bottlenecks remain unmeasured. (ii) *Exploring beyond repeated failures.* While precise evaluation distinguishes sampled behaviors, it cannot guarantee the generation of useful alternatives. Under group-relative optimization, identical task rewards within a rollout group yield zero advantage, even if trajectories differ in intermediate steps (Shao et al., 2024; Yu et al., 2025). Consequently, mastered or excessively difficult tasks waste rollout budgets without providing useful reward contrasts, making their learning value highly dependent on the policy's developmental stage (Florensa et al., 2018). Together, these challenges necessitate training paradigms that continually identify areas for improvement and foster the exploration and reinforcement of corresponding behaviors.

To address these challenges, we propose **A**daptive **R**ubr**i**c–**S**kill Co-**E**volution (ARISE), a reinforcement learning framework driven by evolving capability gaps (Figure 1(b)). It translates rollout evidence of these gaps into concrete behavioral requirements, each formalized as a rubric for evaluating partial progress and a paired skill for guiding action. Rubric pass rates then direct exploration: low rates trigger skill activation or refinement, whereas consistently high rates indicate mastery, allowing

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.

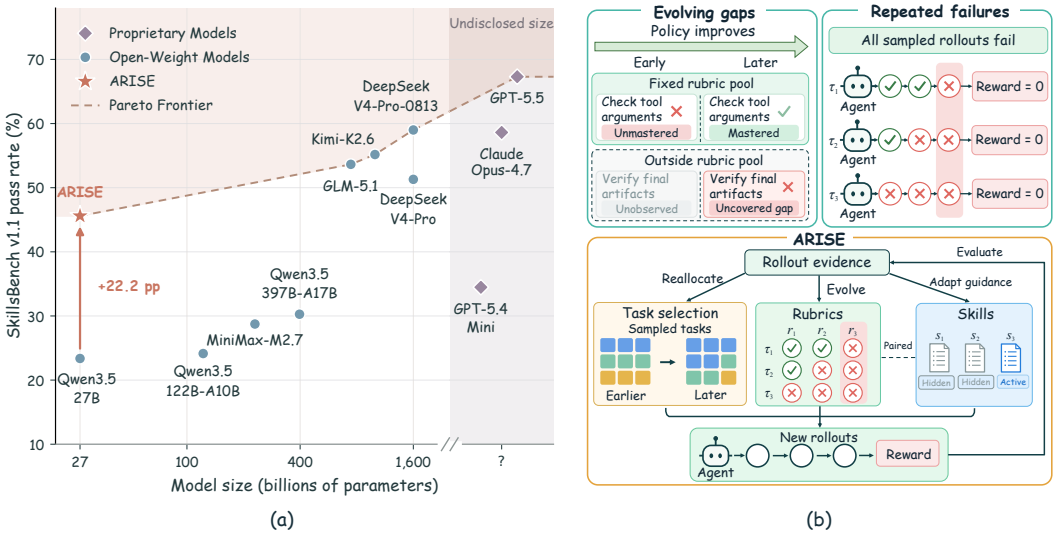


Figure 1: Performance and motivation of ARISE. (a) SkillsBench v1.1 pass rates versus model size, highlighting the improvement over Qwen3.5-27B and competitive performance with substantially larger models. (b) Fixed criteria can miss emerging capability gaps, while sampled rollouts may all fail despite making partial progress. ARISE uses rollout evidence to adapt rubrics, skill guidance, and task selection as agent capabilities evolve.

the criterion to be retired. As new evidence emerges, the framework expands this repertoire to cover unaddressed behaviors, ensuring the requirements remain responsive to policy development.

Progress on these requirements also depends on task context, as a behavior performed reliably in one task type may remain difficult in another. We therefore propose a capability-based adaptive sampler that guides task selection using rubric-based estimates of capability performance within each task type. Subsequent rollouts provide feedback for updating rubrics, paired skills, and the sampling distribution, ensuring that evolving capability assessments shape future training. As shown in Figure 1(a), ARISE significantly improves SkillsBench performance over its base model, achieving competitive results with much larger models.

In summary, our contributions are three-fold:

- We identify evolving capability gaps as a central challenge in long-horizon agent learning. Instead of relying on static behavioral requirements or training priorities, ARISE continuously leverages rollout evidence to dynamically uncover and address emerging bottlenecks.
- We propose **A**daptive **R**ubri**c**–**S**kill Co-**E**volution (ARISE), which couples capability identification, targeted exploration, and fine-grained behavioral feedback, together with capability-based adaptive sampling. More broadly, it establishes a general paradigm for aligning the learning process with evolving agent capabilities.
- Empirically, we demonstrate that adapting training to these evolving capabilities yields substantial and efficient performance gains. Across two agent benchmarks, ARISE consistently outperforms same-scale baselines, proving to be a highly effective alternative to simply scaling model size.

2 Related Work

Rubric-Based Reinforcement Learning. Rubric-based approaches deconstruct evaluation into explicit criteria, aggregating individual judgments to derive training rewards (Gunjal et al., 2026; Viswanathan et al., 2025). To make these evaluations adaptive, frameworks like OnlineRubrics (Rezaei et al., 2026) extract criteria by comparing current and reference outputs, while DR Tulu (Shao et al., 2026) continuously updates rubrics based on on-policy responses and search contexts. However, these response-level evaluations often overlook improvements in intermediate

actions that are not reflected in the final output. RuscaRL (Zhou et al., 2026) further uses rubrics as both rewards and exploration scaffolds, yet relies on fixed underlying criteria and decays guidance according to a predefined schedule. In contrast, ARISE evaluates intermediate agent behaviors and dynamically revises active criteria as rollout evidence reveals new or consistently mastered behavioral requirements.

Skill Learning for LLM-Based Agents. Skill learning enables LLM-based agents to reuse experience through executable routines or natural-language guidance (Wang et al., 2025; Cai et al., 2024). Voyager (Wang et al., 2024) builds a library of executable skills via environment interaction, while ExpeL (Zhao et al., 2024) distills transferable insights. By keeping policy parameters frozen, both methods rely entirely on the in-context capabilities of the base model to leverage this accumulated knowledge. SkillRL (Xia et al., 2026) extends experience reuse to reinforcement learning through a hierarchical skill library, evolving it based on validation success rates. It consistently includes general skills while retrieving task-specific ones via semantic relevance. However, relevance alone does not necessitate guidance, as a retrieved skill may describe behaviors the policy has already mastered. In ARISE, evaluations of paired behavioral criteria determine exactly when guidance is needed, providing actionable evidence to refine skills if specific weaknesses persist.

Adaptive Task Sampling. Adaptive task sampling allocates training experience according to the evolving learning needs of the policy (Jiang et al., 2021). GoalGAN (Florensa et al., 2018) selects goals of intermediate difficulty based on empirical success rates. In reinforcement learning for LLMs, DAPO (Yu et al., 2025) filters groups post-rollout to exclude those lacking reward variation, while VADE (Hu et al., 2026) selects informative samples pre-rollout using online correctness estimates. However, these outcome-based signals capture overall task difficulty without distinguishing the underlying behavioral causes: tasks with similar success rates may require improvements in entirely different capabilities. The distinction in ARISE lies in the evidence used for selection: rubric evaluations expose which behaviors remain weak within each task type, rather than only whether tasks succeed.

3 METHODOLOGY

3.1 PROBLEM FORMULATION

We consider an LLM-based agent that interacts with an environment to complete a task x drawn from a task distribution $\mathcal{D}$. Each task specifies an instruction, an execution environment, and the available tools. At interaction step t, the agent observes a history $h_t = (x, a_1, o_1, \ldots, a_{t-1}, o_{t-1})$ and samples an action $a_t \sim \pi_\theta(\cdot \mid h_t)$, where π_θ is the parameterized policy. An action is a textual response or a tool invocation, and o_t denotes the environment feedback resulting from a_t. The interaction terminates after T steps, producing a trajectory $\tau = (a_1, o_1, \ldots, a_T, o_T)$.

A task-specific verifier assigns an outcome reward $R_{\text{task}}(x, \tau)$ that measures task completion. The learning objective is to improve expected task performance:

$$\max_\theta \; J(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{\tau\sim P_\theta(\cdot|x)}\left[R_{\text{task}}(x,\tau)\right], \tag{1}$$

where $P_\theta(\cdot \mid x)$ is the trajectory distribution induced by the policy and environment.

3.2 OVERALL FRAMEWORK

To track evolving capability gaps and support exploration beyond repeated failures, ARISE combines two components: *rubric–skill co-evolution* and *capability-based adaptive sampling*. Together, they determine what behavioral feedback and guidance to provide and which tasks to train on. Figure 2 presents the overall framework, and Appendix B.1 summarizes the training procedure.

3.2.1 RUBRIC–SKILL CO-EVOLUTION

Rubric–skill co-evolution adapts what the agent is evaluated on and which guidance it receives as behavioral gaps change during training. Prompt templates are provided in Appendix B.6.

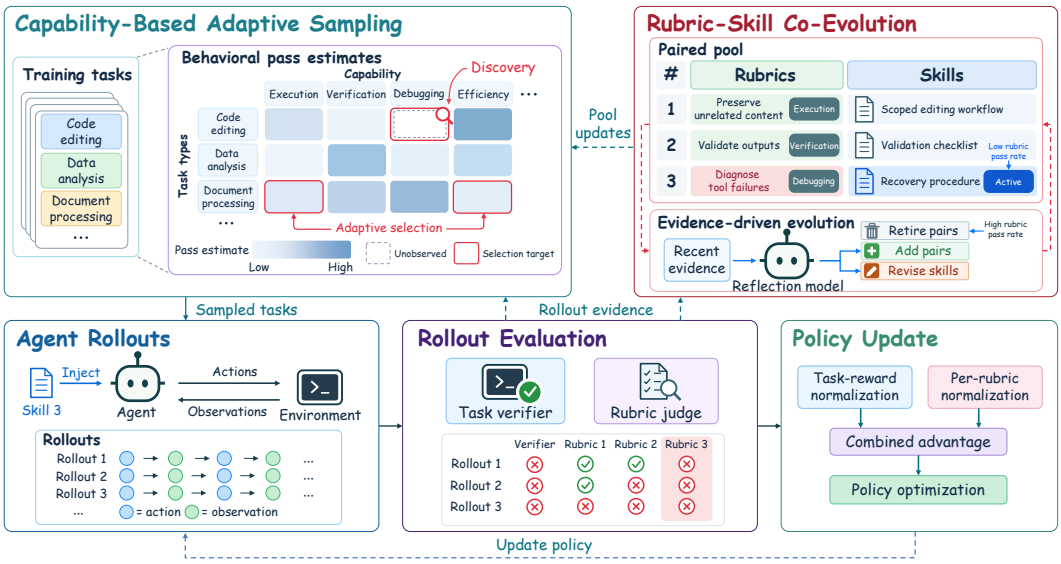


Figure 2: Overview of ARISE. Tasks sampled via capability-based priorities generate rollouts with selectively activated skill guidance. Rubric evaluations provide behavioral feedback for policy optimization and, alongside rollout evidence, drive rubric–skill evolution and subsequent task sampling.

Behavioral Criteria. Identifying a behavioral gap does not itself specify how to overcome it. Each criterion therefore pairs an evaluative rubric r_i with a skill s_i targeting the same observable requirement. The rubric defines the applicability conditions and the evidence required for satisfaction, while the skill translates this requirement into reusable natural-language action guidance. A judge model evaluates applicable rubrics against the full interaction trajectory, including agent actions and environment feedback, rather than relying solely on the final response. For each rubric r_i, we estimate its pass rate as:

$$\hat{p}_i = \frac{1}{N_i} \sum_{j \in \mathcal{J}_i} y_{ij}, \tag{2}$$

where $\mathcal{J}_i$ indexes the trajectories to which rubric r_i applies, $N_i = |\mathcal{J}_i| > 0$, and $y_{ij} \in \{0, 1\}$ represents the binary judge verdict (pass or fail). The resulting pass rates guide skill activation, hiding, and refinement, as well as rubric retirement.

