RISE: REASONING-INDUCED SUPERVISED REINFORCEMENT LEARNING WITH EXPERT-COLLABORATIVE TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

For the post-training of Small Language Models (SLMs), Reinforcement Learning (RL) often struggles with limited successful exploration, while Supervised Fine-Tuning (SFT) can lead to rigid imitation and poor generalization. Supervised Reinforcement Learning (SRL) bridges this gap by providing step-wise supervision within RL. Nevertheless, when applied to SLMs operating in multi-turn interactive environments, existing SRL methods remain impractical due to their neglect of reasoning quality in reward construction and heavy reliance on costly expert LLMs. We propose **RISE**, a **R**easoning-**I**nduced **S**upervised Reinforcement Learning framework with **E**xpert-Collaborative Training. RISE introduces a reasoning-aware SRL approach that incorporates a reasoning-induced information gain reward and an adaptive supervision correction mechanism for fine-grained supervision. Moreover, RISE establishes an expert-collaborative training framework that jointly trains a *Decision Maker* for making primary decisions and a lightweight *Verifier* for action reliability assessment, enabling selective expert invocation only for unreliable decisions. Using two 1.5B-scale SLMs as the *Decision Maker* and *Verifier*, and pairing them with different Expert LLMs, we conduct extensive experiments on the ALFWorld and BabyAI environments, showing that RISE matches or surpasses Expert LLMs' task performance while reducing Expert LLM invocation cost by $>$50% during training and $>$33.33% during inference.

1 INTRODUCTION

Large Language Models (LLMs) have shown growing potential as decision-making agents (Li et al., 2026; Xi et al., 2026), but their high training and inference costs hinder practical deployment. Recently, pre-trained Small Language Models (SLMs), in contrast, have shown considerable practical value across diverse domains (Wang et al., 2025). Yet, effectively post-training SLMs to operate as fully autonomous agents in complex interactive tasks remains a formidable challenge.

A prominent training paradigm leverages Reinforcement Learning (RL), particularly group-based algorithms such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024), to optimize the policy using the final verifiable outcomes of generated rollouts (Liu et al., 2025b; Feng et al., 2026; Yu et al., 2026; Zheng et al., 2025). However, these outcome-based RL methods fundamentally depend on successful exploration, a requirement that proves particularly challenging for SLMs. Given the inherent capacity limitations of SLMs, it is not uncommon for all sampled trajectories within a group to receive zero reward during RL training (Zhang et al., 2026b;a), making it exceedingly difficult to derive informative policy gradients.

Another viable alternative is Supervised Fine-Tuning (SFT), which optimizes the policy by minimizing the prediction loss across expert trajectories (Liu et al., 2025a; Lin et al., 2025; Fu et al., 2025; Xia et al., 2025b). Nevertheless, as agents rarely receive explicit supervision on why a selected action should be preferred over other feasible actions (Xia et al., 2025b), purely SFT-based approaches can cause models to overfit to specific behavioral patterns (Xia et al., 2025a; Qiao et al., 2025), thereby encouraging rigid imitation and substantially limiting their generalization ability. This motivates a unified framework that leverages the structured expert guidance of SFT to alleviate the exploration bottleneck, while harnessing the flexible policy updates of RL to avoid rigid imitation.

Code is publicly available at https://anonymous.4open.science/r/RISE-46BC.

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.

In this context, Supervised Reinforcement Learning (SRL) (Deng et al., 2026; Gao et al., 2026) has been proposed. By turning sparse trajectory-level supervision into fine-grained action-level feedback, and optimizing the policy with GRPO using denser rewards, SRL makes challenging long-horizon interactive decision-making tasks substantially more learnable and data-efficient. However, as highlighted in Section 2, SRL faces two key challenges when applied to SLMs: (1) its action-level reward mechanism overlooks the quality of reasoning, which may reward actions supported by flawed reasoning while undervaluing useful reasoning that leads to an imperfect action (e.g., Cases ② and ③ in Fig. 1), and (2) its heavy reliance on high-quality supervision for action correctness makes it impractical for standalone SLMs, whereas repeatedly invoking stronger LLMs for such supervision incurs substantial cost.

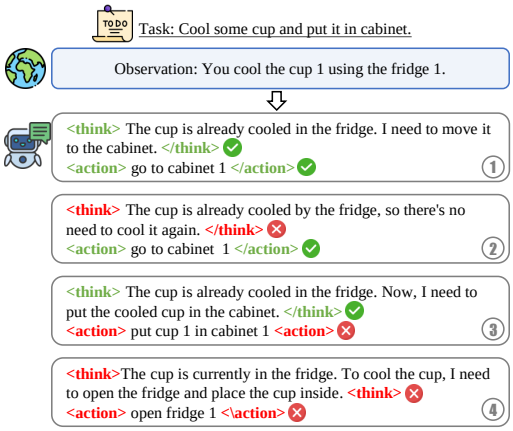

Figure 1: Erroneous supervision signals caused by the neglect of reasoning quality in action-level rewarding mechanisms.

Accordingly, we propose RISE, a **R**easoning-**I**nduced **S**upervised Reinforcement Learning framework with **E**xpert-Collaborative Training, which improves the decision-making capability of SLM agents while substantially reducing the cost of expert invocation. Specifically, the main contributions include:

- **A reasoning-aware SRL approach with convergence guarantee.** We propose an SRL approach with a novel reasoning-induced information gain reward mechanism, which separately evaluates the contributions of reasoning and action to action correctness, and accordingly tailors fine-grained supervision. Moreover, we incorporate adaptive supervision correction to ensure the convergence of GRPO-based optimization when group sampling provides insufficiently informative learning signals.
- **An expert-collaborative training framework with selective expert intervention and high-quality supervision on policy optimization.** We introduce a training framework composed of an SLM as the *Decision Maker*, which is optimized with high-quality supervision from the reasoning-aware SRL, a lightweight *Verifier*, which is trained to reliably determine when expert intervention is needed, and an LLM as the *Expert* for intervention at critical steps.
- **Cost-effective training and inference performance across diverse complex interactive tasks.** Using two 1.5B-scale SLMs as the *Decision Maker* and *Verifier*, and pairing them with different Expert LLMs, we conduct extensive experiments on the ALFWorld and BabyAI environments. Across varying settings, RISE consistently matches or surpasses Expert LLMs' task performance, while reducing the Expert LLM invocation cost by >50% during training and >33.33% during inference.

2 Motivations

2.1 Importance of Reasoning Quality in Supervising SRL Training

Due to the neglect of reasoning quality in action rewarding, SRL may suffer from reduced group sampling efficiency in multi-turn interactive decision-making environments, especially for SLMs. To confirm this, we conduct a controlled experiment by swapping the reasoning of LLMs and SLMs. Specifically, we first randomly select a set of tasks from the ALFWorld environment (Shridhar et al., 2021), on which the stronger LLM *Kimi-2.6* (Moonshot AI, 2026) always succeeds, whereas the weaker SLM Qwen2.5-1.5B-Instruct (hereafter *Qwen2.5-1.5B*) (Yang et al., 2024) consistently fails. Then, at each interaction step of each task, we replace only the reasoning while keeping the action-generating model unchanged. That is, for *Kimi-2.6*, we use the lower-quality reasoning generated by *Qwen2.5-1.5B* under the same interaction prompt as a fixed prefix, and let *Kimi-2.6* generate the subsequent actions, and vice versa for *Qwen2.5-1.5B*.

The measured task success rates of the models under higher-quality (LLM) and lower-quality (SLM) reasoning are shown in Fig. 2. In sharp contrast to the LLM's 100% success rate with its original reasoning, its success rate drops to only 29.1% when conditioned on the SLM's reasoning. Conversely, when conditioned on the LLM's reasoning, the SLM's task success rate surges from 0% to 72.1%. This means the high quality of reasoning is key to shepherding the action correctness of models. However, as current SRL methods neglect reasoning quality in action-level rewarding, they may misalign reasoning effectiveness with action correctness during group sampling, as shown in Fig. 1.

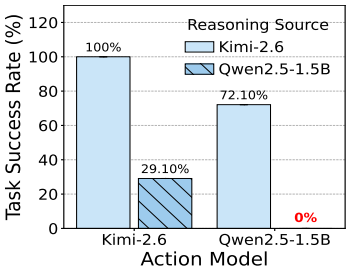

Figure 2: Success rates of models conditioned on high- and low-quality reasoning.

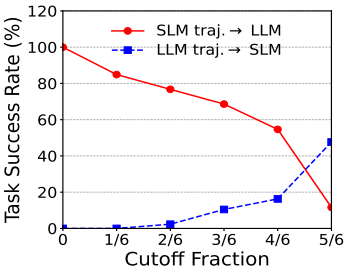

Figure 3: Continued inference from successful LLM and failed SLM trajectories.

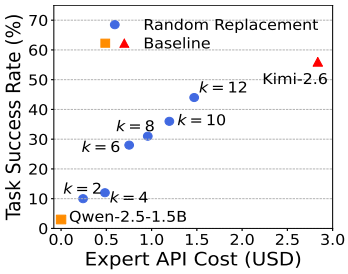

Figure 4: Performance-cost trade-off of LLM-SLM collaboration under varying k values.

As a result, low-quality reasoning may receive high rewards due to accidentally correct actions, introducing noisy positive signals, whereas high-quality reasoning may receive low rewards due to an imperfect action, suppressing potentially useful reasoning signals. This motivates a fine-grained reward mechanism that separately evaluates reasoning and action quality, as detailed in Section 3.2.

2.2 Performance-Cost Trade-Off in Expert-Collaborative Decision-Making

Albeit inexpensive in inference, SLMs alone often underperform due to limited capacity, while repeatedly invoking LLMs for expert supervision in SRL incurs substantial cost. We therefore investigate the potential of LLM-SLM collaboration for better performance-cost trade-off in decision-making. Specifically, we first delve into whether effective collaboration requires continuous high-quality expert involvement throughout interaction. We truncate the successful LLM and failed SLM reasoning trajectories collected in Section 2.1 using cutoff fractions $\{i/6\}_{i=0}^{5}$, retain the first $i/6$ interaction steps per trajectory, and let the SLM and LLM resume inference from the cutoff positions, respectively. We then measure their task success rates under each cutoff fraction, as shown in Fig. 3.

When the SLM continues from successful LLM trajectory prefixes (i.e., `LLM traj→SLM` in Fig. 3), its task success rate increases with the cutoff fraction, as retaining more correct decisions places it in favorable states closer to task completion. In contrast, when the LLM continues from failed SLM trajectory prefixes (i.e., `SLM traj→LLM` in Fig. 3), its success rate decreases with the cutoff fraction because accumulated decision errors lead to less favorable states and leave fewer steps for recovery. These observations suggest that the SLM can make effective subsequent decisions when kept on a favorable reasoning trajectory, whereas accumulated errors may hinder recovery even for a stronger LLM, motivating us to selectively invoke the expert LLM to keep the SLM on favorable states.

We thereby further use several 30-step failed reasoning trajectories generated by the SLM as the basis for making decisions. For each trajectory, we randomly select k interaction steps for the LLM to intervene, while the SLM handles the remaining 30-k steps, until task completion or reasoning end. Then, we measure the task success rates and inference costs of this random LLM-SLM collaborative strategy under varying k values, as shown in Fig. 4. The results of the standalone *Qwen2.5-1.5B* and *Kimi-2.6* are for reference. As expected, both success rate and inference cost increase with k (i.e., more LLM-handled decisions). Interestingly, with only k=10, the strategy already achieves more than half the success rate of *Kimi-2.6* while incurring less than half its inference cost, suggesting that even random expert intervention at a few interaction steps benefits decision-making performance. This motivates a selective expert-collaborative training framework that optimally determines expert intervention at critical steps for cost-effective decision-making, as detailed in Section 3.3.

3 Methodology

3.1 Preliminaries

We reformulate SRL for interactive environments with multi-step decision-making. Given a set of successful reasoning trajectories, we decompose each trajectory into a dataset containing N training instances $\mathcal{D} = \{(x_n, a_n^*)\}_{n=1}^{N}$, where x_n denotes the policy input at the n-th interaction step, including the task instruction and preceding interaction context, and a_n^* is the corresponding supervision action. Thus, each instance in $\mathcal{D}$ provides supervision for a single decision step. For notational simplicity, we omit the instance index n hereafter and denote a generic training instance as $(x, a^*) \sim \mathcal{D}$. For each instance, the old policy $\pi_{\theta_{\text{old}}}$ samples a group of G candidate rollouts, $o^i = (z^i, \hat{a}^i) \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$, for $i = 1, \ldots, G$, where z^i denotes the model-generated reasoning process and $\hat{a}^i$ is the resulting action. The policy is then optimized using the GRPO objective:

$$J_{\mathrm{SRL}}(\theta)=\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o^i|}\sum_{t=1}^{|o^i|}\min\left(\rho_{i,t}(\theta)A_i,\mathrm{clip}(\rho_{i,t}(\theta),1-\epsilon_c,1+\epsilon_c)A_i\right)-\beta D_{\mathrm{KL}}\left(\pi_\theta\|\pi_{\mathrm{ref}}\right)\right], \quad (1)$$

where $\rho_{i,t}(\theta)=\pi_\theta(o_t^i \mid x,o_{<t}^i)/\pi_{\theta_{\mathrm{old}}}(o_t^i \mid x,o_{<t}^i)$ is the token-level probability ratio between the updated and old policies, ϵ_c controls the clipping range, and β controls the strength of the KL regularization. For the G rollouts sampled from the same input, the group-relative advantage is computed as $A_i=\frac{r_i-\mathrm{mean}(\mathbf{r})}{\mathrm{std}(\mathbf{r})+\epsilon_s}$, where $\mathbf{r}=\{r_i\}_{i=1}^G$ and ϵ_s is a small constant for numerical stability.

Following the original SRL formulation (Deng et al., 2026), each rollout is evaluated according to the sequence similarity between its predicted action $\hat{a}^i$ and the corresponding supervision action a^*. We tokenize both actions at the word level and use `difflib.SequenceMatcher` to identify non-overlapping matched words (denoted as $M(\hat{a}^i,a^*)$). The reward per rollout is computed as

$$r_i=\frac{2M(\hat{a}^i,a_n^*)}{|\hat{a}^i|+|a^*|}, \quad (2)$$

where $|\hat{a}^i|$ and $|a^*|$ denote the corresponding numbers of words. The reward lies in [0,1], while rollouts with invalid formats are assigned a penalty of -0.1. SRL further employs dynamic sampling to filter out training instances that provide weak learning signals. Specifically, when the rewards of the G rollouts sampled from the same input are nearly identical, the corresponding group-relative advantages (i.e., A_i) become negligible and contribute little to policy optimization. Therefore, a training instance is retained only if the standard deviation of the corresponding rollout rewards exceeds a positive threshold ϵ_d: $\sqrt{\frac{1}{G}\sum_{i=1}^{G}(r_i-\mathrm{mean}(\mathbf{r}))^2}>\epsilon_d$. To maintain a fixed batch size, sampling and filtering are repeated until a sufficient number of valid training instances are collected.

3.2 Reasoning-Aware Supervised Reinforcement Learning

As illustrated in Section 2.1, effective SRL training depends on both reasoning and action quality. Accordingly, we propose a reasoning-induced reward mechanism that separately evaluates both to provide more reliable supervision for SRL training.

3.2.1 Reasoning-Induced Information Gain

Given that information gain can be used to measure how much the generated reasoning increases the policy's confidence in taking actions (Wang et al., 2026), we introduce reasoning-induced information gain to explicitly quantify reasoning quality. Specifically, we use the old policy $\pi_{\theta_{\mathrm{old}}}$ to evaluate how much a sampled reasoning z^i supports the corresponding supervision action with L tokens (denoted as $a^*=\{v_j^*\}_{j=1}^L$). Its length-normalized log probability conditioned on z^i is defined as

$$\bar{\ell}_{\theta_{\mathrm{old}}}(a^* \mid x,z^i)=\frac{1}{L}\sum_{j=1}^{L}\log\pi_{\theta_{\mathrm{old}}}\left(v_j^* \mid x,z^i,v_{<j}^*\right), \quad (3)$$

where $v_{<j}^*$ denotes the prefix preceding position j. Let $\varnothing$ denote the empty-reasoning baseline, where the supervision action is evaluated without conditioning on any generated reasoning. The reasoning-induced information gain of z^i is defined as

$$g^i=\bar{\ell}_{\theta_{\mathrm{old}}}(a^* \mid x,z^i)-\bar{\ell}_{\theta_{\mathrm{old}}}(a^* \mid x,\varnothing). \quad (4)$$

A positive g^i indicates that the sampled reasoning increases the policy's confidence in the supervision action, whereas a negative g^i indicates that it shifts the policy away from the supervision action. Further exploratory experiments validating this metric are provided in Appendix E.1.

3.2.2 Separate Rewards and Advantages for Reasoning and Action

To enable reasoning-aware rewarding, we assign distinct rewards to reasoning segments within `<think></think>` and action segments within `<action></action>`. Specifically, for a reasoning segment, we convert the reasoning-induced information gain defined in Eq. (4) into a reasoning-level reward via clipping: $r_i^{\mathrm{r}}=[g^i]_+=\max(0,g^i)$. Thus, only reasoning that increases the policy's confidence in the supervision action receives a positive reward, thereby preventing uninformative or harmful reasoning traces from receiving positive relative credit after group-wise normalization. For an action segment, we directly use the action-matching reward defined in Eq. (2), denoted as r_i^{a}.

Since the reasoning and action rewards are derived from log-probability differences and bounded sequence similarity, respectively, they operate on different scales. Accordingly, we further normalize them separately within each sampled group. Specifically, for each segment $j\in\{\mathrm{r},\mathrm{a}\}$, we respectively normalize the corresponding rewards as: $\tilde{r}_i^j=\frac{r_i^j-\mathrm{mean}(\mathbf{r}^j)}{\mathrm{std}(\mathbf{r}^j)+\epsilon_s}$, where $\mathbf{r}^j=\{r_i^j\}_{i=1}^G$, and ϵ_s is a small

constant for numerical stability. Rollouts with invalid formats are assigned a penalty of -0.1 and excluded from the normalization statistics. This separate normalization preserves the relative quality of reasoning and actions while preventing interference caused by their different reward scales.

To assign stronger credit to rollouts that exhibit both high-quality reasoning and well-aligned actions, we define the segment-wise advantages as

$$A_i^{\mathrm{r}} = \tilde{r}_i^{\mathrm{r}} + \mathbb{I}[\tilde{r}_i^{\mathrm{a}} > 0 \wedge \tilde{r}_i^{\mathrm{r}} > 0]\tilde{r}_i^{\mathrm{a}}, \qquad A_i^{\mathrm{a}} = \tilde{r}_i^{\mathrm{a}} + \mathbb{I}[\tilde{r}_i^{\mathrm{a}} > 0 \wedge \tilde{r}_i^{\mathrm{r}} > 0]\tilde{r}_i^{\mathrm{r}}, \tag{5}$$

where A_i^{r} and A_i^{a} are assigned to the reasoning and action tokens, respectively. $\mathbb{I}[\tilde{r}_i^{\mathrm{a}} > 0 \wedge \tilde{r}_i^{\mathrm{r}} > 0] \in \{0, 1\}$ indicates whether the normalized action and reasoning rewards of the i-th rollout are both above the group average. Consequently, high-quality reasoning can still receive supervision when the final action is incorrect (i.e., $A_i^{\mathrm{r}} \geqslant \tilde{r}_i^{\mathrm{r}}$), whereas a correct action does not inflate the reward to its maximum if its reasoning quality is below average (i.e., $\mathbb{I}[\tilde{r}_i^{\mathrm{a}} > 0 \wedge \tilde{r}_i^{\mathrm{r}} > 0] = 0$).

Finally, we apply the segment-wise advantages defined in Eq. (5) to their corresponding tokens to preserve fine-grained credit assignment between reasoning and action. Let $\mathcal{T}_i^{\mathrm{r}}$ and $\mathcal{T}_i^{\mathrm{a}}$ denote the token index sets of the reasoning and action segments in the i-th rollout, respectively. Based on Eq. (1), we define the clipped surrogate of each segment $j \in \{\mathrm{r}, \mathrm{a}\}$ as:

$$\mathcal{S}_i^j(\theta) = \frac{1}{|\mathcal{T}_i^j|} \sum_{t \in \mathcal{T}_i^j} \min\left(\rho_{i,t}(\theta) A_i^j, \mathrm{clip}\left(\rho_{i,t}(\theta), 1-\epsilon_c, 1+\epsilon_c\right) A_i^j\right), \tag{6}$$

In this way, reasoning tokens are primarily optimized according to reasoning quality, while action tokens are directly guided by action quality, with the consistency-aware advantages allowing additional credit to be shared across the two segments when appropriate.

3.2.3 Adaptive Supervision Correction

Although the reasoning-induced reward provides denser supervision, group-relative optimization may still lack sufficient learning signals when both reasoning and action rewards within a sampled group remain low. Accordingly, we replace the original SRL's dynamic sampling strategy with adaptive supervision correction, whose corrections serve as auxiliary supervision for GRPO-based optimization. Specifically, supervision correction is activated only when none of the sampled actions exactly matches the supervision action (i.e., $\max_i r_i^{\mathrm{a}} < 1$), and the reasoning trace with the highest information gain satisfies $g^k > 0$, where $k = \arg\max_i g^i$. Given the corrected reasoning z^k, we construct a conditional preference $a^* \succ \hat{a}^k \mid (x, z^k)$, encouraging the policy to assign a higher conditional likelihood to the supervision action a^* than to the originally sampled action $\hat{a}^k$ under the same policy-generated reasoning context. Thus, the corresponding correction term is formulated as

$$\mathcal{C}_{\mathcal{G}}(\theta) = \begin{cases} \omega_{\mathcal{G}} \log \sigma\left(\bar{\ell}_\theta(a^* \mid x, z^k) - \bar{\ell}_\theta(\hat{a}^k \mid x, z^k)\right), & \max_i r_i^{\mathrm{a}} < 1 \wedge g^k > 0, k = \arg\max_i g^i \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where $\sigma(\cdot)$ denotes the sigmoid function and $\omega_{\mathcal{G}} = 1 - \max_i r_i^{\mathrm{a}}$ measures the residual discrepancy between the best sampled action and the supervision action. Consequently, stronger correction is applied when the sampled actions remain substantially misaligned with the supervision action, while the correction strength gradually diminishes as the best sampled action approaches exact agreement with the supervision action.