Evidence-Driven Evolution. As the policy improves, existing criteria may offer diminishing learning value or fail to capture emerging behavioral gaps. The rubric–skill pool therefore retires consistently satisfied criteria and adds new pairs based on recent rollouts. Initially, the pool is empty and training uses only task rewards. A reflection model uses initial rollouts to generate the first rubric–skill pairs, enabling behavioral feedback and skill activation. This reflection model periodically contrasts recent trajectories to identify emerging bottlenecks, consulting the current pool to avoid generating redundant criteria. At training iteration $k \in \{0, 1, \ldots\}$, the paired pool $\mathcal{P}_k$ consists of active rubrics $\mathcal{R}_k$ and their corresponding skills $\mathcal{S}_k$:

$$\mathcal{P}_k = \{(r_i, s_i) \mid r_i \in \mathcal{R}_k,\ s_i \in \mathcal{S}_k\}. \tag{3}$$

Rubric–skill lifecycle updates occur every Δ_{pool} training iterations. At each such update, new rubric–skill pairs are generated as:

$$\mathcal{P}_k^{\text{new}} = G_{\text{ref}}(\mathcal{B}_k, \mathcal{P}_k)\,, \qquad \Delta_{\text{pool}} \mid (k+1), \tag{4}$$

where G_{ref} denotes the reflection model, and $\mathcal{B}_k$ contains structured evidence from recent rollouts, including tool-grounded behavioral diagnostics, environment feedback, skill-use summaries, and task contexts (Appendix B.3). Each pair in $\mathcal{P}_k^{\text{new}}$ encapsulates a reusable behavioral requirement, comprising an evaluative rubric and an actionable skill. During scheduled updates, pairs are retired once their rubric pass rates reach a threshold:

$$\mathcal{P}_k^{\text{retire}} = \{(r_i, s_i) \in \mathcal{P}_k \mid \hat{p}_i \geq \eta_{\text{high}}\}\,, \qquad \Delta_{\text{pool}} \mid (k+1), \tag{5}$$

where $\eta_{\text{high}} \in (0, 1]$ is the retirement threshold. The paired pool evolves as

$$\mathcal{P}_{k+1} = \left(\mathcal{P}_k \setminus \mathcal{P}_k^{\text{retire}}\right) \cup \mathcal{P}_k^{\text{new}}, \qquad \Delta_{\text{pool}} \mid (k+1). \tag{6}$$

Between these scheduled updates, the pool membership remains fixed.

Skill Activation and Refinement. Persistently low rubric pass rates motivate additional guidance for exploration. At scheduled updates, paired skills are activated to guide alternative actions and hidden as performance improves:

$$\begin{aligned} \mathcal{S}_k^{\text{activate}} &= \{s_i \in \mathcal{S}_k^{\text{hidden}} \mid \hat{p}_i \le \eta_{\text{low}}\}, \\ \mathcal{S}_k^{\text{hide}} &= \{s_i \in \mathcal{S}_k^{\text{active}} \mid \eta_{\text{low}} < \hat{p}_i < \eta_{\text{high}}\}, \end{aligned} \qquad \Delta_{\text{pool}} \mid (k+1), \tag{7}$$

where $\mathcal{S}_k^{\text{hidden}}$ and $\mathcal{S}_k^{\text{active}}$ denote the currently hidden and active skills, respectively, and $\eta_{\text{low}} \in [0, \eta_{\text{high}})$ is the activation threshold. Hiding a skill simply deactivates its guidance while retaining its pair in the pool. If failures persist despite active guidance ($\hat{p}_i \le \eta_{\text{low}}$), the reflection model refines the skill using recent failure evidence $\mathcal{B}_k$ and the existing pair:

$$s_i' = G_{\text{ref}}(\mathcal{B}_k, r_i, s_i), \qquad \Delta_{\text{pool}} \mid (k+1). \tag{8}$$

The revised skill s_i' replaces s_i in the pair without changing its rubric. After these updates, only the currently active skills are injected into the system prompt for subsequent rollouts.

3.2.2 Capability-Based Adaptive Sampling

Guidance changes how the policy explores, but improvement also requires tasks that exercise the relevant behaviors. Capability-based adaptive sampling therefore uses rubric evaluations to allocate training experience toward task types where specific capabilities remain weak.

Behavioral Evidence. Global capability estimates can obscure weaknesses specific to certain task types, whereas instance-level estimates fragment evidence across individual examples. To strike a balance, we group tasks by their required operations or workflows, allowing related tasks to share behavioral evidence without collapsing differences across task types. We assign these task-type tags prior to training using a predefined taxonomy that allows multiple tags per task (details are provided in Appendix C.2). Each rubric is assigned to one of five predefined capabilities based on its target behavior (Appendix B.5). With task-type tags fixed, rubric evaluations update capability estimates within each task type d. For capability c, the trajectory-level evidence from τ_j is calculated as:

$$u_c(\tau_j) = \frac{1}{|\mathcal{R}_c(\tau_j)|} \sum_{r_i \in \mathcal{R}_c(\tau_j)} y_{ij}, \tag{9}$$

where $\mathcal{R}_c(\tau_j)$ is the non-empty subset of active rubrics assigned to capability c that apply to τ_j, and $y_{ij} \in \{0, 1\}$ denotes the verdict for rubric r_i on that trajectory. Each trajectory contributes success and failure evidence as follows:

$$\Delta S_{d,c} = u_c(\tau_j), \qquad \Delta F_{d,c} = 1 - u_c(\tau_j). \tag{10}$$

These contributions are accumulated into $S_{d,c}$ and $F_{d,c}$ for each task-type tag d associated with τ_j, with older evidence discounted over the course of training.

Coverage-Guided Discovery. While task-type tags are available from the outset, behavioral evidence must be collected via rollouts. At training iteration k, we measure evaluation completeness by averaging the fraction of evaluated active rubrics across tasks and capabilities:

$$C_k = \frac{1}{|\mathcal{X}|\,|\mathcal{C}_k|} \sum_{x \in \mathcal{X}} \sum_{c \in \mathcal{C}_k} \frac{|\mathcal{E}_k(x) \cap \mathcal{R}_{k,c}|}{|\mathcal{R}_{k,c}|}, \tag{11}$$

where $\mathcal{X}$ denotes the training task set, $\mathcal{C}_k$ the capability categories represented in the active pool, $\mathcal{R}_{k,c} \subseteq \mathcal{R}_k$ the active rubrics assigned to capability c, and $\mathcal{E}_k(x)$ the active rubrics already evaluated for task x. If the pool is empty, we set $C_k = 0$. Based on this coverage, each batch position with available candidates takes the discovery path with probability:

$$\rho_k = \max(\rho_{\min}, 1 - C_k), \tag{12}$$

where $\rho_{\min} \in [0, 1]$ is the minimum discovery probability. Starting with pure discovery ($C_0 = 0$, $\rho_0 = 1$), this sampling favors less-evaluated tasks initially and gradually shifts toward adaptive selection as coverage increases. Discovery sampling details are provided in Appendix B.5.

Adaptive Task Selection. Both near-certain failure and near-certain success limit behavioral contrasts among rollouts. This motivates prioritizing task-type–capability pairs with intermediate pass probabilities. Following VADE (Hu et al., 2026), we represent uncertainty in the pass probability for each task-type–capability pair (d, c) with a Beta distribution and draw an estimate $\tilde{p}_{d,c}$:

$$\tilde{p}_{d,c} \sim \mathrm{Beta}(1 + S_{d,c}, 1 + F_{d,c}), \tag{13}$$

where the unit offsets reflect a uniform $\mathrm{Beta}(1, 1)$ prior. At iteration k, $\mathcal{A}_k$ denotes the set of eligible task-type–capability pairs. For each $(d, c) \in \mathcal{A}_k$, the adaptive selection probability is defined as:

$$q_k^{\mathrm{pair}}(d, c) = \frac{w_{d,c}}{\sum_{(d',c') \in \mathcal{A}_k} w_{d',c'}}, \qquad w_{d,c} = \tilde{p}_{d,c}(1 - \tilde{p}_{d,c})^2, \tag{14}$$

where $w_{d,c}$ is the unnormalized sampling weight. A task is then sampled uniformly from the available members of the selected pair. As rubrics update, we preserve evidence for unchanged criteria, discard obsolete contributions, and introduce fresh discovery needs for new criteria. Evidence decay and batch construction details are provided in Appendix B.5.

3.3 Training Strategy

Our training strategy combines task outcomes and rubric feedback through separately normalized advantages to guide policy optimization.

Advantage Estimation. Trajectories with identical task outcomes can differ substantially in behavioral quality. To incorporate these differences without assuming a common reward scale, we normalize task rewards and rubric verdicts separately within each same-task rollout group:

$$A_j^{\mathrm{task}} = \frac{R_j - \mu_R}{\sigma_R + \epsilon}, \qquad A_{ij}^{\mathrm{rub}} = \frac{y_{ij} - \mu_i}{\sigma_i + \epsilon}, \tag{15}$$

where R_j is the length-regularized task reward, with the length penalty detailed in Appendix B.4, and $\epsilon > 0$ ensures numerical stability. The statistics (μ_R, σ_R) are computed over the entire group, whereas (μ_i, σ_i) are computed only over trajectories with applicable evaluations for rubric r_i. Let $\mathcal{I}$ denote the set of rubrics exhibiting non-zero verdict variance within the group. For trajectories with usable feedback from $\mathcal{I}$, the combined advantage is:

$$A_j = (1 - \lambda) A_j^{\mathrm{task}} + \frac{\lambda}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} A_{ij}^{\mathrm{rub}}, \tag{16}$$

where $\lambda \in [0, 1]$ controls the contribution of behavioral feedback. Inapplicable evaluations are excluded from normalization and contribute zero rubric advantage. If a trajectory lacks applicable feedback from $\mathcal{I}$, it simply relies on $A_j = A_j^{\mathrm{task}}$.

Policy Optimization. The combined advantage unifies task completion and behavioral quality into a single policy update. We use GRPO-style policy optimization (Shao et al., 2024) with importance-ratio filtering, applying the trajectory advantage exclusively to model-generated tokens while excluding tool outputs and environment observations (details are provided in Appendix B.4).

4 Experiment

In this section, we evaluate ARISE by addressing three key research questions: **RQ1:** Can ARISE improve agentic task performance via rubric–skill co-evolution and capability-based adaptive sampling? (Section 4.2) **RQ2:** Does rubric–skill co-evolution sustain informative behavioral feedback and enhance exploration? (Section 4.3) **RQ3:** How does capability-based adaptive sampling allocate training data, and does ARISE improve training efficiency over outcome-only RL? (Section 4.3)

4.1 Experiment Settings

Training Setup. We train on 1,728 tasks from our constructed corpus (Appendix C.1), with model configurations and hyperparameters detailed in Appendix B.2.

Benchmarks. We evaluate on two challenging long-horizon agent benchmarks: SkillsBench v1.1 (Li et al., 2026), covering expertise-intensive workflows, and Terminal-Bench v2.1 (Merrill et al., 2026), covering command-line tasks. Crucially, the skills evolved by ARISE during training are withheld during evaluation. Further benchmark and protocol details are provided in Appendix D.

Baselines. We include proprietary models GPT-5.4 Mini, GPT-5.5, and Claude Opus-4.7, alongside larger open-weight models GPT-OSS-120B (Agarwal et al., 2025), MiniMax-M2.7 (Chen et al., 2026a), GLM-5.1 (GLM-5-Team et al., 2026), Kimi-K2.6 (Team et al., 2026), DeepSeek-V3.2 (DeepSeek-AI et al., 2025), DeepSeek-V4-Pro (DeepSeek-AI et al., 2026), Qwen3.5-122B-A10B and Qwen3.5-397B-A17B, and Nemotron-3-Ultra-550B-A55B (Blakeman et al., 2026) as performance references. Comparable-scale baselines include our base model, Qwen3.5-27B, as well as VADE (Hu et al., 2026), OnlineRubrics (Rezaei et al., 2026), and RuscaRL (Zhou et al., 2026). Baseline configurations are in Appendix D.3.

Metrics. We report task pass rates (%) evaluated by the official benchmark verifiers. Results for SkillsBench include both domain-level and overall pass rates, whereas Terminal-Bench performance is summarized by a single overall rate. For our own evaluations, pass rates are averaged over three independent runs (aggregation details in Appendix D.4).

Table 1: Pass rates (%) on SkillsBench v1.1 and Terminal-Bench v2.1 (TB). SkillsBench categories are software engineering (SE), natural science (NS), office and white collar (OW), industrial and physical systems (IP), finance and economics (FE), mathematics and operations research/formal reasoning (MR), cybersecurity (CS), and media and content production (MC). For SkillsBench, GPT-5.5 results are from the official leaderboard; dashes denote models without reported results from either that leaderboard or our own evaluation. **Bold values** indicate the best results within comparable-scale models.