Following Eq. (1), we combine the segment-wise GRPO objective with the auxiliary supervision correction signal to formulate the overall optimization objective of the reasoning-aware SRL in RISE as

$$J_{\mathrm{RISE}}(\theta) = \mathbb{E}_{\mathcal{G}}\left[\frac{1}{G}\sum_{i=1}^{G}\left(w_{\mathrm{r}}\mathcal{S}_i^{\mathrm{r}}(\theta) + w_{\mathrm{a}}\mathcal{S}_i^{\mathrm{a}}(\theta)\right) + \lambda_{\mathrm{corr}}\mathcal{C}_{\mathcal{G}}(\theta) - \beta D_{\mathrm{KL}}\left(\pi_\theta \| \pi_{\mathrm{ref}}\right)\right], \tag{8}$$

where $\lambda_{\mathrm{corr}} \geqslant 0$ controls the strength of the auxiliary supervision correction. The reasoning and action surrogates are weighted by w_{r} and w_{a}, respectively, where $w_{\mathrm{r}}, w_{\mathrm{a}} \geqslant 0$ and $w_{\mathrm{r}} + w_{\mathrm{a}} = 1$. This formulation preserves policy-generated reasoning while leveraging supervision knowledge only when sampled rollouts fail to provide sufficient action-level guidance.

3.3 Expert-Collaborative Training Framework

Inspired by the observations in Section 2.2, we further develop a selective expert-collaborative training framework for cost-effective decision-making, which consists of the following two phases.

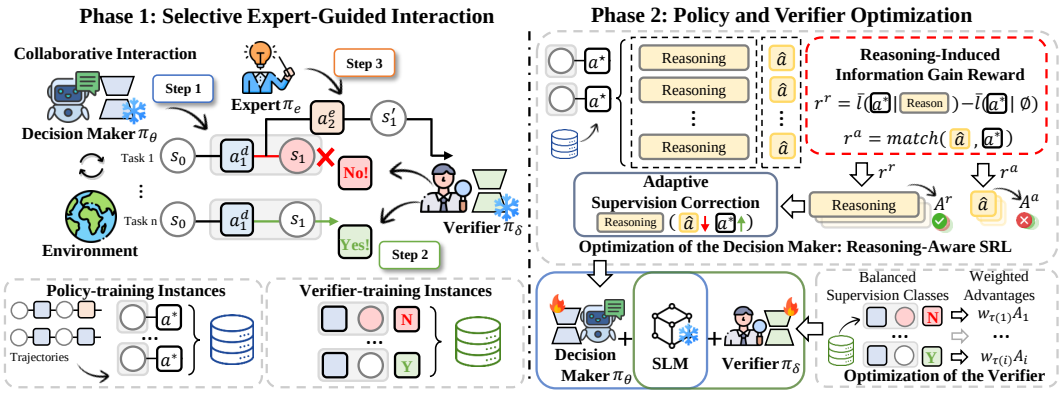

Figure 5: Overview of the expert-collaborative training framework.

3.3.1 Phase I: Selective Expert-Guided Interaction

First, we propose an expert-guided interaction framework that enables an SLM to make most decisions autonomously while selectively invoking an expert LLM for critical decisions. As shown in the left half of Fig. 5, the framework consists of three primary components: an SLM as the *Decision Maker* (denoted as π_θ), a lightweight *Verifier* (denoted as π_δ), and an LLM as the *Expert* (denoted as π_e). Specifically, π_θ is the primary decision-making agent throughout interaction. π_δ assesses the reliability of candidate decisions through step-wise action evaluation and triggers a rollback when a decision is rejected, a strategy shown to effectively mitigate error propagation in multi-step LLM agent interaction (Li et al., 2025). π_e provides corrective actions when π_δ judges that expert intervention is necessary. To save training and deployment overhead, π_θ and π_δ are parameterized by two separate LoRA adapters over a shared frozen SLM backbone, enabling role-specific adaptation with a few additional trainable parameters. The collaborative interaction at each step t proceeds as:

- **Step 1: Candidate Decision.** π_θ first proposes a candidate action a_t^d, which is tentatively executed to obtain the resulting environment state s_{t+1}.
- **Step 2: Reliability Verification.** Given the candidate transition, π_δ outputs a verification decision $y_t \in \{\texttt{yes}, \texttt{no}\}$. If $y_t = \texttt{yes}$, the transition is retained, and the interaction proceeds from s_{t+1}.
- **Step 3: Expert Invocation.** If $y_t = \texttt{no}$, the interaction is rolled back to s_t, and π_e is queried to generate an alternative action a_t^e, which is executed instead. Otherwise, the candidate transition is retained, and π_e proceeds with the next decision from s_{t+1}.

To improve the efficiency of verification and expert decision-making in Steps 2 and 3, we further design a trajectory-level transition memory mechanism. Following the above steps, π_θ makes most decisions autonomously, while π_e only intervenes when decisions are identified as unreliable by π_δ. After each episode, successful collaborative trajectories are decomposed into step-level policy-training instances for subsequent optimization. Meanwhile, π_δ's assessments along successful trajectories are retained as supervision for its subsequent training. Further details for the memory mechanism and the prompts of these components are provided in Appendices C.1 and C.2, respectively.

3.3.2 Phase II: Policy and Verifier Optimization

Phase I produces two types of step-level supervision from successful collaborative trajectories: policy-training instances for π_θ, and verification-training instances for π_δ. Phase II uses the instances to optimize the components separately according to their different decision spaces:

- **Optimization of the *Decision Maker*.** We optimize π_θ on the policy-training instances using the reasoning-aware SRL introduced in Section 3.2. Specifically, as illustrated in the right half of Fig. 5, the reasoning-induced information gain reward explicitly evaluates the quality of policy-generated reasoning. We further employ segment-wise reasoning and action advantages to provide fine-grained credit assignment to their respective tokens. Finally, adaptive supervision correction provides auxiliary supervision when sampled rollouts fail to offer sufficient action-level learning signals. The stationary convergence of this procedure is established in Appendices B.1 and B.2.
- **Optimization of the *Verifier*.** Since π_δ performs a binary reliability decision over yes and no, rather than open-ended actions, we optimize it with the standard SRL (Section 3.1), using the verifier-training instances as step-level supervision. Specifically, as illustrated in the right half of Fig. 5, we address two sources of bias in the *Verifier*'s supervision: bias toward accepting candidate decisions

or invoking the *Expert*, and disproportionate influence from trajectories dominated by rejection decisions. To mitigate the former, we balance the two supervision classes so that each update contains equal numbers of yes and no instances. To mitigate the latter, we introduce trajectory-level weighting: for a successful trajectory τ with verifier-training instances $\mathcal{I}_\tau$, we define its rejection ratio as $q_\tau = \frac{1}{|\mathcal{I}_\tau|} \sum_{j \in \mathcal{I}_\tau} \mathbb{I}[y_j = \text{no}]$ and assign the trajectory weight $w_\tau = 1 - \alpha q_\tau$, where $\alpha \in [0, 1)$ controls the strength of down-weighting. This weight is applied after standard group-wise advantage normalization: for instance i, its normalized advantage A_i is replaced by $\widetilde{A}_i = w_{\tau(i)} A_i$, where $\tau(i)$ denotes the trajectory containing i. Since $0 < w_\tau \leqslant 1$, this weighting preserves the sign of each instance-level optimization signal while down-weighting rejection-heavy trajectories. The resulting weighted binary SRL inherits the convergence analysis established above, as shown in Appendix B.3.

After Phase II, the updated *Decision Maker* and *Verifier* are deployed in the next round of Phase I to collect new collaborative trajectories under an expert-intervention budget. We alternate the two phases for multiple rounds, such that the supervision collected in each interaction phase is subsequently used to update both components while controlling the expert cost of data collection. The complete training procedure is summarized as Algorithm 1 in Appendix C.3.

4 Performance Evaluation

4.1 Experimental Setup

Benchmarks. We conduct experiments on the ALFWorld (Shridhar et al., 2021), a text-based embodied environment where an agent completes a natural-language goal by iteratively observing the environment and executing actions over multiple interaction steps. The results on more benchmarks (e.g., BabyAI) and Expert LLMs (e.g. Kimi-2.6) are provided in Appendices E.2 and E.3. For ALFWorld, we report results on both in-distribution (ID) and out-of-distribution (OOD) tasks.

Baselines. (i) **Base SLM**, standalone *Qwen2.5-1.5B-Instruct* (Yang et al., 2024); (ii) **Expert LLM**, standalone *GLM-5* (Zeng et al., 2026) in non-thinking mode and accessed via API; (iii) **GRPO** (Shao et al., 2024), which estimates relative advantages among sampled trajectory groups for critic-free policy optimization; (iv) **GiGPO** (Feng et al., 2026), which estimates relative advantages at both episode and step levels for fine-grained credit assignment in long-horizon agent training; (v) **SFT**, which adopts a ReAct-style interaction format and imitates both the reasoning and actions of the expert; (vi) **SFT+GRPO** (Zhang et al., 2026c), which initializes the policy via SFT before GRPO training; (vii) **SRL** (Deng et al., 2026), which trains the policy with rewards based on the similarity between generated and expert actions; (viii) **STEER** (Lee et al., 2026), a confidence-based collaboration design that queries the expert LLM when the SLM confidence falls below a predefined threshold; and (ix) **Random** (k=8) / **Random** (k=10), the k-step random LLM-SLM collaborative strategy introduced in Section 2.2. For fair comparison, all methods can only train the Base SLM and use the same LoRA-based adaptation setting. More detailed configurations are provided in Appendix D.

Metrics. The metrics for evaluating task performance and cost efficiency are listed as follows.

Name	Representation	Definition
Task Success Rate (SR)	$\text{SR} = \frac{N_{\text{succ}}}{N_{\text{task}}} \times 100\%$	The ratio of tasks successfully completed within the maximum interaction horizon, where N_{succ} and N_{task} denote the number of successfully completed tasks and the total number of evaluated tasks, respectively
Incremental Benefit Per Cost (IBC)	$\text{IBC} = \frac{\text{SR} - \text{SR}_{\text{SLM}}}{C_{\text{API}}}$	The task-performance improvement over Base SLM achieved per unit of expert API cost during inference (Aggarwal et al., 2024), where SR_{SLM} denotes the task success rate of Base SLM
Expert API Cost	$C_{\text{API}} = \frac{N_{\text{in}} P_{\text{in}} + N_{\text{out}} P_{\text{out}}}{10^6}$	The monetary cost incurred by LLM calls according to the official API pricing at the time of our experiments, where N_{in} and N_{out} denote the total numbers of input and output tokens, respectively, and P_{in} and P_{out} denote the corresponding prices per million tokens (Ong et al., 2025)
Training Time	$T_{\text{train}} = t_{\text{end}} - t_{\text{start}}$	The end-to-end wall-clock time required to complete the entire training process. t_{start} and t_{end} denote the wall-clock timestamps at the beginning and completion of the training process, respectively

4.2 Task Performance and Cost-Effectiveness

Table 1 summarizes the overall performance of the methods under different categories. The task performance of all the methods is evaluated on the same ID and OOD tasks in terms of task success

Table 1: Comparison of task performance, expert API cost and training efficiency

Category	Method	Success Rate (pp)		IBC (pp/USD)		Expert API Cost of Training ($)	# of Reasoning Traj.	Training Time (h)
		ID	OOD	ID	OOD			
Standalone Models	Base SLM	4.29	0.75	–	–	–	–	–
	Expert LLM	72.86	76.87	43.12	51.08	–	–	–
RL-based Methods	GRPO	7.86	7.46	–	–	–	–	107.4
	GiGPO	9.52	9.95	–	–	–	–	74
Supervision-based Methods	SFT	25.00	25.37	–	–	0.94	50	1.33
	SFT+GRPO	19.30	14.20	–	–	0.94	50	61.33
	SRL	21.43	20.15	–	–	0.94	50	3.06
Expert-Collaborative Methods	Random (k=8)	39.28	37.31	47.28	50.08	–	–	–
	Random (k=10)	45.71	43.28	47.06	48.32	–	–	–
	STEER	62.10	67.21	43.46	54.02	–	–	–
	RISE	**74.29**	**68.66**	**120.68**	**99.86**	**0.94**	**112**	**6**

rate (percentage point, pp in short). For expert-collaborative methods, we additionally report their IBC scores (pp/USD) to quantify how effectively the methods can translate the inference-stage expert API cost into task performance improvement, where a higher IBC indicates more cost-effective use of expert assistance. For supervision-based methods, the reasoning trajectories used for training are collected entirely by the Expert LLM under the same training-stage expert API cost budget as RISE. Accordingly, for this category, we additionally report the training-stage expert API cost (USD), the number of reasoning trajectories used during training, and the total training time (hour, h in short).

Among the methods that do not rely on expert supervision, although GRPO and GiGPO improve the success rate of Base SLM on both ID and OOD tasks, they consume substantially longer training time than the other categories of methods, making them impractical for real-world deployment. These results align with prior findings: given the limited capacity of SLMs, they struggle to derive informative policy gradients through self-exploration alone (Zhang et al., 2026b;a).

In comparison, supervision-based methods achieve much higher task success rate than the RL-based ones, highlighting the value of expert reasoning for agent decision-making. However, with the same reasoning trajectories, the success rates follow SFT>SRL>SFT+GRPO on the tasks. This is mainly because, in the ALFWorld environment where the reasoning process dictates task understanding, SFT can guide the agent to complete several tasks by directly following the structured trajectories. SRL, in contrast, overlooks the quality of reasoning in rewarding, resulting in insufficient exploitation of the same expert supervision signals. Notably, SFT+GRPO instead deteriorates performance despite requiring 61.33 hours of training. This suggests that the benefits of SFT and RL cannot simply be combined: without an effective mechanism for deriving informative learning signals, GRPO may drive the SLM away from the beneficial expert behaviors acquired via SFT.

As for expert-collaborative methods, we further observe that even Random (k=8) and Random (k=10) already outperform all the RL-based and supervision-based methods, reaffirming the value of LLM-SLM collaboration for decision-making. By more reliably determining expert intervention, both STEER and RISE substantially improve the agent's decision-making capability over the others, matching the task performance of the Expert LLM. However, when comparing their IBC scores, STEER is no better than the Expert LLM. These results imply that STEER can barely exploit the inexpensive inference of SLMs for decision-making, but degrades into frequently calling the Expert LLM for costly decision-making. The underlying reason is that STEER relies solely on the Base SLM's internal confidence to trigger expert intervention, disregarding an action's suitability for current environment state or its downstream consequences. Consequently, overconfident but incorrect actions may result in extra expert calls for failure recovery, whereas underconfident but correct actions may trigger unnecessary expert calls, ultimately lowering the cost-effectiveness of STEER.

In contrast, the IBC scores of RISE are almost 3× or 2× those of the Expert LLM, meaning it matches the Expert LLM's task performance at half or one-third of the cost. This cost-efficiency stems from the selective expert intervention mechanism in RISE. Specifically, as illustrated in Fig. 5, the *Verifier* judges the reliability of candidate actions based on current environment state and action evaluation, allowing the *Decision Maker* (SLM) to handle most interaction steps independently while invoking the *Expert* (LLM) only when necessary. Consequently, both the SLM and LLM contribute high-quality supervision rather than relying solely on costly expert LLM inference. This also explains why, under the same $0.94 training-stage expert API cost budget, RISE collects 112 collaborative reasoning trajectories, more than 2× the 50 reasoning trajectories available to the supervision-based methods. Throughout the collection process, these trajectories are decomposed into policy-training

and verifier-training instances, which respectively train a more capable *Decision Maker* and *Verifier*, forming a virtuous cycle that further raises the cost-effectiveness of RISE.

4.3 Ablation Studies

Ablation of Expert-Collaborative Training Framework. We use the complete RISE that trains both the *Decision Maker* and *Verifier* as the baseline (denoted as *Train-Train*), and individually enable or disable the training of the two components to construct three variants: *Base-Base*, *Base-Train* and *Train-Base*. Then, in Table 2, we report the task success rate and IBC scores of these variants.

Table 2: Ablation study of expert-collaborative training framework.

Decision Maker	*Verifier*	SR (pp)		IBC (pp/USD)	
		ID	OOD	ID	OOD
Base	Base	51.43	47.76	39.98	42.01
Base	Train	53.57	50.75	41.58	45.78
Train	Base	60.71	54.48	63.89	61.54
Train	**Train**	**74.29**	**68.66**	**120.68**	**99.86**

We observe that disabling the training of the *Verifier* (i.e., *Train-Base*) causes a noticeable performance degradation, especially on IBC scores. This directly reflects the importance of incorporating an experienced *Verifier* for reliably identifying the critical interaction steps for expert intervention. We also notice that even though the training of the *Verifier* is disabled, the IBC scores of *Train-Base* remain higher than those of the Expert LLM in Table 1. This indicates that an untrained *Verifier* already provides a basic capability for coordinating the *Decision Maker* with the Expert LLM. However, due to the absence of the policy optimization for the *Verifier* introduced in Section 3.3.2, such a vanilla *Verifier* cannot reliably determine the time for expert intervention.

Moreover, we notice that disabling the training of the *Decision Maker* (i.e., *Base-Base* and *Base-Train*) causes far more severe performance degradation than *Train-Base*, especially in IBC scores, almost regardless of whether the *Verifier* is trained. This is expected, as the *Decision Maker* is central to autonomous decision-making and alleviating the reliance on the Expert LLM. If it remains weak, it will significantly constrain the benefit that a more reliable *Verifier* alone can provide.

Ablation of Reasoning-Aware SRL. Given the criticality of the *Decision Maker*, we further investigate the effectiveness of the underlying algorithm that drives its training, the reasoning-aware SRL (RSRL in short) introduced in Section 3.2. Specifically, we construct a variant of RSRL, denoted as RSRL-ASC, by removing the adaptive supervision correction mechanism introduced in Section 3.2.3. Then, to avoid the impact of varying expert guidance, we preclude the SLM-LLM collaboration and instead provide the same pre-collected expert reasoning trajectories for decision-making. Finally, we train the standalone *Decision Maker* with SRL, RSLR and RSRL-ASC, respectively, and collect their success rates on the ID and OOD tasks, as shown in Fig. 6.

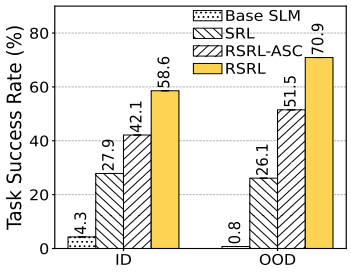

Figure 6: Ablation study of reasoning-aware SRL.

We observe that RSRL-ASC outperforms standard SRL, indicating that our design of reasoning-induced fine-grained rewards and separate advantages provides more effective supervision from sampled rollouts than action-level feedback alone. Moreover, RSRL further improves upon RSRL-ASC, suggesting that the introduced correction mechanism provides additional guidance when group sampling fails to yield informative learning signals, thereby further improving policy performance.

5 Conclusion

In this paper, we have systematically analyzed the importance of reasoning quality to agent decision-making and the impracticality of existing methods for post-training SLMs as fully autonomous agents. To this end, we proposed RISE, a reasoning-induced supervised RL framework with expert-collaborative training for improving the decision-making capability of SLM agents while substantially reducing their reliance on costly expert LLMs. It introduces a reasoning-induced information gain reward and adaptive supervision correction for fine-grained supervision, and an expert-collaborative training framework that jointly trains a *Decision Maker* and a *Verifier* for selective invocation of expert LLMs. Extensive experiments on multiple interactive environments and diverse models demonstrated the cost-effectiveness of RISE for agent decision-making backed by SLMs. In future work, we plan to extend RISE to more challenging chain-of-thought reasoning tasks and investigate multi-expert collaboration to further explore its generality and practicality.

6 AI USE STATEMENT

In this work, we used generative AI tools for paper writing, including spell checking, grammar checking, and polish writing. We did not use generative AI to conduct research, generate research ideas, or draft any section of the manuscript from scratch; all text originates from the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

7 REPRODUCIBILITY STATEMENT

The paper has fully disclosed all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and conclusions of the paper. We have provided a link (https://anonymous.4open.science/r/RISE-46BC) to a downloadable source code with detailed explanations and annotations to support the reproducibility of our work. In addition, the necessary hyperparameter settings are described in Appendix D.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. Automix: Automatically mixing language models. In *Proc. of NeurIPS*, 2024.

Mohammad Hossein Amani, Aryo Lotfi, Nicolas Mario Baldwin, Samy Bengio, Mehrdad Farajtabar, Emmanuel Abbe, and Robert West. Rl for reasoning by adaptively revealing rationales. In *Proc. of ICLR*, 2026.

Xinjie Chen, Minpeng Liao, Guoxin Chen, Chengxi Li, Biao Fu, Kai Fan, and Xinggao Liu. From data-centric to sample-centric: Enhancing LLM reasoning via progressive optimization. In *Proc. of ACL*, 2026.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *Proc. of ICLR*, 2019.

Yihe Deng, I Hsu, Jun Yan, Zifeng Wang, Rujun Han, Gufeng Zhang, Yanfei Chen, Wei Wang, Tomas Pfister, Chen-Yu Lee, et al. Supervised reinforcement learning: From expert trajectories to step-wise reasoning. In *Proc. of ICLR*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Proc. of NeurIPS*, 2026.

Dayuan Fu, Keqing He, Yejie Wang, Wentao Hong, Zhuoma Gongque, Weihao Zeng, Wei Wang, Jingang Wang, Xunliang Cai, and Weiran Xu. Agentrefine: Enhancing agent generalization through refinement tuning. In *Proc. of ICLR*, 2025.

Longxi Gao, Li Zhang, Pengzhi Gao, Wei Liu, Jian Luan, and Mengwei Xu. Gui-shift: Enhancing vlm-based gui agents through self-supervised reinforcement learning. In *Proc. of ICLR*, 2026.