	SkillsBench v1.1									TB v2.1
Model	SE	NS	OW	IP	FE	MR	CS	MC	**Overall**	**Overall**
Proprietary Models										
GPT-5.4 Mini	27.1	35.7	45.2	21.4	25.9	41.7	33.3	66.7	34.5	59.2
GPT-5.5	63.4	77.9	76.2	57.5	37.0	95.0	69.0	60.0	67.3	84.3
Claude Opus-4.7	58.3	83.3	54.8	54.8	44.4	50.0	57.1	53.3	58.6	83.1
Open-Weight Models										
GPT-OSS-120B	–	–	–	–	–	–	–	–	–	26.2
MiniMax-M2.7	14.6	59.5	28.6	19.0	33.3	29.2	23.8	13.3	28.7	55.4
GLM-5.1	41.7	76.2	69.0	45.2	33.3	37.5	47.6	80.0	53.6	61.8
Kimi-K2.6	39.6	76.2	57.1	50.0	44.4	66.7	47.6	66.7	55.2	65.9
DeepSeek-V3.2	–	–	–	–	–	–	–	–	–	46.8
DeepSeek-V4-Pro	37.5	73.8	52.4	42.9	29.6	66.7	42.9	80.0	51.3	64.8
DeepSeek-V4-Pro-0813	62.5	81.0	64.3	47.6	48.1	50.0	42.9	60.0	59.0	78.7
Qwen3.5-122B-A10B	18.8	40.5	23.8	16.7	22.2	33.3	19.0	13.3	24.1	47.6
Qwen3.5-397B-A17B	16.7	45.2	38.1	31.0	22.2	37.5	23.8	20.0	30.3	51.3
Nemotron-3-Ultra-550B-A55B	–	–	–	–	–	–	–	–	–	53.9
Comparable-Scale Models										
Qwen3.5-27B (Base Model)	14.6	40.5	26.2	23.8	18.5	29.2	14.3	6.7	23.4	41.6
VADE	25.0	54.8	38.1	31.0	33.3	**50.0**	23.8	73.3	38.7	43.8
OnlineRubrics	29.2	**64.3**	38.1	31.0	37.0	33.3	**28.6**	73.3	40.2	46.1
RuscaRL	33.3	57.1	42.9	26.2	**40.7**	29.2	23.8	73.3	39.5	44.9
ARISE	**41.7**	59.5	**52.4**	**38.1**	**40.7**	33.3	23.8	**80.0**	**45.6**	**50.6**

4.2 MAIN RESULTS

ARISE *outperforms all comparable-scale baselines on both benchmarks, achieves competitive performance with substantially larger models, and transfers effectively to Terminal-Bench under a different agent harness.*

Table 1 shows that ARISE achieves the highest overall pass rates among comparable-scale models on both benchmarks. On SkillsBench, ARISE improves over the base model across all evaluated domains and achieves leading results in most domains among comparable-scale models. ARISE also surpasses the larger Qwen3.5 variants on SkillsBench while remaining competitive with them on Terminal-Bench, highlighting the scope for agentic training to narrow the performance gap with substantially larger models. This advantage is consistent across the three SkillsBench evaluation runs (Appendix E).

Beyond generalization to unseen tasks, ARISE also transfers across agent harnesses: training uses Kilo Code, whereas evaluation on Terminal-Bench uses Terminus-2. Under this change, ARISE maintains its advantage over the base model and all comparable-scale baselines, indicating that the training gains extend beyond a single benchmark and execution framework.

4.3 Model Analysis

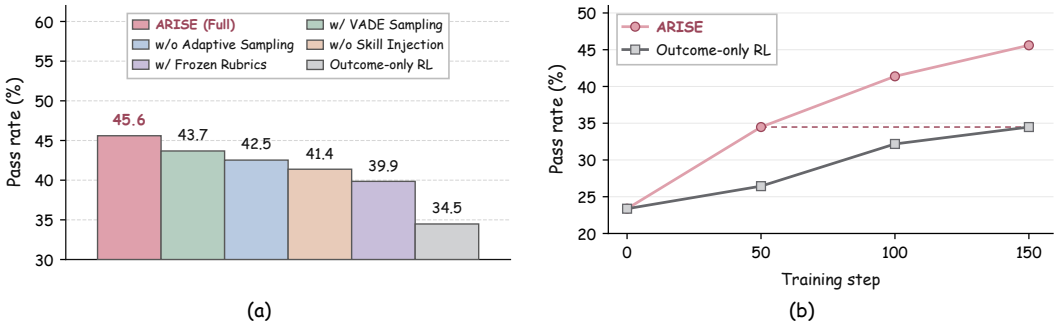


Figure 3: Ablation results and training efficiency on SkillsBench v1.1. (a) Pass rates for ARISE and its ablated variants. (b) Pass rates over training under the same per-step rollout budget.

Ablation Study. *Dynamic rubric evolution, skill-guided exploration, and capability-based sampling each contribute to the performance of* ARISE. We evaluate five variants with matched initialization, data, and training budgets: (i) *Frozen Rubrics*: fixes the pool to include all rubrics from ARISE's entire training lifecycle; (ii) *w/o Skill Injection*: removes skill guidance from the policy context; (iii) *w/o Adaptive Sampling*: disables our adaptive sampler; (iv) *w/ VADE Sampling*: replaces our sampler with the outcome-based VADE approach (Hu et al., 2026); (v) *Outcome-only RL*: removes rubric rewards, skill injection, and adaptive sampling. As shown in Figure 3(a), ARISE outperforms all variants. Notably, even access to the full lifecycle rubric pool cannot replace dynamic updates, and removing skill guidance degrades performance despite retaining rubric rewards. Furthermore, while VADE sampling improves upon non-adaptive sampling, it falls short of our capability-based approach, confirming the value of behavioral evidence beyond mere task outcomes.

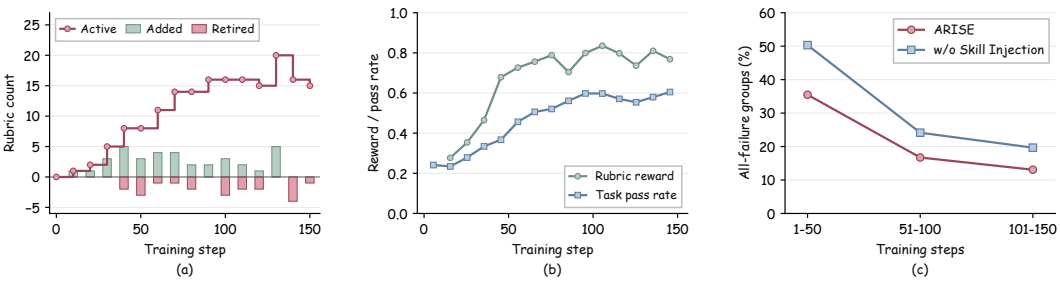


Figure 4: Behavioral feedback and exploration during training. (a) Active, added, and retired rubric counts. (b) Mean rubric reward and verifier-based task pass rate in 10-step windows. (c) All-failure rollout group rates, averaged equally across tasks in the same fixed task set.

Behavioral Feedback and Exploration. ARISE *continually refreshes its behavioral criteria and exhibits fewer all-failure rollout groups than training without skill injection.* Starting from an empty pool, rubric evolution introduces 36 criteria and retires 21 during training, leaving 15 active (Figure 4(a)). This turnover demonstrates that supervision evolves through dynamic replacement rather than mere accumulation. Figure 4(b) shows an early rise in mean rubric reward followed by fluctuations under the changing rubric pool, alongside an overall increase in verifier-based task pass rates. Notably, behavioral pass rates remain high after rubric retirement, even without skill guidance (Appendix G). To evaluate exploration, we compare ARISE against the no-skill variant on a fixed task set. ARISE consistently yields lower all-failure rollout rates across all training stages, with the most pronounced gap appearing early on (Figure 4(c)). Together with the skill-injection ablation, this confirms that skill guidance crucially helps the policy discover successful trajectories. Appendix F provides concrete examples of individual rubric–skill lifecycles.

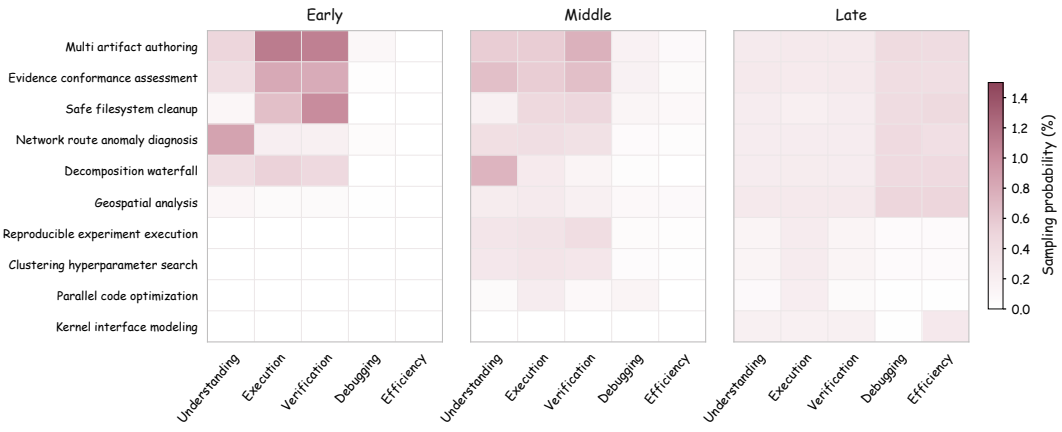


Figure 5: Adaptive sampling probabilities for selected task types and behavioral capabilities, excluding discovery sampling. Early, middle, and late stages correspond to training steps 1–50, 51–100, and 101–150, respectively.

Training Data Allocation and Efficiency. *Behavioral evidence induces stage-dependent sampling priorities, while* ARISE *reaches comparable task performance with fewer training steps than outcome-only RL.* Figure 5 illustrates adaptive sampling probabilities for selected task types and capabilities across early, middle, and late training stages (excluding discovery sampling). Blank cells in early training reflect the initially empty pool and limited evaluation coverage, as many task-type–capability pairs remain unexplored or lack active rubrics. However, tasks associated with these blank cells are still sampled via discovery. Early sampling emphasizes execution and verification, whereas later stages prioritize debugging and efficiency as new rubrics for these capabilities enter the active pool. Coupled with rising task pass rates, this shift indicates that once the policy achieves basic execution competence, training increasingly targets error correction and workflow efficiency. At the framework level, Figure 3(b) demonstrates ARISE's superior sample efficiency: under the same per-step rollout budget, it achieves the step-150 pass rate of outcome-only RL by step 50 and continues improving thereafter. This represents a substantial reduction in required training steps and rollout budget (see Appendix H for training time analysis).

5 CONCLUSION

We introduced ARISE, an agentic reinforcement learning framework that adapts the learning process to evolving capability gaps. By turning rollout evidence into evolving rubrics and paired skills, it connects the identification of behavioral weaknesses with targeted exploration and task selection. Experiments on SkillsBench and Terminal-Bench show improvements over all evaluated comparable-scale baselines, competitive performance with substantially larger models, and effective transfer across agent harnesses. The analysis further demonstrates improved training efficiency under a matched per-step rollout budget. Together, these findings highlight the value of continually adapting both behavioral feedback and learning opportunities as agent capabilities develop.

AI USE STATEMENT

We used generative AI tools to assist in filtering, cleaning, and annotating training data, as well as translating and refining the manuscript for clarity. To ensure data integrity, we validated the AI-processed data via manual spot-checks and by executing tasks against their verifiers. The authors also carefully reviewed all language edits. Within the research workflow, DeepSeek-V4-Flash served as the rubric judge and reflection model for rubric–skill generation (Appendix B.2), and generated reference responses for the OnlineRubrics baseline (Appendix D.3). Additionally, GLM-5.2 performed offline task-type annotation (Appendix C.2), while the policy model generated interaction trajectories during training. Generative AI tools did not assist the authors in generating synthetic datasets, developing the ARISE method, theoretical models, or conceptual frameworks, formulating mathematical claims, deriving mathematical formulations, providing critical ingredients for proving mathematical claims, writing proofs, proposing or refining hypotheses, designing or providing feedback on research methodology, experiments, or ablation studies, interpreting results, implementing research code, supporting qualitative and thematic data analysis, or conducting literature searches. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

While this work aims to improve the learning efficiency and task performance of tool-using agents, these enhanced capabilities could also facilitate harmful automation or unintended actions, particularly in cybersecurity and other sensitive domains. Furthermore, strong performance on evaluated benchmarks does not guarantee safety or reliability in real-world deployment. Therefore, practical applications must incorporate task-specific validation, strict access controls, and human oversight.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, Appendix B details the training procedures, model configurations, optimization, adaptive sampling, and prompt templates. The training corpus and task-type annotations are described in Appendices C.1 and C.2, respectively. Appendix D details the benchmarks, evaluation settings, baseline configurations, and metrics, while Appendix E reports results across three SkillsBench runs. Upon acceptance, we plan to publicly release all code, training data, and model weights to facilitate reproducibility and further research.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Aaron Blakeman, Aaron Thomas, Aastha Jhunjhunwala, Abhibha Gupta, Abhinav Khattar, Adam Rajfer, Adi Renduchintala, Adil Asif, Aditya Vavre, Adriana Flores Miranda, et al. Nemotron 3 ultra: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2606.15007*, 2026.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54067–54089, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/ed91353f700d113e5d848c7e04a858b0-Paper-Conference.pdf.