Sangmook Lee, Dohyung Kim, Hyukhun Koh, Nakyeong Yang, and Kyomin Jung. Confidence-guided stepwise model routing for cost-efficient reasoning. In *Proc. of AAAI*, 2026.

Xingzuo Li, Kehai Chen, Yunfei Long, Xuefeng Bai, Yong Xu, and Min Zhang. Generator-assistant stepwise rollback framework for large language model agent. In *Proc. of EMNLP*, 2025.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. In *Proc. of ICLR*, 2026.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. In *Proc. of CVPR*, pp. 19498–19508, 2025.

Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong WANG, et al. Toolace: Winning the points of llm function calling. In *Proc. of ICLR*, 2025a.

Zhihan Liu, Hao Hu, Shenao Zhang, Hongyi Guo, Shuqi Ke, Boyi Liu, and Zhaoran Wang. Reason for future, act for now: a principled architecture for autonomous llm agents. In *Proc. of ICML*, 2024.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Wei Liu, Jian Luan, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. Demystifying small language models for edge deployment. In *Proc. of ACL*, 2025.

Renjie Luo, Jiaxi Li, Chen Huang, and Wei Lu. Through the valley: Path to effective long CoT training for small language models. In *Proc. of EMNLP*, 2025.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. In *Proc. of NeurIPS*, 2024.

Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Wong, Junbo Niu, Chengyu Shen, Runming He, Yanhao Li, et al. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. In *Proc. of ICLR*, 2026.

Moonshot AI. Kimi k2.6: Advancing open-source coding, 2026. URL https://www.kimi.com/en/blog/kimi-k2-6. Accessed: 2026-09-01.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, Mohammed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *Proc. of ICLR*, 2025.

Shuofei Qiao, Zhisong Qiu, Baochang Ren, Xiaobin Wang, Xiangyuan Ru, Ningyu Zhang, Xiang Chen, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agentic knowledgeable self-awareness. In *Proc. of ACL*, 2025.

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

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Proc. of NeurIPS*, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *Proc. of ICLR*, 2021.

Fali Wang, Minhua Lin, Yao Ma, Hui Liu, Qi He, Xianfeng Tang, Jiliang Tang, Jian Pei, and Suhang Wang. A survey on small language models in the era of large language models: Architecture, capabilities, and trustworthiness. In *Proc. of SIGKDD*, 2025.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information gain-based policy optimization: A simple and effective approach for multi-turn llm agents. In *Proc. of ICLR*, 2026.

Zhiheng Xi, Wenxiang Chen, Boyang Hong, Senjie Jin, Rui Zheng, Wei He, Yiwen Ding, Shichun Liu, Xin Guo, Junzhe Wang, Honglin Guo, Wei Shen, Xiaoran Fan, Yuhao Zhou, Shihan Dou, Xiao Wang, Xinbo Zhang, Peng Sun, Tao Gui, Qi Zhang, and Xuanjing Huang. Training large language models for reasoning through reverse curriculum reinforcement learning. In *Proc. of ICML*, 2024.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, et al. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning. In *Proc. of ICLR*, 2026.

Yu Xia, Jingru Fan, Weize Chen, Siyu Yan, Xin Cong, Zhong Zhang, Yaxi Lu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. AgentRM: Enhancing agent generalization with reward modeling. In *Proc. of ACL*, 2025a.

Yu Xia, Yiran Jenny Shen, Junda Wu, Tong Yu, Sungchul Kim, Ryan A. Rossi, Lina Yao, and Julian McAuley. SAND: Boosting LLM agents with self-taught action deliberation. In *Proc. of EMNLP*, 2025b.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. In *Proc. of NeurIPS*, 2026.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. Scaf-grpo: Scaffolded group relative policy optimization for enhancing llm reasoning. In *Proc. of ICLR*, 2026a.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. Bread: Branched rollouts from expert anchors bridge sft & rl for reasoning. In *Proc. of NeurIPS*, 2026b.

Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. Rlvmr: Reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents. In *Proc. of ICLR*, 2026c.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Appendix

List of Contents

A: Related Work
- A.1 RL-Based Methods for LLM Agents
- A.2 SFT-Based Methods for LLM Agents
- A.3 Supervision-Guided RL for LLM Agents

B: Theoretical Analysis and Proofs
- B.1 Regularity and Structural Properties
- B.2 Stationary Convergence of the Decision Maker
- B.3 Verifier Optimization
- B.4 Empirical Training Dynamics

C: Algorithmic Details
- C.1 Trajectory-level Transition Memory
- C.2 Prompt Templates
- C.3 Algorithm Pseudocode

D: Additional Implementation Details

E: Additional Experimental Results
- E.1 Validation of Reasoning-Induced Information Gain
- E.2 Generalization across Expert LLMs
- E.3 Generalization across Interactive Environments

F: Limitations

A Related Work

A.1 RL-Based Methods for LLM Agents

Existing RL-based methods for training LLM agents typically construct rewards from environmental feedback and optimize agent policies with algorithms such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024). Dr.GRPO (Liu et al., 2025b) provided an unbiased variant of GRPO that removes the optimization bias to improve token efficiency while maintaining reasoning performance. GiGPO (Feng et al., 2026) constructed both episode-level and step-level relative advantages through anchor-state grouping, enabling finer-grained credit assignment for multi-turn LLM agents. DAPO (Yu et al., 2026) combined asymmetric clipping, dynamic sampling, token-level policy gradients, and overlong reward shaping to improve training scalability. GSPO (Zheng et al., 2025) defined importance ratios based on sequence likelihood and performed sequence-level clipping and optimization to stabilize large-scale RL training. However, the effectiveness of these methods still largely relies on the quality and diversity of sampled trajectories within a limited rollout budget. When SLMs produce consistently failed trajectories, which is not uncommon due to their capacity limitations, outcome-based rewards cannot distinguish partially useful steps, yielding sparse or uninformative learning signals.

A.2 SFT-Based Methods for LLM Agents

Unlike RL-based approaches, another line of work trains LLM agents through SFT on expert-generated reasoning trajectories, a paradigm widely adopted across tool-use (Liu et al., 2025a), GUI interaction (Lin et al., 2025), etc. Beyond directly imitating successful demonstrations, AgentRefine (Fu et al., 2025) incorporated erroneous actions, environment feedback, and subsequent corrections into refinement trajectories, enabling agents to learn error recovery. SAND (Xia et al., 2025b) utilized self-consistency sampling and execution-guided critiques to synthesize deliberation trajectories that explicitly compare candidate actions before selecting one. However, conventional SFT primarily encourages models to reproduce expert reasoning trajectories at the token level, rather than to understand the underlying decision principles. This reliance on direct imitation can lead to overfitting to trajectory-specific patterns, thus limiting the generalization ability of SLMs.

A.3 Supervision-Guided RL for LLM Agents

Recently, several methods have explored incorporating supervised guidance into RL for LLM agents. By leveraging the structured expert guidance of SFT to alleviate the exploration bottleneck, and harnessing the flexible policy updates of RL to avoid rigid imitation, this paradigm makes challenging long-horizon interactive decision-making tasks substantially more learnable and data-efficient. R^3 (Xi et al., 2024) leveraged correct reasoning demonstrations to progressively shift exploration from near-complete trajectories toward the original problem, making sparse outcome rewards more informative. Following the idea of partial guidance, Building on this paradigm of partial expert guidance, BREAD (Zhang et al., 2026b) branched policy rollouts from adaptively selected expert anchors, while LPPO (Chen et al., 2026) introduced prefix-guided sampling for challenging instances. AdaBack (Amani et al., 2026) further adaptively adjusted the amount of revealed expert rationale based on learning progress. ReLIFT (Ma et al., 2026) further interleaved RL with targeted SFT on examples that are difficult to learn via self-exploration. However, these approaches mainly introduce supervision at the trajectory, prefix, or sample level, and thus provide limited direct guidance for individual reasoning and action steps. To provide more fine-grained supervision, SRL (Deng et al., 2026) and GUI-Shift (Gao et al., 2026) decomposed expert trajectories into logical actions and constructed dense step-wise rewards based on action similarity, thereby directly incorporating expert guidance into RL optimization. Nevertheless, SRL overlooks the quality of reasoning, which may reward actions supported by flawed reasoning while undervaluing useful reasoning that leads to an imperfect action. Moreover, its heavy reliance on high-quality supervision for action correctness makes them impractical for standalone SLMs, whereas repeatedly invoking stronger LLMs for such supervision incurs substantial cost.

B Theoretical Analysis and Proofs

We provide theoretical analysis of RISE from two aspects. First, we establish stationary convergence of each RISE policy-optimization phase under a fixed old-policy snapshot. Then, we show that the weighted binary SRL objective used for the optimization of the *Verifier* inherits the same convergence property.

B.1 Regularity and Structural Properties

We analyze one policy-optimization phase with a fixed old-policy snapshot and define the equivalent minimization objective as $F(\theta) = -J_{\mathrm{RISE}}(\theta)$. Within this phase, the sampled rollouts and the rollout-derived quantities g^i, A_i^{r}, A_i^{a}, the correction index k, and the correction weight $\omega_{\mathcal{G}}$ are fixed with respect to the current optimization parameter $\theta \in \Theta$, where Θ denotes a closed convex parameter region.

Assumption 1 (Regular Policy Parameterization)**.** *All optimization iterates and their associated proximal points remain in Θ. For finite constants $B_{\rho,2}$, $B_{\ell,1}$, and $B_{\ell,2}$,*

$$\|\nabla_\theta^2 \rho_{i,t}(\theta)\| \leqslant B_{\rho,2}, \qquad \|\nabla_\theta \log \pi_\theta(u \mid c)\| \leqslant B_{\ell,1}, \qquad \|\nabla_\theta^2 \log \pi_\theta(u \mid c)\| \leqslant B_{\ell,2}. \tag{9}$$

Assumption 2 (Regular KL Term)**.** *There exists a finite constant $\mu_{\mathrm{KL}} \geqslant 0$ such that*

$$\nabla_\theta^2 D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}}) \succeq -\mu_{\mathrm{KL}} I. \tag{10}$$

Assumption 3 (Stochastic First-Order Oracle)**.** *Let $\mathcal{F}_j$ denote the optimization history before iteration j. The stochastic update direction $\widehat{g}_j$ satisfies*

$$\mathbb{E}[\widehat{g}_j \mid \mathcal{F}_j] \in \partial F(\theta_j), \qquad \mathbb{E}[\|\widehat{g}_j\|^2 \mid \mathcal{F}_j] \leqslant G_F^2 \tag{11}$$

for a finite constant G_F.

Lemma 1 (Bounded RISE Advantages)**.** *For $G \geqslant 2$, every per-token advantage entering the RISE PPO loss is bounded as*

$$|A_{i,t}| \leqslant C_A, \qquad C_A = 2\sqrt{G-1}. \tag{12}$$

Proof. Consider one segment type $j \in \{\mathrm{r}, \mathrm{a}\}$ and let $m \leqslant G$ be the number of valid rewards used for group normalization. For $m = 1$, the normalized reward is zero. For $m \geqslant 2$, we define $d_i = r_i^j - \mathrm{mean}(\mathbf{r}^j)$. Since $\sum_{i=1}^m d_i = 0$, from the Cauchy-Schwarz inequality, we have

$$d_i^2 = \left(\sum_{k \neq i} d_k\right)^2 \leqslant (m-1) \sum_{k \neq i} d_k^2. \tag{13}$$

If ζ denotes the standard deviation used for normalization, this implies $|d_i| \leqslant \sqrt{m-1}\,\zeta$ and hence $|\widetilde{r}_i^j| \leqslant \sqrt{G-1}$.