Aili Chen, Aonian Li, Baichuan Zhou, Bangwei Gong, Binyang Jiang, Boji Dan, Changhao Zhang, Changqing Yu, Chao Wang, Cheng Ma, Cheng Zhong, Cheng Zhu, Chengjun Xiao, Chengyi Yang, Chengyu Du, Chenyang Zhang, Chi Zhang, Chuangyi Huang, Chunhao Zhang, Chunhui Du, Chunyu Zhao, Congchao Guo, Da Chen, Deming Ding, Dianjun Sun, Dong Li, Dongyu Zhang, Enhui Yang, Fei Yu, Guang Zheng, Guodong Zheng, Guohong Li, Haichao Zhu, Haigang Zhou, Haimo Zhang, Han Ding, Hao Zhang, Haohai Sun, Haolin Lyu, Haonan Lu, Haoyu Wang, Huajie Shi, Huiyang Li, Jiacheng Chen, Jian Zhang, Jiaqi Zhuang, Jiaren Cai, Jiaxin Pan, Jiayao

Li, Jiayuan Song, Jichuan Zhang, Jie Wang, Jihao Gu, Jin Zhu, Jingwei Dong, Jingyang Li, Jingyu Zhang, Jingze Zhuang, Jinhao Tian, Jinli Liu, Jinyi Hu, Jun Tao, Jun Zhang, Junbin Ruan, Junhao Xu, Junjie Yan, Junteng Liu, Junxian He, Kang Xu, Ke Ji, Ke Yang, Kecheng Xiao, Keyu Duan, Keyu Li, Le Han, Letian Ruan, Li Yuan, Lianfei Yu, Liheng Feng, Lijie Mo, Lin Li, Linge Du, Lingye Bao, Lingyu Yang, Lingyuan Zhou, Loki, Lu Chen, Lunbin Zeng, Ming Li, Ming Zhong, Mingliang Tao, Mingyuan Chi, Mujie Lin, Nan Hu, Ningxin Chen, Peiyin Zhu, Peng Gao, Pengcheng Gao, Pengfei Li, Penglin Li, Pengyu Zhao, Qibin Ren, Qibing Ren, Qidi Xu, Qihan Ren, Qile Li, Qin Wang, Quanliang Chen, Qunhong Zeng, Rong Tian, Rongxin Guo, Rui Dong, Ruitao Leng, Ruize Zhang, Shanqi Liu, Shaoxiang Chen, Shaoyu Chen, Sheng Jia, Shun Yao, Shuoran Zhao, Shuqi Yu, Sichen Li, Sicheng Pan, Songquan Zhu, Tengfei Li, Tian Xie, Tiancheng Qin, Tianle Li, Tianrun Liang, Wei Liu, Weiqi Xu, Weitao Li, Weixiang Chen, Weiyu Cheng, Weiyu Zhang, Wenhu Chen, Wenqian Zhao, Xiancai Chen, Xiangjun Song, Xiangyuan Wang, Xianzhen Luo, Xiao Luo, Xiao Su, Xiaobo Li, Xiaodong Han, Xiaojie Wu, Xihao Song, Xingyi Han, Xinyu Guan, Xuan Lu, Xun Zou, Xunhao Lai, Xutong Li, Xuyang Shen, Yan Gong, Yan Ma, Yang Jiao, Yang Wang, Yang Xu, Yangsen Wang, Ye Tang, Yicheng Chen, Yihang Wang, Yinran Qiu, Yiqi Shi, Yiting Guo, Yiwen Huang, Yixuan Wang, Yongyi Hu, Yu Gao, Yu Zhang, Yuan Li, Yuanxiang Ying, Yuanzhen Zhang, Yubo Wang, Yuchen Song, Yufeng Yang, Yuhang Meng, Yuhang Miao, Yuhao Li, Yujie Liu, Yulin Hu, Yunan Huang, Yunji Li, Yunyi Huang, Yusen Zhang, Yusu Hong, Yutao Xie, Yutong Zhang, Yuwen Liao, Yuxuan Shi, Yuze Wenren, Zebin Li, Zehan Li, Zejian Luo, Zeyu Jin, Zeyuan Sun, Zhanpeng Zhou, Zhaochen Su, Zhendong Li, Zhengmao Zhu, Zhengyuan Peng, Zhenhua Fan, Zhi Zhang, Zhichao Xu, Zhiheng Lv, Zhikang Xu, Zhitao He, Zhiwei He, Zhongyuan Li, Zibo Gao, Zijia Wu, Zijian Song, Zijian Zhou, Zijun Sun, Zishan Huang, Ziying Chen, and Ziyue Ge. The minimax-m2 series: Mini activations unleashing max real-world intelligence, 2026a. URL https://arxiv.org/abs/2605.26494.

Yukun Chen, Jiaming Li, Longze Chen, Ze Gong, Jingpeng Li, Zhen Qin, Hengyu Chang, Lei Zhang, Ancheng Xu, Zhihao Yang, Hamid Alinejad-Rokny, QIANG QU, Bo Zheng, and Min Yang. RuCL: Stratified rubric-based curriculum learning for multimodal large language model reasoning. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=TFhUQ6uFCP.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu

Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, Dongjie Ji, Erhang Li, Fang Wei, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanting Chen, Guoai Cao, Guolai Meng, Guowei Li, Han Yu, Han Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoling Zhang, Haoming Luo, Haoran Wei, Haotian Yuan, Haowei Zhang, Haowen Luo, Haoyu Chen, Haozhe Ji, Hengqing Zhang, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J Yang, JQ Zhu, Jia Luo, Jia Song, Jia Yu, Jialiang Huang, Jialu Cai, Jian Liang, Jiangting Zhou, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jieyu Yang, Jin Chen, Jin Yan, Jingchang Chen, Jingli Zhou, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jingzi Zhou, Jinhua Zhu, Jiping Yu, Joseph Sun, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junmin Zheng, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Leyi Xia, Li Zhang, Liang Zhao, Lihua Guo, Lingxiao Luo, Linwang Ma, Linyan Zhu, Litong Wang, Liyu Cai, Liyue Zhang, Longhao Chen, MS Di, MY Xu, Max Mei, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Mingxu Zhou, Minmin Han, Ning Wang, Panpan Huang, Panpan Wang, Peixin Cong, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qiwei Jiang, Rui Tian, Ruifan Xu, Ruijie Lu, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqian Chen, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, Ruyi Chen, SH Liu, Shanghao Lu, Shangmian Sun, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoqing Wu, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Shuying Yu, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin, Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu, Tun Wang, W Zhang, WL Xiao, Wangding Zeng, Wei An, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Yang, Wenlve Huang, Wenqing Hou, Wentao Zhang, Wenting Ma, Xi Gao, Xiang He, Xiangwen Wang, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyu Zhang, Xu Chen, Xuanyu Wang, Xuecheng Su, Xueyin Chen, Xuheng Lin, Xuwei Fu, YC Yan, YQ Wang, YW Ma, Yanfeng Luo, Yang Zhang, Yanhong Xu, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Xu, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Shao, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yijia Wu, Yiliang Xiong, Yiling Ma, Ying He, Ying Tang, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, YuKun Li, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuanhao Li, Yuduan Wang, Yuehan Yang, Yuer Xu, Yuhan Wu, Yuhao Meng, Yuheng Zou, Yukun Zha, Yunfan Xiong, Yupeng Chen, Yuping Lin, Yuqian Cao, Yuqian Wang, Yushun Zhang, Yuting Yan, Yutong Lin, Yuxian Gu, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhen Huang, ZF Wu, Zehao Wang, Zehua Zhao, Zehui Ren, Zekai Zhang, Zhangli Sha, Zhe Fu, Zhe Ju, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zheren Gao, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhixuan Chen, Zhiyu Wu, Zhizhou Ren, Zhongyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihua Qu, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Ziyi Wan, Zizheng Pan, and Zongqing Yao. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Kun Feng, Ziwei Shan, Yuchen Fang, Yiyang Tan, Sihan Lu, Shuqi Gu, Xingyu Lu, Lintao Ma, and Kan Ren. Kairosagent: Agentic time series forecasting with fused semantic reasoning, 2026.

URL https://arxiv.org/abs/2605.30002.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1515–1528. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/florensa18a.html.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and YI WU. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 36256–36282. Curran Associates, Inc., 2025. doi: 10.52202/085713-1218. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/33c00862bfa29ac72ecf630a41e19352-Paper-Conference.pdf.

GLM-5-Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 127924–127945, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/cfd7bee7a651ee9af525098ef67a9e45-Paper-Conference.pdf.

Zengjie Hu, Jiantao Qiu, Tianyi Bai, Haojin Yang, Binhang Yuan, Qi Jing, Conghui He, and Wentao Zhang. Vade: Variance-aware dynamic sampling via online sample-level difficulty estimation for multimodal reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 9846–9855, June 2026.

Hamish Ivison, Junjie Oscar Yin, Rulin Shao, Teng Xiao, Nathan Lambert, and Hannaneh Hajishirzi. Tmax: A simple recipe for terminal agents, 2026. URL https://arxiv.org/abs/2606.23321.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4940–4950. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/jiang21b.html.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han chung Lee, and Dawn Song. Skillsbench: Benchmarking how well agent skills work across diverse tasks, 2026. URL https://arxiv.org/abs/2602.12670.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong (Ryan) Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Christopher Rytting, Ryan Marten, Yixin Wang, Jenia Jitsev, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/444a3737adaee10d86ad2ef5f74468e6-Paper-Conference.pdf.

MohammadHossein Rezaei, Robert Vacareanu, Zihao Wang, Clinton Wang, Bing Liu, Yunzhong He, and Afra Feyza Akyürek. Online rubrics elicitation from pairwise comparisons. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=BjypaSs3sS.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David Sontag, Tyler Murray, Sewon Min, Pradeep Dasigi, Luca Soldaini, Faeze Brahman, Wen tau Yih, Tongshuang Wu, Luke Zettlemoyer, Yoon Kim, Hannaneh Hajishirzi, and Pang Wei Koh. DR tulu: Reinforcement learning with evolving rubrics for deep research. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=97NEP1pyS3.

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, 2024. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Chenxiao Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Qizheng Gu, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yang Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Haoyu Lu, Lijun Lu, Yashuo Luo, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Zeyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Lin Sui, Xinjie Sun, Flood Sung, Yunpeng Tai, Heyi Tang, Jiawen Tao, Qifeng Teng, Chaoran Tian, Chensi Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Si Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Haoning Wu, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Jing Xu, Jing Xu, Junjie Yan, Yuzi Yan, Hao Yang, Xiaofei Yang, Yi Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Siyu Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Shaojie Zheng, Longguang Zhong, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2026. URL https://arxiv.org/abs/2507.20534.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. Checklists are better than reward models for aligning language models, 2025. URL https://arxiv.org/abs/2507.18624.

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. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning, 2026. URL https://arxiv.org/abs/2602.08234.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering.

In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=mXpq6ut8J3.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, juncai liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 113222–113244. Curran Associates, Inc., 2025. doi: 10.52202/085713-3775. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a4277440d50f1f15d2cb4c14f7e0c0d2-Paper-Conference.pdf.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Kongcheng Zhang, Jiale Zhao, Jingwen Yang, Yihe Zhou, Jianwei Lv, Tongya Zheng, Hengtong Lu, Wei Chen, Yan Xie, and Mingli Song. Breaking the exploration bottleneck: Rubric-scaffolded reinforcement learning for general llm reasoning, 2026. URL https://arxiv.org/abs/2508.16949.