Each valid segment-wise advantage in Eq. (5) contains at most the sum of two normalized rewards, giving $|A_i^{\mathrm{r}}|, |A_i^{\mathrm{a}}| \leqslant 2\sqrt{G-1}$. For an invalid-format rollout, the implementation directly assigns the final advantage -0.1 without group normalization or cross-segment coupling. Since $0.1 < 2\sqrt{G-1}$ for $G \geqslant 2$, the same bound applies to all tokens. □

Lemma 2 (Weak Convexity and Lower Boundedness of Decision-Maker)**.** *Under Assumptions 1-2, the fixed-snapshot minimization objective $F = -J_{\mathrm{RISE}}$ is μ-weakly convex and bounded from below, where*

$$\mu = \mu_{\mathrm{clip}} + \lambda_{\mathrm{corr}} \mu_{\mathrm{corr}} + \beta \mu_{\mathrm{KL}}, \qquad \mu_{\mathrm{clip}} = C_A B_{\rho,2}, \qquad \mu_{\mathrm{corr}} = B_{\ell,1}^2 + 2B_{\ell,2}. \tag{14}$$

Proof. We first establish the weak convexity of the clipped PPO surrogate, adaptive supervision correction term, and KL regularizer, and then show that the resulting objective is bounded from below.

Clipped PPO surrogate. For a fixed advantage A, we define the negative token-level clipped surrogate as

$$\phi_A(\theta) = -\min\left(\rho(\theta)A, \mathrm{clip}(\rho(\theta), 1-\epsilon_c, 1+\epsilon_c)A\right). \tag{15}$$

For $A \geqslant 0$,

$$\phi_A(\theta) = \max\left\{-A\rho(\theta), -A(1+\epsilon_c)\right\}, \tag{16}$$

whereas for $A < 0$,

$$\phi_A(\theta) = \max\left\{|A|\rho(\theta), |A|(1-\epsilon_c)\right\}. \tag{17}$$

For every non-constant branch, Assumption 1 and Lemma 1 give

$$\left\|\nabla_\theta^2\left(\pm A\rho(\theta)\right)\right\| \leqslant |A|B_{\rho,2} \leqslant C_A B_{\rho,2} = \mu_{\mathrm{clip}}. \tag{18}$$

Thus the Hessian of each non-constant branch is bounded below by $-\mu_{\mathrm{clip}} I$, and each branch becomes convex after adding $\frac{\mu_{\mathrm{clip}}}{2}\|\theta\|^2$. Since the pointwise maximum of convex functions is convex, ϕ_A is μ_{clip}-weakly convex. Averaging over tokens and rollouts, together with the non-negative segment weights w_{r} and w_{a} satisfying $w_{\mathrm{r}} + w_{\mathrm{a}} = 1$, preserves the same weak-convexity bound.

Adaptive supervision correction. For an activated correction formulated as Eq. (7), we define the normalized log-likelihood margin as

$$\Delta_\theta = \bar{\ell}_\theta(a^* \mid x, z^k) - \bar{\ell}_\theta(\hat{a}^k \mid x, z^k). \tag{19}$$

Since $-\log\sigma(x) = \mathrm{softplus}(-x)$, the negative correction term can be written as

$$-\mathcal{C}_{\mathcal{G}}(\theta) = \omega_{\mathcal{G}} q(\Delta_\theta), \qquad q(x) = \mathrm{softplus}(-x). \tag{20}$$

The correction mechanism uses the action-matching reward $r_i^{\mathrm{a}} \in [0,1]$, and therefore

$$0 \leqslant \omega_{\mathcal{G}} = 1 - \max_i r_i^{\mathrm{a}} \leqslant 1. \tag{21}$$

For $q(x) = \mathrm{softplus}(-x)$, $|q'(x)| \leqslant 1$ and $0 \leqslant q''(x) \leqslant 1/4$. Since each length-normalized log-likelihood is an average of token-level log-probabilities, by Assumption 1 and the triangle inequality, we have

$$\|\nabla_\theta \Delta_\theta\| \leqslant 2B_{\ell,1}, \qquad \|\nabla_\theta^2 \Delta_\theta\| \leqslant 2B_{\ell,2}. \tag{22}$$

Therefore,

$$\begin{aligned}\left\|\nabla_\theta^2 q(\Delta_\theta)\right\| &\leqslant q''(\Delta_\theta)\|\nabla_\theta\Delta_\theta\|^2 + |q'(\Delta_\theta)|\|\nabla_\theta^2\Delta_\theta\| \\ &\leqslant \frac{1}{4}(2B_{\ell,1})^2 + 2B_{\ell,2} \\ &= B_{\ell,1}^2 + 2B_{\ell,2}.\end{aligned} \tag{23}$$

Since $0 \leqslant \omega_{\mathcal{G}} \leqslant 1$, multiplying by $\omega_{\mathcal{G}}$ does not increase this curvature bound. Thus, the negative correction term is μ_{corr}-weakly convex with $\mu_{\mathrm{corr}} = B_{\ell,1}^2 + 2B_{\ell,2}$.

Overall objective. In the minimization objective $F = -J_{\mathrm{RISE}}$ (i.e., Eq. (8)), the clipped surrogate contributes μ_{clip}, the correction term is scaled by λ_{corr}, and the KL term is scaled by β. By Assumption 2, the KL regularizer is μ_{KL}-weakly convex. Since non-negative weighted sums of weakly convex functions remain weakly convex, F is μ-weakly convex with

$$\mu = \mu_{\mathrm{clip}} + \lambda_{\mathrm{corr}}\mu_{\mathrm{corr}} + \beta\mu_{\mathrm{KL}}. \tag{24}$$

It remains to show that F is bounded. For $A \geqslant 0$, the clipped PPO contribution satisfies

$$\min(\rho A, \mathrm{clip}(\rho, 1-\epsilon_c, 1+\epsilon_c)A) \leqslant (1+\epsilon_c)C_A, \tag{25}$$

whereas for $A < 0$, the corresponding contribution is non-positive. In addition, $\mathcal{C}_{\mathcal{G}}(\theta) \leqslant 0$ because $\log\sigma(\cdot) \leqslant 0$, and $-D_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}}) \leqslant 0$ because KL divergence is non-negative. Hence, J_{RISE} admits a finite upper bound. □

B.2 Stationary Convergence of the Decision Maker

For $0 < \lambda < 1/\mu$, we define the Moreau envelope of F as

$$F_\lambda(\theta) = \min_{\vartheta} \left\{ F(\vartheta) + \frac{1}{2\lambda}\|\vartheta - \theta\|^2 \right\}. \tag{26}$$

Since F is μ-weakly convex, F_λ is continuously differentiable for $\lambda < 1/\mu$, and $\|\nabla F_\lambda(\theta)\|$ provides a first-order stationarity measure for F.

Theorem 1 (Stationarity of a Fixed-Snapshot RISE Optimization Phase)**.** *Suppose the policy is updated as $\theta_{s+1} = \theta_s - \eta\widehat{g}_s$, where $\widehat{g}_s$ satisfies Assumption 3. Let $R \sim \mathrm{Unif}\{0, \dots, K-1\}$ be the index of an iterate selected uniformly at random from the first K iterates. Then we have*

$$\mathbb{E}\left[\|\nabla F_\lambda(\theta_R)\|^2\right] \leqslant \frac{\Delta_F}{\eta(1-\lambda\mu)K} + \frac{\eta G_F^2}{2\lambda(1-\lambda\mu)}, \tag{27}$$

where $F_{\lambda,\inf} := \inf_\theta F_\lambda(\theta)$ and $\Delta_F = F_\lambda(\theta_0) - F_{\lambda,\inf}$. For fixed $0 < \lambda < 1/\mu$ and $\eta = \Theta(K^{-1/2})$, $\mathbb{E}\left[\|\nabla F_\lambda(\theta_R)\|^2\right]$ scales as $\mathcal{O}(K^{-1/2})$.

Lemma 3 (Subgradient Alignment)**.** *Let $\bar{g}_s = \mathbb{E}[\widehat{g}_s \mid \mathcal{F}_s] \in \partial F(\theta_s)$. Then*

$$\langle \bar{g}_s, \nabla F_\lambda(\theta_s) \rangle \geqslant (1-\lambda\mu)\|\nabla F_\lambda(\theta_s)\|^2. \tag{28}$$

Proof. Let $\widehat{\theta}_s = \mathrm{prox}_{\lambda F}(\theta_s)$. The Moreau-envelope identity gives $\nabla F_\lambda(\theta_s) = (\theta_s - \widehat{\theta}_s)/\lambda$, while proximal optimality gives $\nabla F_\lambda(\theta_s) \in \partial F(\widehat{\theta}_s)$. Since F is μ-weakly convex, its subdifferential is μ-hypomonotone, and hence

$$\left\langle \bar{g}_s - \nabla F_\lambda(\theta_s), \theta_s - \widehat{\theta}_s \right\rangle \geqslant -\mu\|\theta_s - \widehat{\theta}_s\|^2. \tag{29}$$

Substituting $\theta_s - \widehat{\theta}_s = \lambda\nabla F_\lambda(\theta_s)$ and dividing by λ yields Eq. (28). □

Proof of Theorem 1. Let $\widehat{\theta}_s = \mathrm{prox}_{\lambda F}(\theta_s)$. Evaluating the Moreau envelope at the previous proximal point and using $\theta_{s+1} = \theta_s - \eta\widehat{g}_s$ gives

$$F_\lambda(\theta_{s+1}) \leqslant F_\lambda(\theta_s) - \eta\langle \nabla F_\lambda(\theta_s), \widehat{g}_s \rangle + \frac{\eta^2}{2\lambda}\|\widehat{g}_s\|^2. \tag{30}$$

Taking conditional expectation and applying Assumption 3 and Lemma 3 yields

$$\mathbb{E}[F_\lambda(\theta_{s+1}) \mid \mathcal{F}_s] \leqslant F_\lambda(\theta_s) - \eta(1-\lambda\mu)\|\nabla F_\lambda(\theta_s)\|^2 + \frac{\eta^2 G_F^2}{2\lambda}. \tag{31}$$

Taking total expectation and summing over $s = 0, \dots, K-1$ gives

$$\eta(1-\lambda\mu)\sum_{s=0}^{K-1}\mathbb{E}\left[\|\nabla F_\lambda(\theta_s)\|^2\right] \leqslant \Delta_F + \frac{K\eta^2 G_F^2}{2\lambda}. \tag{32}$$

Dividing both sides by $\eta(1-\lambda\mu)K$ gives

$$\frac{1}{K}\sum_{s=0}^{K-1}\mathbb{E}\left[\|\nabla F_\lambda(\theta_s)\|^2\right] \leqslant \frac{\Delta_F}{\eta(1-\lambda\mu)K} + \frac{\eta G_F^2}{2\lambda(1-\lambda\mu)}. \tag{33}$$

Since $R \sim \mathrm{Unif}\{0, \dots, K-1\}$,

$$\mathbb{E}\left[\|\nabla F_\lambda(\theta_R)\|^2\right] = \frac{1}{K}\sum_{s=0}^{K-1}\mathbb{E}\left[\|\nabla F_\lambda(\theta_s)\|^2\right]. \tag{34}$$

Combining Eq. (33) and Eq. (34) yields Eq. (27). For fixed $0 < \lambda < 1/\mu$ and $\eta = \Theta(K^{-1/2})$, both terms on the right-hand side of Eq. (33) scales as $\mathcal{O}(K^{-1/2})$. Thus, $\mathbb{E}\left[\|\nabla F_\lambda(\theta_R)\|^2\right]$ also scales as $\mathcal{O}(K^{-1/2})$. □

B.3 Verifier Optimization

The *Verifier* uses the standard binary SRL objective with trajectory-weighted normalized advantages. Since $w_\tau = 1 - \alpha q_\tau$ with $q_\tau \in [0, 1]$ and $\alpha \in [0, 1)$, we have $0 < w_\tau \leqslant 1$. Thus trajectory weighting does not increase the magnitude of any normalized advantage.