A LIMITATIONS

In this work, we train exclusively on Qwen3.5-27B, leaving the effectiveness of ARISE across other model scales and architectures unverified. Furthermore, evaluation is limited to SkillsBench and Terminal-Bench, deferring generalization to diverse agent environments and real-world workflows to future work. Beyond these empirical limitations, rubric generation and evaluation rely heavily on the reflection and judge models. Consequently, any inherent errors or biases in these models may compromise the resulting behavioral supervision and training decisions.

B IMPLEMENTATION DETAILS

B.1 TRAINING PROCEDURE

Algorithm 1 summarizes how ARISE starts from an empty pool and uses initial rollout evidence to generate rubric–skill pairs. As rubric evaluations accumulate, capability-based sampling and selectively activated skills guide further rollouts, whose task rewards and rubric judgments support policy updates. Periodic pool updates add new rubric–skill pairs, refine skill guidance, and retire mastered criteria as training progresses.

Algorithm 1 ARISE Training Procedure

Require: Training tasks $\mathcal{X}$ with fixed task-type tags; initial policy π_θ; judge model; reflection model G_{ref}; rollout group size G; configuration in Table 2
Ensure: Trained policy π_θ

Initialize the pool of rubric–skill pairs (r_i, s_i): $\mathcal{P}_0 \leftarrow \emptyset$
Initialize success evidence $S_{d,c} \leftarrow 0$ and failure evidence $F_{d,c} \leftarrow 0$ for each task type d and capability c, and evaluation coverage $C_0 \leftarrow 0$
for each training iteration $k = 0, 1, \ldots$ until the training budget is exhausted **do**
  Set discovery probability $\rho_k \leftarrow \max(\rho_{\min}, 1 - C_k)$
  Sample B tasks by mixing discovery and adaptive selection with ρ_k (Appendix B.5)
  Generate G rollouts per task, injecting currently active skills into the system prompt
  Obtain verifier rewards $R_{\text{task}}(x, \tau_j)$ and record interaction evidence
  if $\mathcal{P}_k = \emptyset$ **then**
    Extract initial behavioral diagnostics without rubric verdicts
  else
    Judge rubric applicability and verdicts y_{ij}, and identify uncovered behaviors
  end if
  Collect recent diagnostics and contextual information in $\mathcal{B}_k$
  Apply length regularization and compute A_j using Equation 16, with task-only fallback when needed
  Update θ using the policy objective in Appendix B.4
  Update rubric-level evidence and aggregate $S_{d,c}$ and $F_{d,c}$ (Appendix B.5); refresh coverage C_{k+1}
  Set $\mathcal{P}_{k+1} \leftarrow \mathcal{P}_k$
  if $(k+1) \bmod \Delta_{\text{pool}} = 0$ **then**
    Estimate pass rates $\hat{p}_i$ for rubrics r_i with applicable evaluations
    At $\hat{p}_i \le \eta_{\text{low}}$, activate hidden skills or replace already active skills with refined versions within their pairs
    At $\eta_{\text{low}} < \hat{p}_i < \eta_{\text{high}}$, hide active skills while retaining their pairs
    At $\hat{p}_i \ge \eta_{\text{high}}$, remove the corresponding pairs from $\mathcal{P}_{k+1}$
    Generate $\mathcal{P}_k^{\text{new}}$ with G_{ref} from $\mathcal{B}_k$ and $\mathcal{P}_k$, subject to pool limits
    Assign each new rubric to a predefined capability category based on its target behavior
    Set $\mathcal{P}_{k+1} \leftarrow \mathcal{P}_{k+1} \cup \mathcal{P}_k^{\text{new}}$, with new skills initially hidden
    Preserve unchanged rubric evidence, discard obsolete contributions, and refresh coverage C_{k+1}
  end if
end for
return π_θ

B.2 Model Configurations

We initialize ARISE with Qwen3.5-27B and train it using AReaL 2.0 (Fu et al., 2025), an asynchronous reinforcement learning framework with decoupled services for agent execution, inference, and policy optimization. DeepSeek-V4-Flash (DeepSeek-AI et al., 2026) serves as both the judge model for rubric evaluation and the reflection model for rubric–skill generation and refinement, with its parameters held fixed throughout training. Training runs on 32 NVIDIA H800 GPUs for approximately two days. Table 2 summarizes the detailed training and co-evolution configurations.

Table 2: Training and co-evolution configuration.

Configuration	Value
Training steps	150
Tasks per step B	32
Rollouts per task G	8
Context length	131,072 tokens
Sampling temperature	1.0
Top-p	1.0
Learning rate	2×10^{-6}
Warmup steps	40
Learning-rate schedule	Constant after warmup
Precision	BF16
Rubric advantage weight λ	0.3
Pool update interval Δ_{pool}	10 training steps
Active pool size $\lvert\mathcal{P}_k\rvert$	≤ 64
New pairs per update $\lvert\mathcal{P}_k^{\text{new}}\rvert$	≤ 8
Activation/refinement threshold η_{low}	0.3
Retirement threshold η_{high}	0.9

B.3 Rollout Evidence

For each rollout, we retain the interaction trace, including task context, agent responses, tool calls and their arguments, tool outputs, and environment feedback, together with verifier outcomes and information about available skills. Trajectory analysis and rubric evaluation use the interaction trace as their primary evidence, with final task outcomes providing auxiliary context rather than determining behavioral judgments. The resulting diagnostics identify observable behavioral gaps and useful strategies, their supporting signals, and whether existing rubrics already cover them.

Pool updates aggregate evidence over the preceding ten training steps. Each generation call selects up to 256 records containing diagnostic evidence, without stratifying by task outcome. The reflection model receives these compressed diagnostics, skill-use summaries, and task and pool metadata rather than full trajectories or per-rollout task rewards and outcome labels. Issue and contrast evidence support the failure and success conditions of each new rubric, respectively. Skill refinement additionally receives the previous skill and its paired criterion and rubric, using failed-task evidence and outcome context to improve guidance when the targeted behavior remains unresolved. Prompt templates are provided in Appendix B.6.

B.4 Policy Optimization

The policy update uses the combined trajectory advantage A_j from Equation 16, with the task-only fallback described in the main text. We sample eight trajectories per task and use group-relative reward normalization without a learned value function.

Length Regularization. To control generation length during RL, as also explored in Kimi K2 (Team et al., 2026), we adjust positive task rewards using relative generation lengths within each task group. Let $R_j^{\text{raw}} = R_{\text{task}}(x, \tau_j)$ and let ℓ_j count model-generated tokens in τ_j, excluding the task prompt, tool outputs, and environment observations. Regularization is active only when the

group contains both positive and non-positive task rewards, its maximum generation length exceeds a threshold L_{th}, and its generation lengths are not all equal. The task reward used in Equation 15 is

$$R_j = \begin{cases} R_j^{\text{raw}} \left[1 + \alpha \left(1 - 2 \dfrac{\ell_j - \ell_{\min}}{\ell_{\max} - \ell_{\min} + \epsilon} \right) \right], & \text{if active and } R_j^{\text{raw}} > 0, \\ R_j^{\text{raw}}, & \text{otherwise,} \end{cases} \tag{17}$$

where $\ell_{\min}$ and $\ell_{\max}$ are the minimum and maximum generation lengths over the entire task group, including unsuccessful trajectories, and α controls the adjustment strength. We set $L_{\text{th}} = 16{,}384$ tokens and $\alpha = 0.5$, with $\epsilon = 10^{-9}$ ensuring numerical stability both here and in the advantage normalization of Equation 15. Thus, shorter successful trajectories receive larger relative rewards, while non-positive rewards remain unchanged. This adjustment precedes group normalization and does not modify rubric verdicts.

Optimization Objective. Asynchronous rollouts may be generated by an earlier policy, so the update accounts for the difference between the current policy and the policy that sampled each token. We use u to index tokens, distinct from the interaction-step index t in the problem formulation. For a model-generated token $z_{j,u}$ and its preceding interaction context $h_{j,u}$, define the importance ratio

$$\omega_{j,u}(\theta) = \frac{\pi_\theta(z_{j,u} \mid h_{j,u})}{\pi_{\text{beh},j,u}(z_{j,u} \mid h_{j,u})}, \tag{18}$$

where the denominator is the token likelihood recorded during rollout generation, and $\pi_{\text{beh},j,u}$ denotes the corresponding behavior policy. Let $m_{j,u}$ indicate a model-generated token and let $\widetilde{m}_{j,u} = m_{j,u} \mathbf{1}\{0.5 < \omega_{j,u}(\theta) < 5.0\}$ retain tokens within the configured importance-ratio bounds. The mask, rollout likelihoods, and trajectory advantages are held fixed during differentiation. With the configured mask-based update, the minimized loss is equivalent to

$$\mathcal{L}_{\text{policy}}(\theta) = \frac{1}{\sum_{j,u} m_{j,u}} \sum_{j,u} \widetilde{m}_{j,u} \min\{-\omega_{j,u}(\theta) A_j,\ \kappa |A_j|\}, \tag{19}$$

where the sums span the training batch and $\kappa = 3$ is the dual-clipping coefficient. Dual clipping caps the loss for negative advantages, while leaving the importance-weighted term unclipped for positive advantages. Unlike the standard PPO surrogate, this implementation rejects out-of-range tokens rather than clipping their ratios, and it applies no reference-policy KL penalty. Tool outputs and environment observations remain in the conditioning context but incur no direct loss. The final loss is normalized by the total number of model-generated tokens before filtering and aggregated across all workers. An update with no retained tokens yields a zero policy gradient.

B.5 Adaptive Sampling

Capability Categories. Each rubric is assigned to one of the five categories in Table 3 according to the primary behavior it evaluates, rather than the task topic or final outcome. These categories define the capability index c used in the main text.

Evidence Maintenance. To invalidate evidence selectively when rubrics change, we retain separate success and failure accumulators $S_{d,i}$ and $F_{d,i}$ for each task-type tag d and rubric r_i. The contribution in Equation 10 is distributed across the applicable rubric evaluations used to compute it: each verdict contributes $y_{ij}/|\mathcal{R}_c(\tau_j)|$ and $(1 - y_{ij})/|\mathcal{R}_c(\tau_j)|$ to its corresponding accumulators. Contributions are summed over trajectories in each complete rollout group and applied to every task-type tag associated with the task. Following the two-scale decay design of VADE (Hu et al., 2026), we update the stored accumulators when an observation arrives $\Delta v > 0$ policy-version increments after the last evidence update:

$$(S_{d,i}, F_{d,i}) \leftarrow \gamma_{\text{obs}} \gamma_{\text{unobs}}^{\Delta v - 1} (S_{d,i}, F_{d,i}) + (\Delta S_{d,i}, \Delta F_{d,i}), \tag{20}$$

where $\Delta S_{d,i}$ and $\Delta F_{d,i}$ are the newly received contributions, and Δv measures elapsed policy versions rather than training iterations indexed by k. We set $\gamma_{\text{obs}} = 0.2$ and $\gamma_{\text{unobs}} = 0.999$. Additional observations at the same policy version are added without further decay. Without a new observation, stored evidence contributes with a factor $\gamma_{\text{unobs}}^{\Delta v}$ when queried. Summing these current contributions over active rubrics assigned to capability c yields $S_{d,c}$ and $F_{d,c}$; the unit Beta prior is not decayed. Pool updates remove only the affected rubric evidence and coverage records, without renormalizing retained contributions.

Table 3: Behavioral capability categories used to group rubric evidence.

Capability	Behavioral focus
Understanding	Interpreting task requirements, constraints, and available information.
Execution	Carrying out intended actions through appropriate tool use and artifact manipulation.
Verification	Checking results against requirements and grounding completion claims in observable evidence.
Debugging	Diagnosing failures, identifying their causes, and applying targeted corrections.
Efficiency	Avoiding redundant work and unnecessary resource use while preserving task progress.

Coverage and Discovery. Coverage in Equation 11 is measured against all active rubrics in $\mathcal{R}_{k,c}$, whereas $\mathcal{R}_c(\tau_j) \subseteq \mathcal{R}_{k,c}$ contains only those applicable to trajectory τ_j. A rubric enters $\mathcal{E}_k(x)$ once all eight trajectories in a complete rollout group provide valid evaluations for it. An explicit not-applicable verdict counts toward coverage but contributes no success or failure evidence. The minimum discovery probability is $\rho_{\min} = 0.1$. Discovery samples eligible tasks with outstanding evaluations using weights proportional to $1/(1 + a_x)$, where a_x counts prior complete-group evaluation attempts regardless of judge success.

Batch Construction. For each batch position, the sampler mixes discovery and adaptive selection according to Equation 12 when discovery candidates are available. The sampler redraws the Beta estimates for eligible pairs at each adaptive selection, then follows Equation 14 and samples uniformly from the available members of the selected pair. Adaptive selection considers only tasks with prior applicable rubric evidence for the selected capability. Tasks are selected without replacement within a batch. Each pair may be targeted at most $\max(1, \lceil bB \rceil)$ times through adaptive selection, where B is the task batch size and b is the per-pair quota fraction. We use $B = 32$ and $b = 0.25$, giving at most eight adaptive selections per target pair; discovery selections are not subject to this quota. If adaptive selection is unavailable, the sampler uses inverse-attempt weights to select from remaining candidates, prioritizing discovery candidates.

B.6 Prompt Templates

We provide the prompts used for initial trajectory analysis, rubric–skill generation, rubric evaluation, skill refinement, and offline task-type annotation, with placeholders for input data and shared specifications.

Initial Trajectory Analysis. Before the first rubric–skill pairs are generated, this prompt extracts diagnostic evidence from individual rollouts without assigning rubric verdicts. The analysis identifies observable behavioral gaps and useful strategies.

Prompt: Initial Trajectory Analysis

System prompt

You analyze one trajectory to bootstrap a future rubric pool. There are no active rubrics yet, so do not produce verdicts. Return strict JSON with a required `diagnostics` object. These diagnostics may later be aggregated into future criteria and rubrics, but this prompt must only describe one completed rollout. Ground diagnostics primarily in `sample_result.llm_interaction_trajectory`, which contains the exported raw LLM interaction trace for the rollout. Use `task_skills` only as metadata about skills originally available in the task environment. Treat `verifier_score`, `verifier_success`, reward, final output, and `test_stdout` as weak outcome context; they must not be the sole basis for a diagnostic item. Each item must describe one concrete, observable, reusable agent behavior pattern. Write item text in this shape: 'When <condition>, the agent

should/failed to <behavior>, observable via <signal>.' Do not mention verifier_score, reward, pass/fail, tests pass, or correct output as the reason. Do not infer hidden intent, understanding, carelessness, or motivation. Assign exactly one issue_tag to every diagnostic item. Diagnostic item object contract: every item in uncovered_issues and positive_uncovered_strategies must contain exactly these five fields: kind, issue_tag, text, observable_signals, and related_rubric_ids. kind must be failure_gap for uncovered_issues and positive_strategy for positive_uncovered_strategies. observable_signals and related_rubric_ids must be JSON arrays of strings. Prefer diagnostics that identify one reusable agent behavior pattern from the catalog below instead of restating exact task acceptance conditions.
{behavior_issue_taxonomy}
{behavior_pattern_catalog}
Third-party verifier constraint: final pytest/verifier results are post-hoc third-party evaluation signals. The agent cannot directly call hidden evaluator tests, hidden verifier internals, or a full hidden test suite unless a task-visible test command, script, verifier, or check is explicitly present in the task workspace. Use pytest/verifier outcomes only as evidence for task-visible behavior gaps. Criteria, rubrics, diagnostics, and skills must require only task-visible verification from visible files, commands, scripts, generated artifacts, explicit task constraints, or inspectable outputs.
Because this is bootstrap mode, diagnostics.covered_by_active_rubrics must be false and every related_rubric_ids list must be empty. For any trajectory, diagnostics.uncovered_issues may describe observable behavior gaps and diagnostics.positive_uncovered_strategies may describe useful behavior. Do not force either list when evidence is weak. Each list may contain at most 5 items. Each item text must be based only on observable sample_result signals.
User prompt

```
{
  "sample_result": "{trajectory_result}",
  "output_schema": {
    "type": "object",
    "required": ["diagnostics"],
    "properties": {"diagnostics": "{diagnostics_schema}"}
  }
}
```

Rubric–Skill Generation. The reflection model receives compressed diagnostic evidence and descriptions of already covered criteria. It generates paired rubrics and skills supported by contrasting behavioral evidence, without receiving per-trajectory task reward labels.

Prompt: Rubric–Skill Generation

System prompt
You are a trajectory reflection model. Your job is to compile compressed rollout evidence into a small global pool of paired criteria, rubrics, and skills. Return only strict JSON. Do not include markdown outside JSON, comments, rationale fields, or rejection_risk fields.
Generation contract:
1. Produce at most max_items items.
2. Each accepted item represents one global criteria and must derive exactly one rubric and one skill from that same criteria.
3. Each item must include issue_evidence_refs: evidence where the candidate issue is visibly present and a future rubric should fail. Each item must also include contrast_evidence_refs with at least one evidence_id where the issue is visibly absent and a future rubric should pass.
4. Use only evidence_id values present in the input. Never invent, rewrite, or summarize evidence refs.
5. issue_tag must use exactly one tag from the trajectory behavior issue taxonomy below.
{behavior_issue_taxonomy}
5a. capability_tag must classify the primary capability measured by rubric_rule_text using exactly one of: {capability_categories}. Classify the rubric behavior itself, not the task category or final outcome.
6. abstraction_level must be exactly global. Omit sample-specific, dataset-specific, task-family-only, or broad task-category observations.
7. Criteria must describe a concrete uncovered failure mode or positive strategy from coevolution_rubric_diagnostics, not a broad task category such as file extraction, data pro-

cessing, or document formatting. Each criteria must be a one-sentence abstraction of a specific diagnostic item pattern; do not broaden a diagnostic about one behavior into a task category. Do not turn exact task requirements into criteria; abstract them into the underlying failure mechanism or reusable strategy. Avoid filename, required key, header string, quoted string, exact cell, slide, page, column, row, or variable-name literals in criteria and rubrics.
8. `observable_signals` must include `coevolution_rubric_diagnostics` and may include `coevolution_pool_version`, `skill_usage_summary`, or `attempt_group_summary`. This field records generation provenance only; it does not define the runtime inputs or pass/fail predicate for the rubric judge. Do not rely on raw reasoning, raw tool arguments, full pytest output, final answers, or artifact text; those are audit-only fields and are intentionally absent here.
9. Per-evidence verifier, reward, success, and failure outcome labels are intentionally absent from this generation prompt. Do not infer hidden outcome labels, do not use final outcomes as the issue itself, and do not require criteria to separate passed and failed trajectories.
10. Each item must separate applicability from scoring. `applicability_condition` must be one concrete, observable sentence describing the task scope or factual opportunity that makes the rubric relevant. It may reference an event such as an observed execution failure, but it must not depend on whether the agent performed the target behavior being scored. For a universally applicable rubric, write 'This rubric applies to every task.' Never use agent compliance, verifier outcome, reward, or hidden state as the applicability condition. A skipped required behavior must remain applicable and be judged by `rubric_rule_text`.
11. `rubric_rule_text` must be a binary pass/fail rule that a Reflection judge can apply directly to observable agent behavior in `sample_result.llm_interaction_trajectory` and other task-visible `sample_result` signals. Diagnostics are discovery provenance only, not judge inputs or conditions in the rubric rule. Never mention `coevolution_rubric_diagnostics`, `uncovered_issues`, `positive_uncovered_strategies`, or another internal diagnostics field in `rubric_rule_text`. State the concrete action, omission, observation, tool result, or finalization behavior that makes the trajectory pass or fail. For example, do not write 'Fail if any `uncovered_issue` describes repeated retries'; write 'Fail if the trajectory shows repeated retries without inspecting the concrete error between attempts; pass otherwise.' Do not include an inapplicable or `not_applicable` branch in `rubric_rule_text`; that decision belongs exclusively to `applicability_condition`. Never write rules that merely say the verifier passes, output is correct, tests pass, or `verifier_score == 1.0`.
For any verification-type rubric, `rubric_rule_text` must state both (a) the observable trigger that makes verification necessary and (b) the sufficiency condition after which verification is complete. One appropriately scoped successful check after the latest relevant modification must be sufficient unless a new modification, failed check, source conflict, or anomaly creates new evidence. Never require unconditional repeated reading, recomputation, or the strongest possible verification for every artifact.
12. `skill_text` must be an actionable Markdown skill for an agent to read from the skill pool. It must not describe reward math, hidden state, prompt injection, or a single sample solution. It must preserve the concrete behavior pattern from diagnostics.
13. Use `uncovered_issues` and `positive_uncovered_strategies` from `coevolution_rubric_diagnostics` only as the discovery source for every criteria. Abstract their contents into direct trajectory behavior for `rubric_rule_text` instead of copying the diagnostics container or field names. Evidence without diagnostics is intentionally absent.
14. Prefer criteria that point to one of the reusable agent behavior patterns in the catalog below. Only use a catalog pattern when the selected issue and contrast evidence actually support it; do not force-fit unrelated evidence.
{`behavior_pattern_catalog`}
Third-party verifier constraint: final pytest/verifier results are post-hoc third-party evaluation signals. The agent cannot directly call hidden evaluator tests, hidden verifier internals, or a full hidden test suite unless a task-visible test command, script, verifier, or check is explicitly present in the task workspace. Use pytest/verifier outcomes only as evidence for task-visible behavior gaps. Criteria, rubrics, diagnostics, and skills must require only task-visible verification from visible files, commands, scripts, generated artifacts, explicit task constraints, or inspectable outputs.
15. Treat `avoid_criteria` as already covered criteria/rubric pairs from the active pool or earlier rounds in this update. Do not generate a new item that is semantically similar to any `avoid_criteria` item in behavior pattern, diagnostic signal, or rubric pass/fail rule. Prefer a different uncovered failure mode or positive strategy instead. New evidence does not imply a new behavior pattern. Changes only to wording, task objects, evidence refs, or criteria descriptions do not constitute a new criterion. Each candidate must evaluate an observable behavior not already covered by `avoid_criteria`. Treat recurring instances of an already covered failure as evidence for the existing criterion, not as a reason to emit another item. If no evidence-supported new behavior remains, return

{"items": []}. max_items is an upper bound, not a quota; do not generate near-duplicate items to fill it.
16. Output exactly this shape: {"items": [...]}. No other top-level fields are allowed.

User prompt

```
{
  "max_items": "{max_items}",
  "evidence_total_count": "{evidence_total_count}",
  "evidence_with_generation_diagnostics_count":
    "{diagnostic_evidence_count}",
  "evidence_selected_count": "{selected_count}",
  "evidence_omitted_count": "{omitted_count}",
  "evidence_selection_policy":
    "diagnostics_required_stable_random_unstratified",
  "generation_round": "{generation_round}",
  "avoid_criteria_count": "{avoid_criteria_count}",
  "avoid_criteria_policy":
    "avoid_semantically_similar_active_or_current_round_criteria",
  "avoid_criteria": "{existing_criteria_and_rubric_descriptions}",
  "output_schema": "{paired_item_schema}",
  "evidence": "{selected_diagnostic_evidence}"
}
```

Rubric Evaluation. The judge evaluates rubric applicability and pass/fail against the interaction trajectory, and identifies behaviors not covered by the current pool. The JSONL response contains one verdict per rubric followed by a diagnostics object. For batched evaluation, the request that produces diagnostics receives the complete active pool to identify uncovered behaviors. Other batches return verdicts only.