Corollary 2 (Stationary Convergence of Verifier Optimization)**.** *Consider a fixed set of* Verifier*-training instances and let $F_v(\delta) = -J_v(\delta)$ denote the corresponding weighted binary SRL objective. Under the same regularity conditions as Theorem 1, there exists a finite weak-convexity constant μ_v such that, for $0 < \lambda < 1/\mu_v$,*

$$\mathbb{E}\left[\|\nabla(F_v)_\lambda(\delta_R)\|^2\right] \leqslant \frac{\Delta_v}{\eta_v(1-\lambda\mu_v)K} + \frac{\eta_v G_v^2}{2\lambda(1-\lambda\mu_v)}, \tag{35}$$

where $\Delta_v = (F_v)_\lambda(\delta_0) - (F_v)_{\lambda,\inf}$, and G_v^2 bounds the conditional second moment of the stochastic update direction for the Verifier*. For $\eta_v = \Theta(K^{-1/2})$,*

$$\mathbb{E}\left[\|\nabla(F_v)_\lambda(\delta_R)\|^2\right] = \mathcal{O}(K^{-1/2}). \tag{36}$$

Proof. For every verifier-training instance, $|w_{\tau(i)}A_i| \leqslant |A_i|$ because $0 < w_\tau \leqslant 1$. Thus trajectory weighting preserves the bounded-advantage property of the standard group-normalized SRL objective.

The Verifier objective contains only the clipped PPO surrogate and the KL regularizer. The same argument used in Lemma 2 therefore shows that F_v is μ_v-weakly convex for some finite μ_v and is bounded from below. Under the stochastic first-order oracle condition, the proof of Theorem 1 applies directly to F_v, yielding Eq. (35). Choosing $\eta_v = \Theta(K^{-1/2})$ gives Eq. (36). □

B.4 Empirical Training Dynamics

To complement the theoretical analysis, we further examine the standalone performance of the *Decision Maker* throughout the iterative RISE training process. Fig. 7 reports the ID and OOD task success rates of the *Decision Maker* checkpoint after each of training rounds, evaluated without *Verifier* or *Expert* assistance.

Although a slight performance drop is observed in the initial rounds, likely due to the variability of the collected training trajectories, the performance subsequently improves on both ID and OOD tasks and gradually stabilizes. These empirical observations are consistent with the phase-wise stationary-convergence analysis above and provide empirical support for the stability of the RISE optimization process. Specifically, the sustained improvement and eventual stabilization of the standalone *Decision Maker* across successive training rounds indicate that the optimization in each round can effectively leverage the newly collected trajectories, leading to progressive policy improvement over the iterative training process.

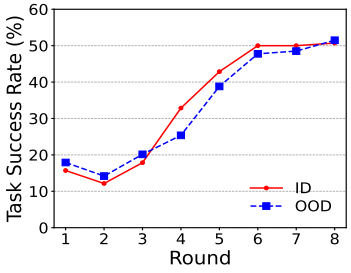

Figure 7: Round-wise standalone performance of the *Decision Maker* during RISE training.

C Algorithmic Details

C.1 Trajectory-Level Transition Memory

To improve the efficiency of verification and expert decision making, we design a trajectory-level transition memory inspired by (Shinn et al., 2023; Liu et al., 2024). This mechanism maintains a trajectory-scoped transition memory and provides the *Verifier* and *Expert* with additional information from prior interactions, enabling them to leverage relevant evidence during verification and decision making. Specifically, we maintain a persistent memory of observed transitions induced by actions that have been approved and executed during the interaction. Given a candidate action, we retrieve the historical transitions associated with it, rank them by their contextual relevance to the current state, and feed back the most relevant ones to the *Verifier* π_δ and *Expert* π_e as auxiliary evidence for assessing the candidate action and making expert decisions, respectively. In our implementation, the retrieved transitions are inserted into the `{action_memory}` field of the *Verifier* and *Expert* prompts provided in Appendix C.2.

C.2 Prompt Templates

Prompt for Decision Maker

```
You are an expert agent operating in the ALFRED Embodied Environment. Your task
is to: {task_description}
Prior to this step, you have already taken {step_count} step(s). Below are the
most recent {history_length} observations and the corresponding actions you took:
{action_history}
You are now at step {current_step} and your current observation is:
{current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].
Now it's your turn to take an action. You should first reason step-by-step about
the current situation. This reasoning process MUST be enclosed within <think>
</think> tags.
Once you've finished your reasoning, you should choose an admissible action for
current step and present it within <action> </action> tags.
```

Prompt for Verifier

```
You are an expert evaluator responsible for judging actions in the ALFRED Embodied
Environment. The given task is: {task_description}
Prior to the action being evaluated, the acting agent has completed {step_count}
interaction step(s). Below is its recent interaction history, including the most
recent {history_length} observations and corresponding actions: {action_history}
You are now judging one executed action at step {current_step}.
Previous observation before the action: {previous_observation}
Executed action: {action}
Next observation after executing the action: {next_observation}
About this action you already knows that: {action_memory}
Your task is to judge whether the executed action is a good action for completing
the given ALFRED task.
Good Action: Advances the task by progressing toward the goal, obtaining/moving
to needed objects, performing relevant actions, revealing useful information, or
completing prerequisites.
Bad Action: Hinders the task by moving away from the goal, repeating useless
actions, interacting with irrelevant objects, causing no state change, undoing
progress, or wasting steps.
You should first reason briefly about the previous observation, the executed action,
and the next observation. This reasoning process MUST be enclosed within <think>
</think> tags.
After reasoning, output your final judgment within <judge> </judge> tags. The
judgment MUST be exactly one of:
- yes
- no
Output EXACTLY TWO blocks and no other text:
<think>brief reasoning about whether the action helps complete the task</think>
<judge>yes or no</judge>
```

Prompt for Expert

```
You are an expert agent operating in the ALFRED Embodied Environment. Your task
is to: {task_description}
Prior to this step, you have already taken {step_count} step(s). Below are the
most recent {history_length} observations and the corresponding actions you took:
{action_history}
You are now at step {current_step} and your current observation is:
{current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].
Here are some additional information: {action_memory}
```

Prompt for Expert

```
Now it's your turn to take an action.
You should first reason step-by-step about the current situation. This reasoning
process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for
current step and present it within <action> </action> tags.
```

C.3 Algorithm Pseudocode

Algorithm 1: Workflow of RISE

Require: Decision Maker π_θ, Verifier π_δ, Expert π_e, task set T, number of training rounds R

1 **for** $r = 1, \ldots, R$ **do**
2 $\mathcal{D}_\theta, \mathcal{D}_\delta \leftarrow \varnothing$;
3 **Phase I: Selective Expert-Guided Interaction**;
4 **foreach** *task* $\xi \in T$ **do**
5 Initialize state s_0 and transition memory $\mathcal{M}$;
6 **foreach** *interaction step* t **do**
7 π_θ generates candidate action a_t^d and tentatively executes it to obtain s_{t+1}^d;
8 Retrieve relevant transitions $\mathcal{R}_t$ from $\mathcal{M}$ and obtain $y_t \in \{\texttt{yes}, \texttt{no}\}$ from π_δ;
9 **if** $y_t = \texttt{yes}$ **then**
10 Retain (s_t, a_t^d, s_{t+1}^d);
11 **else**
12 Roll back to s_t, query π_e for a_t^e, and execute a_t^e;
13 Update $\mathcal{M}$ with the committed transition;
14 **if** *the collaborative trajectory* τ *succeeds* **then**
15 Decompose τ into policy-training instances and add them to $\mathcal{D}_\theta$;
16 Add the corresponding verifier-training instances and decisions to $\mathcal{D}_\delta$;
17 **Phase II: Policy and Verifier Optimization**;
18 *Optimization of the Decision Maker*;
19 **foreach** *policy-training instance* $(x, a^*) \in \mathcal{D}_\theta$ **do**
20 Sample G rollouts $\{o^i = (z^i, \hat{a}^i)\}_{i=1}^G$ from $\pi_{\theta_{\text{old}}}(\cdot \mid x)$;
21 Compute the reasoning-induced information gain g^i for each z^i according to Eq. (4);
22 Obtain reasoning rewards $r_i^{\text{r}} = \max(0, g^i)$ and action rewards r_i^{a} according to Eq. (2);
23 Normalize $\{r_i^{\text{r}}\}_{i=1}^G$ and $\{r_i^{\text{a}}\}_{i=1}^G$ separately within the sampled group;
24 Compute the reasoning and action advantages A_i^{r} and A_i^{a} according to Eq. (5);
25 Assign A_i^{r} and A_i^{a} to the reasoning and action tokens, respectively, and compute the segment-wise surrogates by Eq. (6);
26 $k \leftarrow \arg\max_i g^i$;
27 **if** $\max_i r_i^{\text{a}} < 1$ *and* $g^k > 0$ **then**
28 Construct the preference $a^* \succ \hat{a}^k \mid (x, z^k)$ and compute adaptive supervision correction by Eq. (7);
29 Update π_θ by maximizing the RISE objective in Eq. (8);
30 *Optimization of the Verifier*;
31 Balance the yes and no instances in $\mathcal{D}_\delta$;
32 **foreach** *successful trajectory* τ *in* $\mathcal{D}_\delta$ **do**
33 Compute its rejection ratio q_τ and trajectory weight $w_\tau = 1 - \alpha q_\tau$;
34 **foreach** *verifier-training instance* $i \in \mathcal{D}_\delta$ **do**
35 Apply standard SRL to obtain the group-relative advantage A_i;
36 Reweight the advantage as $\widetilde{A}_i = w_{\tau(i)} A_i$;
37 Update π_δ using the trajectory-weighted standard SRL objective;

The detailed workflow of RISE is summarized in Algorithm 1. At Lines 1–13, RISE performs selective expert-guided interaction, where the *Decision Maker*, *Verifier*, and *Expert* collaboratively generate

successful trajectories. At Lines 14–16, each successful collaborative trajectory is decomposed into step-wise policy-training instances, while the verifier interaction records collected during its generation are retained as the corresponding verifier-training instances. At Lines 18–24, the *Decision Maker* is optimized with segment-wise reasoning and action advantages derived from the reasoning-induced information gain rewards and action rewards, respectively. At Lines 25–29, adaptive supervision correction is applied when none of the sampled rollouts exactly matches the supervision action, providing an additional learning signal when rollout-based supervision is insufficient. Finally, at Lines 31–37, the *Verifier* is optimized with trajectory-weighted standard SRL based on the rejection ratios of successful trajectories.