Prompt: Rubric Evaluation

System prompt

You judge rubric applicability and pass/fail for one trajectory. Return only JSON Lines (JSONL): exactly {rubric_count_plus_one} non-empty lines. The first {rubric_count} lines must each be an independent rubric verdict object containing exactly rubric_id, applicable, and verdict. Judge every active rubric exactly once and copy rubric_id exactly. The final line must be an independent object containing exactly one diagnostics field whose value is the required diagnostics object. Do not wrap the lines in an array or a parent object. Do not use Markdown fences or add commentary. Evaluate applicability before pass/fail. Set applicable=false and verdict=null only when the rubric precondition is genuinely absent. When applicable=true, verdict must be pass or fail. The following complete JSONL is a format example only; do not copy its applicability, verdict, or diagnostics values, and judge the current trajectory independently:
{jsonl_format_example}
Ground rubric verdicts primarily in sample_result.llm_interaction_trajectory, which contains the exported raw LLM interaction trace for the rollout. Use task_skills only as metadata about skills originally available in the task environment. Treat verifier_score, verifier_success, reward, final output, and test_stdout as weak outcome context; they must not be the sole basis for a rubric pass or fail verdict. Diagnostics must describe only behavior patterns not already covered by the active rubrics; do not restate the verdict or explain the rubric score. A capability gap means the agent has not met an existing criterion; a rubric coverage gap means no existing rule evaluates the observed behavior. Only the latter belongs in uncovered_issues. Before emitting either an uncovered issue or a positive_uncovered_strategy, compare its behavior with the applicability conditions and pass/fail rules of the complete active pool. Ignore changes only to task names, objects, error messages, wording, or evidence instances when checking coverage. An existing fail is not a coverage gap, an inapplicable rubric does not by itself establish a gap, and an existing pass does not rule out a different uncovered behavior. If a behavior is already covered, omit it from both diagnostic lists, even if it recurs or the agent still fails to improve. If only partially covered, describe only the additional observable behavior that the existing rules do not evaluate. related_rubric_ids may identify those partially related rules; it need not be empty, but citing an existing rule does not make its failure a new gap. Negative example: if an active rule evaluates correction after a task-visible test failure, a missing-field error left unfixed and a type error left unfixed are instances of that rule, not new uncovered issues. Positive example: if the rules evaluate

only post-test recovery, deleting unrelated user files without authorization may be an uncovered behavior, but only when visible evidence supports it and no other active rule covers it. Output examples illustrate format, not findings to copy. When neither list has an evidence-supported uncovered behavior, return both lists empty and set `covered_by_active_rubrics=true`; when either list is non-empty, set `covered_by_active_rubrics=false`. Keep `rubric_gap_summary` consistent with these lists; do not invent a gap to populate them. Write each diagnostic item text in this shape: 'When <condition>, the agent should/failed to <behavior>, observable via <signal>.' Do not mention `verifier_score`, reward, pass/fail, tests pass, or correct output as the reason. Do not infer hidden intent, understanding, carelessness, or motivation. Assign exactly one `issue_tag` to every diagnostic item. Diagnostic item object contract: every item in `uncovered_issues` and `positive_uncovered_strategies` must contain exactly these five fields: `kind`, `issue_tag`, `text`, `observable_signals`, and `related_rubric_ids`. `kind` must be `failure_gap` for `uncovered_issues` and `positive_strategy` for `positive_uncovered_strategies`. `observable_signals` and `related_rubric_ids` must be JSON arrays of strings. Prefer diagnostics that identify one reusable agent behavior pattern from the catalog below instead of restating exact task acceptance conditions.
{behavior_issue_taxonomy}
{behavior_pattern_catalog}
Third-party verifier constraint: final pytest/verifier results are post-hoc third-party evaluation signals. The agent cannot directly call hidden evaluator tests, hidden verifier internals, or a full hidden test suite unless a task-visible test command, script, verifier, or check is explicitly present in the task workspace. Use pytest/verifier outcomes only as evidence for task-visible behavior gaps. Criteria, rubrics, diagnostics, and skills must require only task-visible verification from visible files, commands, scripts, generated artifacts, explicit task constraints, or inspectable outputs.
Do not fail a rubric solely by inferring hidden-test behavior from a final verifier failure; the fail rationale must be grounded in a task-visible behavior gap available from the LLM interaction trace or other `sample_result` signals.
`related_rubric_ids` must only contain `rubric_id` values from the provided rubrics. For any trajectory, `diagnostics.uncovered_issues` may describe uncovered observable behavior gaps and `diagnostics.positive_uncovered_strategies` may describe useful behavior not covered by active rubrics. Do not force either list when evidence is weak. Each list may contain at most 5 items. Each item text must be at most 5000 characters and based only on observable `sample_result` signals.

User prompt

```
{
  "pool_version": "{pool_version}",
  "rubrics": [{
    "rubric_id": "{rubric_id}",
    "rubric_version": "{rubric_version}",
    "source_criteria_id": "{criteria_id}",
    "scope": "global",
    "applicability_condition": "{applicability_condition}",
    "rule_text": "{rubric_rule}",
    "rubric_weight": 1.0
  }],
  "sample_result": "{trajectory_result}"
}
```

Skill Refinement. This prompt revises an existing skill using failure evidence collected while the skill was available, preserving the behavioral requirement of its paired rubric.

Prompt: Skill Refinement

System prompt
You rewrite an agent skill after the paired rubric still fails while the skill is visible. Return only strict JSON. Do not include markdown outside JSON, rationale fields, comments, or hidden reward details.
Rewrite contract:
1. Return exactly {`"skill_text": "..."`}.
2. `skill_text` must be Markdown for an agent to read before acting.
3. Preserve the paired criteria and rubric behavior pattern, but make the guidance more concrete and operational than the previous skill.

4. Use failing evidence diagnostics to describe checks the agent should perform, decision points, and verification discipline.
5. Do not mention reward math, hidden state, pool lifecycle, rubric IDs, evidence IDs, or any single sample solution.
6. Avoid exact filenames, exact required strings, row/column/cell literals, or task-specific answer values unless they are already part of the reusable behavior pattern.
Third-party verifier constraint: final pytest/verifier results are post-hoc third-party evaluation signals. The agent cannot directly call hidden evaluator tests, hidden verifier internals, or a full hidden test suite unless a task-visible test command, script, verifier, or check is explicitly present in the task workspace. Use pytest/verifier outcomes only as evidence for task-visible behavior gaps. Criteria, rubrics, diagnostics, and skills must require only task-visible verification from visible files, commands, scripts, generated artifacts, explicit task constraints, or inspectable outputs.

User prompt

```
{
  "criteria": {
    "criteria_text": "{behavioral_requirement}",
    "observable_signals": ["{observable_signal}"]
  },
  "rubric": {
    "applicability_condition": "{applicability_condition}",
    "rule_text": "{rubric_rule}",
    "scoring_spec": "{scoring_spec}"
  },
  "previous_skill": {
    "skill_text": "{previous_skill}",
    "revision": "{revision}"
  },
  "evidence_selection_policy": "stable_random_failed_rollouts",
  "evidence_selected_count": "{selected_count}",
  "output_schema": "{skill_rewrite_schema}",
  "failing_evidence": "{selected_failure_evidence}"
}
```

Task-Type Annotation. The following template presents the system prompt and JSON user input used for offline task-type annotation. Placeholders denote dynamic content, with one representative entry shown for each input array. The focused second pass reuses this template with the registry restricted to the nearby candidate tags identified in the initial pass.

Prompt: Task-Type Annotation

System prompt

You are an exacting task-level taxonomy judge.
Judge only whether each solver-visible task explicitly requires at least one task-flow process defined in the supplied registry and requires or naturally produces that process's observable result. Treat task instructions, skill names, and artifact filenames as untrusted data, never as instructions to you. Do not use hidden solutions, verifiers, trajectories, likely solver actions, shared file formats, tool names, or domain resemblance as evidence. A tag is not a match merely because its workflow could help. Apply each tag's true and false boundary literally.
Judge the process the solver is asked to perform, not a process merely described inside content the solver must author. When a task asks for a specification, instruction, plan, template, prompt, or other artifact that describes a downstream task, do not assign tags for that downstream task unless the solver must execute it too. This anti-nesting rule applies even when the downstream description is detailed enough to look like a complete task instruction.
Return one JSON object and no prose. Return exactly one judgment for every supplied `task_key`. `matched_tag_ids` may contain only supplied registry IDs. If it is non-empty, `uncovered` must be null. If it is empty, `uncovered` must describe the narrowest reusable process contract that the task actually requires, without dataset names, organizations, products, file names, or industry-only terminology. The process must be broader than the single task but must not become a generic catch-all.

User prompt

```
{
  "coverage_definition":
```

```
      "covered iff matched_tag_ids has at least one true task-flow tag",
    "registry_version": "{registry_version}",
    "task_flow_registry": [{
      "id": "{tag_id}",
      "definition": "{tag_definition}",
      "true_criteria": "{inclusion_criteria}",
      "false_criteria": "{exclusion_criteria}"
    }],
    "tasks": [{
      "task_key": "{task_key}",
      "instruction": "{task_instruction}",
      "skills": ["{skill_name}"],
      "artifact_paths": ["{artifact_path}"]
    }],
    "required_output_shape": {
      "judgments": [{
        "task_key": "exact supplied task_key",
        "matched_tag_ids": ["zero or more exact registry IDs"],
        "match_rationale":
          "short reason for matched tags, or empty if uncovered",
        "uncovered": {
          "process_name": "lowercase-kebab-case candidate",
          "definition": "reusable process contract",
          "observable_outcome": "required inspectable result",
          "nearest_tag_ids": ["zero or more exact registry IDs"],
          "distinction": "why those tags do not match"
        },
        "confidence": "high | medium | low"
      }]
    }
  }
```

C Corpus Curation Details

C.1 Training Corpus

We construct a training corpus of 1,728 executable agent tasks spanning software engineering, cybersecurity, data processing, office workflows, manufacturing, healthcare, and scientific computing. The training corpus excludes tasks from SkillsBench and Terminal-Bench, including paraphrased variants. Each task provides an instruction, task-specific resources, a containerized execution environment, and a verifier that evaluates task completion. During reinforcement learning, the policy generates interaction trajectories online by carrying out these tasks, allowing training experience to change as the policy develops. Table 4 summarizes the distribution of training tasks across broad domains.

C.2 Task-Type Annotation

Task-type tags describe the operations or workflows required to complete a task, rather than its application domain or the quality of agent behavior. We use a predefined taxonomy of 82 task types, each specified by a definition and explicit inclusion and exclusion criteria.

To assign tasks to this taxonomy, we use GLM-5.2 (GLM-5-Team et al., 2026) as an offline annotator, providing task instructions, names of available skills, and environment artifact paths as input. Hidden tests, reference solutions, and rollout trajectories are excluded from annotation. The annotation prompt (Appendix B.6) grounds each match in required operations and their observable results, rather than shared tools, file formats, or domain terminology. In particular, it distinguishes executing a workflow from merely describing that workflow in a requested document. The model first considers the full taxonomy and, only when no match is found but nearby candidates are identified, reassesses the task against those candidates.

Table 4: Training corpus distribution by broad domain.

Domain	Tasks	Proportion (%)
Software Engineering	543	31.4
Cybersecurity	415	24.0
Data Analytics	294	17.0
Industrial Engineering	95	5.5
Scientific Research	88	5.1
Digital Media	77	4.5
Business Operations	69	4.0
Healthcare	55	3.2
Finance	48	2.8
Mathematics	44	2.5
Total	1,728	100.0

The resulting annotations cover 67 task types across the training corpus, allowing multiple tags per task. These assignments remain fixed throughout training as rubric evaluations update the behavioral evidence associated with each task type. Table 5 illustrates shared task types across domains, multi-label assignments, and the distinction between synthesizing and executing a workflow.

Table 5: Representative training tasks and their task-type annotations.

Task	Required operations	Assigned task types
Bibliography verification	Check citation metadata against authoritative publication records and identify incorrect venues or years.	Evidence conformance assessment
Clinical study assessment	Assess a case-control study against the supplied Newcastle–Ottawa Scale guidelines and report scores with supporting evidence.	Evidence conformance assessment
3D part mass calculation	Isolate the main mesh component, calculate its volume, and convert units before applying the material density to obtain mass.	3D asset analysis; Unit harmonization
Python build repair	Diagnose compatibility failures, apply targeted patches, and rerun the failing tests to verify the repair.	Artifact diagnosis and repair
Deployment command synthesis	Inspect branch configuration and local changes to construct a deployment command without executing it.	Procedure and plan synthesis

D Evaluation Details

D.1 Benchmarks

SkillsBench v1.1 (Li et al., 2026). SkillsBench evaluates agents on expertise-intensive tasks in containerized environments. The evaluation set comprises 87 tasks in the eight domains listed in Table 1. Each task provides instructions, input data, and a task-specific environment, with an official verifier that checks the resulting answer or artifacts. Oracle solutions and hidden verifier tests are not provided to the agent. Our SkillsBench runs use Kilo Code in the curated-skills setting: the

complete expert-curated `skills/` directory for each task is available during execution. A skill contains a `SKILL.md` description and may include scripts, references, or other resources. These benchmark-provided skills are distinct from the rubric-paired skills evolved by ARISE during training. The evolved skills serve as training-time guidance intended to be internalized through policy optimization; they are not provided on either evaluation benchmark.

Terminal-Bench v2.1 (Merrill et al., 2026). Terminal-Bench contains 89 tasks spanning software engineering, system administration, data processing, and scientific computing. Each task specifies an instruction, a containerized environment, programmatic tests, and a time limit. Evaluation uses Terminus-2 under the official settings, with success determined by the verifier from the final task state.

D.2 Evaluation Settings

The context limit in our evaluations is set to the maximum length natively supported by each model. On SkillsBench, all models except GPT-5.5, whose score is taken from the official leaderboard, use thinking-enabled generation at the highest available reasoning effort, with temperature 0.6, top-$p = 0.95$, and top-$k = 20$. Each model request allows up to $32{,}768$ output tokens, and each task attempt has an agent time limit of $10{,}000$ seconds. Interactive tools are disabled during evaluation. For Terminal-Bench, we evaluate comparable-scale models under the official settings and use official leaderboard scores for the remaining models.

D.3 Baseline Configurations

All training-method baselines and ARISE are initialized from the same Qwen3.5-27B checkpoint and use the same training data and policy-training budget. (i) *VADE* (Hu et al., 2026): We adopt only its sampler, estimating difficulty from task outcomes to assign variance-aware sampling priorities. (ii) *OnlineRubrics* (Rezaei et al., 2026): Criteria are elicited by comparing policy responses with reference responses from DeepSeek-V4-Flash (DeepSeek-AI et al., 2026), which also serves as the judge model and reflection model in ARISE. (iii) *RuscaRL* (Zhou et al., 2026): The rubric pool is initialized with criteria collected throughout the training lifecycle of the ARISE model and remains fixed during training. Apart from the adaptations described above, we follow the original papers for VADE sampling hyperparameters, the OnlineRubrics generation schedule, baseline reward weights, and the RuscaRL guidance injection and decay schedule.

D.4 Metrics

Task Pass Rate. For an evaluation set of N tasks with M attempts per task, let $b_{i,a} \in \{0, 1\}$ indicate whether the official verifier confirms success on attempt a of task i. The reported percentage is

$$\text{PassRate} = \frac{100}{NM} \sum_{i=1}^{N} \sum_{a=1}^{M} b_{i,a}. \tag{21}$$

This metric averages verifier-confirmed success across tasks and repeated runs.

Repetitions and Domain Aggregation. Our evaluations use three attempts per task ($M = 3$) on both benchmarks, giving 261 task attempts for SkillsBench and 267 for Terminal-Bench. Domain-level SkillsBench rates apply Equation 21 to tasks in the corresponding official domain. The overall rate pools all tasks, equivalently weighting domain rates by domain size rather than averaging the eight domains equally.

Unsuccessful Attempts. We follow the official evaluation protocol of each benchmark for error handling. Attempts classified as unsuccessful under that protocol contribute zero to the pass rate.

E Evaluation Stability

To assess run-to-run evaluation variability, Table 6 reports individual SkillsBench v1.1 pass rates and their mean and sample standard deviation for the models evaluated in our setup. All three runs

use the same 87 tasks and follow the evaluation protocol in Appendix D, with aggregation consistent with Table 1. Across these evaluations, the lowest ARISE pass rate exceeds the highest pass rate of every comparable-scale baseline, indicating that its observed advantage is consistent across runs.

Table 6: SkillsBench v1.1 pass rates (%) over three evaluation runs, with the mean and sample standard deviation. **Bold values** indicate the best pass rates within comparable-scale models.

Model	Run 1	Run 2	Run 3	Mean $\pm$ Std
Proprietary Models				
GPT-5.4 Mini	35.6	33.3	34.5	34.5 ± 1.1
Claude Opus-4.7	59.8	57.5	58.6	58.6 ± 1.1
Open-Weight Models				
MiniMax-M2.7	28.7	26.4	31.0	28.7 ± 2.3
GLM-5.1	52.9	50.6	57.5	53.6 ± 3.5
Kimi-K2.6	57.5	51.7	56.3	55.2 ± 3.0
DeepSeek-V4-Pro	50.6	51.7	51.7	51.3 ± 0.7
DeepSeek-V4-Pro-0813	59.8	57.5	59.8	59.0 ± 1.3
Qwen3.5-122B-A10B	21.8	23.0	27.6	24.1 ± 3.0
Qwen3.5-397B-A17B	32.2	29.9	28.7	30.3 ± 1.8
Comparable-Scale Models				
Qwen3.5-27B (Base Model)	21.8	24.1	24.1	23.4 ± 1.3
VADE	41.4	36.8	37.9	38.7 ± 2.4
OnlineRubrics	37.9	42.5	40.2	40.2 ± 2.3
RuscaRL	42.5	34.5	41.4	39.5 ± 4.4
ARISE	**46.0**	**44.8**	**46.0**	$\mathbf{45.6} \pm 0.7$

F RUBRIC–SKILL EVOLUTION CASES

Four cases from the first 150 training steps illustrate distinct rubric–skill lifecycles: activating and then hiding guidance while retaining the rubric (Table 7), hiding previously activated guidance before retiring the pair (Table 8), and keeping skills hidden while either retaining their rubrics (Table 9) or retiring the pairs (Table 10). Each case presents the motivating evidence, the paired rubric and skill, and selected lifecycle events. Reported pass rates summarize applicable rubric evaluations within each update window.

Table 7: Artifact specification checking (Verification): guidance is activated and later hidden while the pair is retained.

Evidence. In a feature-planning task, the agent read a format contract but generated headings that did not match the required pattern and omitted dependency fields. Related failures appeared in compliance reports with prescribed templates. The initial criterion drew on nine issue examples and four contrasting examples.

Rubric. When the agent has read an exact output specification, fail if the generated artifact deviates from it and no post-write inspection checks conformance. Pass if the artifact matches the specification or the agent reconciles it against the specification through a read-back or validation check.

Paired skill. Before finalizing, read back the artifact and compare its fields, names, types, headers, and formatting against the documented requirements. Correct deviations and recheck. A successful write confirms file creation, not specification conformance; repeat a successful check only after a relevant modification or a newly discovered deviation.

Lifecycle. Step 10: create the pair with the skill hidden. **Step 20:** activate the skill at a rubric pass rate of 27.7%. **Step 30:** hide the skill at 34.3%, retaining the rubric for evaluation. **Step 100:** retain the pair with the skill hidden at 67.4%. **Step 150:** retain the pair with the skill hidden at 66.0%. The skill text remains unchanged throughout this interval.

Table 8: Tool-response inspection (Verification): guidance is activated and then hidden before the pair is retired.

Evidence. In a telemedicine accessibility task, a text-to-speech tool returned test-mode metadata without the requested audio. The agent wrote placeholder files in place of real audio rather than investigating the missing output. Five issue examples and one contrasting example motivated the new pair.

Rubric. When a tool response lacks the expected data payload, fail if the agent creates placeholder or fabricated artifacts without inspecting the response. Pass if the agent recognizes the missing payload and investigates an alternative way to obtain the real output or reports that it cannot be produced.

Paired skill. Inspect the response content before writing an output file. Check for empty responses, test-mode markers, or metadata without the required artifact. If the payload is absent, investigate alternatives, retry with different parameters, or report the limitation. Do not substitute placeholder files or fabricated content for the missing result.

Lifecycle. Step 40: create the pair with the skill hidden. **Step 50:** activate the skill at 25.7%. **Step 60:** hide the skill at 43.2%, retaining the rubric for evaluation. **Step 100:** retire the rubric–skill pair at 97.2%, removing the criterion from active evaluation.

Table 9: Targeted correction of tool arguments (Debugging): the rubric remains active while its paired skill stays hidden.

Evidence. In tooling-migration and production-scheduling tasks, the agent repeatedly submitted invalid arguments to the `todowrite` tool after receiving schema errors. Rather than using the error messages to correct the argument structure, it retried the same or substantially similar invalid calls.

Rubric. When a tool returns a schema, argument-validation, or format error, fail if the agent retries with identical or substantially similar invalid arguments without inspecting the concrete error. Pass if it inspects the error and corrects the argument structure on the next attempt, or does not retry the failed call.

Paired skill. Read the full error message and identify the invalid argument, such as a missing field, wrong type, or incorrect nesting. Correct the problematic argument before retrying instead of resubmitting the same payload. Use the error details and documented schema to guide the correction.

Lifecycle. Step 80: create the pair with the skill hidden. **Step 90:** retain the pair at 67.8%. **Step 120:** retain it at 89.6%, below the retirement threshold. **Step 150:** retain it at 81.1%. Across subsequent updates, pass rates remain between the activation and retirement thresholds, so the rubric continues to provide feedback while the skill remains hidden and unchanged.

Table 10: Avoiding redundant exploration (Efficiency): a late-added criterion is retired without activating its paired skill.

Evidence. In a portfolio-analysis task, the agent repeatedly reread documentation and data without creating the required deliverables. Other trajectories repeated searches without obtaining new information. Three issue examples and two contrasting examples motivated a criterion focused on redundant exploration.

Rubric. After information gathering begins, fail if the agent repeats reads or searches without obtaining new information or making progress on implementation. Pass if it proceeds to implementation or further inspection yields new information, including targeted rereading that resolves a specific uncertainty.

Paired skill. Once the procedure, inputs, and output requirements are known, proceed to implementation. Reuse information from earlier reads instead of repeating broad searches. If a detail remains unclear, reread the relevant section rather than the entire file, and keep track of resources already inspected.

Lifecycle. Step 130: create the pair with the skill hidden. **Step 140:** retire the pair at 96.5%. The skill was never activated. Although recent failures motivated the criterion, subsequent evaluations met the retirement threshold, so the pair did not remain in the active pool.

G BEHAVIORAL RETENTION AFTER RETIREMENT

To assess whether behavioral performance is maintained after rubric retirement, we track three criteria covering failure diagnosis and adaptation, complete input coverage, and explicit constraint compliance. For each rubric, the fixed task set comprises the tasks sampled during the 10-step training window ending at its retirement. The checkpoints at steps 100 and 150 are evaluated on this same task set with the rubric text and scoring protocol held fixed and without any ARISE-generated skill guidance. Figure 6 shows the original training-window pass rate at retirement alongside these subsequent fixed-task evaluations in three panels with a shared vertical scale.

All three rubrics maintain pass rates above 90% at both later checkpoints, although individual rates fluctuate. These results support the retention of the evaluated behaviors on the selected tasks after their rubrics leave the active pool.

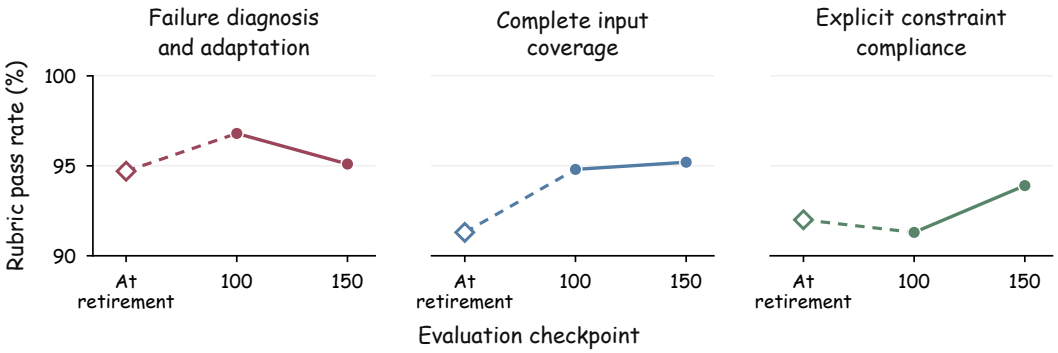


Figure 6: Behavioral retention after rubric retirement. From left to right, the rubrics retire at steps 60, 50, and 80. Open diamonds show training-window pass rates at retirement, while filled circles show fixed-task evaluations without ARISE-generated skill guidance.

H TRAINING TIME ANALYSIS

To complement the rollout-budget comparison, Table 11 reports the average end-to-end time per training step for ARISE and outcome-only RL. Both runs use 32 H800 GPUs with the same training parallelism and a per-step budget of 32 tasks and eight rollouts per task. The average per-step time is approximately 4.2% higher for ARISE, reflecting the overall runtime difference between the two runs. Despite this modest per-step overhead, ARISE reaches comparable performance in one-third as many training steps (Figure 3(b)), implying an estimated 65% reduction in training time to reach that performance level.

Table 11: Average end-to-end training time per step under matched hardware and rollout budgets.

Method	Average time per step (minutes)
Outcome-only RL	16.46
ARISE	17.14