D Implementation Details

Table 3: Training hyperparameters of RISE.

Training Hyperparameter	Value	Training Hyperparameter	Value
Optimizer	AdamW	Training Rounds	8
Epochs per Round	1	Source Batch Size	4
PPO Mini-Batch Size	32	Group Size	8
Learning Rate	1×10^{-5}	PPO Clip Ratio	0.2
Entropy Coefficient	0.001	KL Loss Coefficient	0.01
Gradient Clipping	1.0	Reasoning / Action Reward Weights	0.5 / 0.5
Expert Correction Coefficient	0.25	Invalid-Format Penalty	-0.1
Maximum Environment Steps	30	Interaction History Length	2
LoRA Rank	32	LoRA Alpha	64

We implement all methods in PyTorch, with the agent-environment interaction and RL pipeline built on top of `verl-agent`. All experiments are conducted on a server equipped with two NVIDIA A100 PCIe GPUs with 80 GB memory each. The detailed hyperparameter settings of RISE are provided in Table 3. All trainable methods use the same base model and LoRA configuration as RISE. It should be noted that although there is no universally agreed parameter threshold for SLMs, recent studies have characterized models with 100M-5B or up to 3B parameters as SLMs (Lu et al., 2025; Luo et al., 2025). Following these conventions, we deliberately adopt a 1.5B model as the Base SLM to evaluate RISE in a more capacity-constrained setting. In addition, for methods that require prompting, we consistently use the prompt template of the *Decision Maker* provided in Appendix C.2.

In Section 4.2, for the GRPO and GiGPO baselines, we train the models for 150 epochs with a learning rate of 5×10^{-6}, a PPO mini-batch size of 64, and a KL-loss coefficient of 0.001. For GiGPO, we additionally set the discount factor γ to 0.95 and the step-level advantage weight to 1.0. For SFT and SRL, training trajectories are collected entirely by the stronger expert, i.e., the Expert LLM used in our experiments, under the same expert-usage budget as RISE. To ensure a consistent collection setting, we use the same expert API endpoint and number of parallel processes as RISE. For SFT+GRPO, we adopt the same SFT and GRPO configurations described above, except that the subsequent GRPO stage is trained for 100 epochs.

For STEER, the token-level logits required by its original implementation are unavailable through the expert LLM API. We therefore adapt STEER to our API-based setting while preserving its original SLM-side confidence computation and confidence-triggered expert intervention mechanism. Specifically, the SLM queries the expert LLM only when its confidence falls below a predefined threshold, which is set to 0.5 in our experiments.

Additional experimental results are provided in Appendix E. Unless otherwise specified, these experiments follow the same settings as the main evaluation. In Appendix E.2, we replace *GLM-5* in non-thinking mode with *Kimi-2.6* in non-thinking mode as the expert LLM. In Appendix E.3, we replace ALFWorld with the BabyAI (Chevalier-Boisvert et al., 2019) environment and follow AgentBoard (Ma et al., 2024) for the environment configuration and test-set split.

E Additional Experimental Results

E.1 Validation of Reasoning-Induced Information Gain

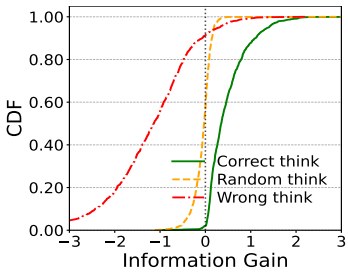

Figure 8: CDF of reasoning information gain for various reasoning.

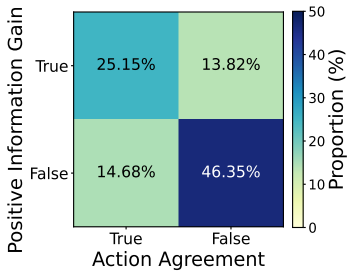

Figure 9: Reasoning and action quality distribution of rollouts during SRL training.

To evaluate the effectiveness of this metric, we construct three types of reasoning content based on expert trajectories: correct reasoning, which corresponds to the original reasoning associated with the expert action; incorrect reasoning, which is extracted from other rollouts; and random reasoning, which consists of randomly generated tokens. We then calculate their reasoning-induced information gains. Fig. 8 presents the cumulative distribution functions of the three groups. The correct reasoning traces generally achieve higher information gains, with most values being positive. In contrast, incorrect reasoning traces tend to produce negative information gains, while randomly generated reasoning falls between the two. These observations indicate that reasoning information gain can effectively distinguish reasoning traces according to their contribution to correct action prediction.

We further apply this metric to the rollouts obtained through group sampling by the same base SLM on expert trajectories, with the results shown in Fig. 9. The figure reports the proportions of samples under different combinations of reasoning-induced information gain and action agreement with the expert. We observe that a non-negligible fraction of samples contain reasoning that contributes positively to predicting the correct action but still produces an incorrect final action. Conversely, some samples match the expert action even though their reasoning provides little or no positive contribution to that prediction. These results are consistent with our earlier observations and further demonstrate that action-level rewards alone cannot adequately reflect the quality of the underlying reasoning.

E.2 Generalization across Expert LLMs

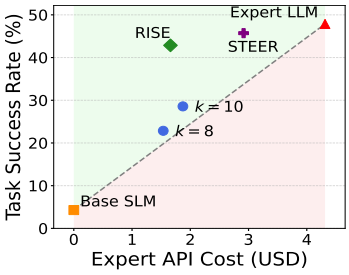

Figure 10: Performance-cost trade-off in ID environments (with Expert *Kimi-2.6*).

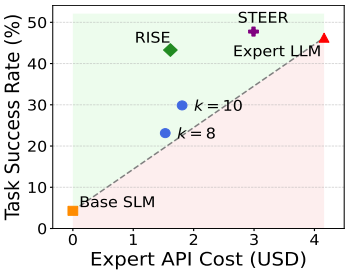

Figure 11: Performance-cost trade-off in OOD environments (with Expert *Kimi-2.6*).

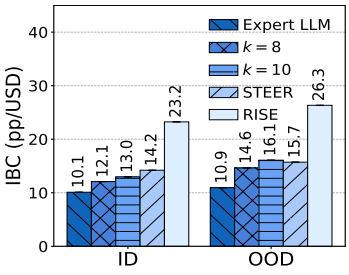

Figure 12: IBC results in ID and OOD environments (with Expert *Kimi-2.6*).

To investigate the impact of the expert model on RISE, we replace the original expert LLM with *Kimi-2.6* in non-thinking mode, retrain RISE under this setting, and compare it with the other expert-collaborative methods. Fig. 10 and Fig. 11 illustrate the corresponding performance-cost trade-offs on ID and OOD tasks, respectively. Compared with the *GLM-5* setting in Section 4.2, however, all expert-collaborative methods incur noticeably higher expert API costs while generally achieving lower task success rates. As shown in the figures, RISE continues to achieve a favorable trade-off

between task performance and expert API cost on both ID and OOD tasks. We also observe that STEER and the k-step random selection strategies with $k = 8$ and $k = 10$ lie above the reference line under this setting, indicating that these methods can also utilize the more expensive expert model more efficiently than direct expert invocation. Nevertheless, RISE remains clearly separated from these baselines, achieving substantially higher success rates at comparable or lower expert API costs.

Fig. 12 further compares the IBC values of different methods on ID and OOD tasks. Although STEER, $k = 8$, and $k = 10$ all achieve higher IBC values than directly using the Expert LLM, RISE consistently provides the largest improvement, reaching IBC values of 23.2 and 26.3 on ID and OOD tasks, respectively. These results indicate that, even when expert assistance becomes more costly and provides a lower absolute task performance than in the *GLM-5* setting, RISE can still make more effective use of expert calls and maintain a favorable performance-cost trade-off.

Overall, this experiment provides additional evidence that RISE is effective across expert LLMs with different performance and pricing characteristics. At the same time, it also shows that the absolute task performance and inference cost of RISE are naturally affected by the capability and API cost of the underlying expert model, since both the supervision provided by the expert and the cost of expert intervention directly depend on the selected expert.

E.3 Generalization across Interactive Environments

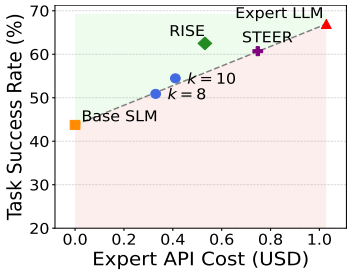

Figure 13: Performance-cost trade-off in BabyAI test tasks.

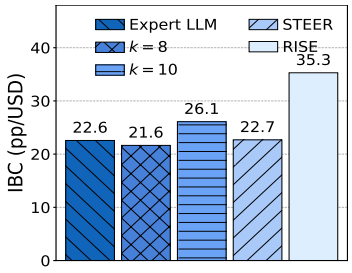

Figure 14: IBC results in BabyAI test tasks.

To examine the generalization of RISE across interactive environments, we further evaluate it in the BabyAI (Chevalier-Boisvert et al., 2019) environment and compare it with the same expert-collaborative baselines. Unless otherwise specified, we keep the experimental configurations consistent with the main evaluation. Fig. 13 and Fig. 14 illustrate the corresponding performance-cost trade-offs on the BabyAI test tasks.

Compared with the ALFWorld results in Section 4.2, the Base SLM already exhibits non-trivial decision-making capability on BabyAI, suggesting that these tasks are relatively easier for the underlying model. Consequently, all expert-collaborative methods achieve relatively high success rates, with their performance lying between the Base SLM and Expert LLM reference levels. Despite the reduced room for improvement in this setting, RISE still achieves higher task success rates at comparable or lower expert API costs than the competing expert-collaborative methods.

Fig. 12 further compares the IBC values of the different methods on the BabyAI test tasks. While STEER and the k-step random-selection baselines achieve IBC values broadly comparable to direct use of the Expert LLM, RISE maintains a substantially higher IBC of 35.3, exceeding that of the Expert LLM by 12.7. This indicates that RISE continues to translate expert API expenditure into task-performance improvement more efficiently in a different interactive environment.

Overall, these results provide additional evidence for the applicability of RISE across different interactive environments. They also highlight that the absolute task performance and expert API cost are naturally environment-dependent: when the underlying tasks are easier and the Base SLM is already more capable, the potential performance gain from additional expert intervention becomes smaller. Nevertheless, RISE consistently preserves its advantage in expert-cost efficiency.

F LIMITATIONS

RISE reduces expert intervention but does not eliminate its dependence on the underlying expert model. The quality of expert-provided supervision is naturally affected by the expert's capability, while its invocation cost influences the resulting performance-cost trade-off. Future work could investigate more adaptive and multi-expert collaboration strategies to better balance supervision quality and expert cost, and further extend RISE to more challenging chain-of-thought reasoning tasks.