Counterfactual Distillation: Internalizing Reflective Experience into LLM Agent

Rui Ge[1*] **Yichao Fu**[1] **Yu-Yang Qian**[2] **Junda Su**[1] **Yiming Zhao**[1] **Hao Zhang**[1†]
[1]University of California San Diego [2]Nanjing University

Abstract

Large language models are increasingly deployed as autonomous agents that plan and act over long-horizon interactions, learning from both final outcomes and rich environmental feedback. However, optimizing from such interaction experience remains challenging, as feedback is heterogeneous and rarely carries explicit reward information. Existing post-training approaches rely on outcome-driven methods, such as reinforcement learning with verifiable rewards (RLVR), which primarily exploit final success signals while leaving interaction experience and feedback underutilized. In sparse-reward, long-horizon tasks, this often results in *distribution sharpening*: the policy reinforces a narrow set of already-successful behaviors, without substantially improving the feedback-grounded agency needed for broader problem-solving capacity(e.g., Pass@k). We propose *Counterfactual Distillation*, a framework that converts exploration-derived experience into trainable policy supervision. Specifically, we organize exploration as a tree-structured search process, where the agent reflects on past decision points and makes new experience-guided decisions to form alternative branches. These corrections are then distilled as counterfactual targets: we train the model to predict the revised action from the original history without explicitly providing the experience, thereby introducing out-of-distribution behavior patterns and expanding exploration capacity. Across diverse interactive coding and agentic tasks, our method outperforms outcome-driven baselines such as GRPO and experience-based methods such as Early Experience, with gains of up to 14 percentage points over the base model on Pass@128. When interleaved with reinforcement learning updates, it further raises the performance ceiling, improving Pass@1 on Sokoban by 8 percentage points over extended GRPO. Our code is available at `https://github.com/motigrez/Counterfactual-Distillation`.

1 Introduction

Large Language Models (LLMs) are increasingly deployed as autonomous agents that plan, act, and adapt over *long-horizon, multi-turn* interactions with complex environments (Yao et al., 2022b; Nakano et al., 2021). In such settings, success depends less on raw single-shot reasoning and more on the agent's ability to understand and adapt to the environment, since many failures arise from unfamiliarity with the rules or recurring pitfalls of the environment (Yang et al., 2024; Shinn et al., 2023). This makes *learning from experience* a central capability for long-horizon agents, allowing them to convert lessons from past failures into actionable knowledge for future attempts. A growing body of work exploits this idea at test time: agents accumulate insights into evolving playbooks or external knowledge bases for in-context reuse (Zhang et al., 2025b), or perform inference-time rollback and search to repair failing trajectories (Li et al., 2025b). While effective, these approaches

*Work done during an internship at the University of California San Diego.
†Corresponding Author: Hao Zhang <haozhang@ucsd.edu>.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

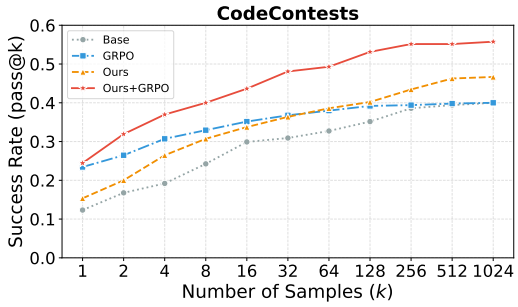


Figure 1: Internalizing feedback-grounded agency improves model capability (i.e., Pass@K) in long-horizon interaction, while outcome-only training (e.g., GRPO) yields limited gains beyond the base model.

rely on substantial test-time compute or external memory infrastructure, which is often impractical under latency- or cost-constrained deployment, where each query may afford only a single rollout.

A widely adopted alternative is reinforcement learning with verifiable rewards (RLVR) (Guo et al., 2025; Zeng et al., 2025b; Shao et al., 2024), which directly optimizes the policy from a terminal outcome reward and reduces reliance on test-time scaling. Despite its appealing simplicity, RLVR has several notable limitations in long-horizon agentic settings. First, it collapses the rich, structured environmental feedback (e.g., compiler errors, invalid actions, intermediate observations) into a single scalar at the episode level, leaving most interaction information unused. Second, rollouts are treated as independent samples scored only by their terminal reward, with no mechanism for accumulating or transferring experience across attempts. Third, sparse rewards make credit assignment difficult in long-horizon tasks: without reliable step-level signals, exploratory deviations cannot be properly attributed or reinforced, and optimization is biased toward behaviors already supported by the model. As a consequence, RLVR tends to behave like *distribution sharpening* (Yue et al., 2025; Zhao et al., 2025b): it concentrates probability mass on already-successful behaviors latent in the base model, boosting Pass@1 while yielding limited or even negative gains at large k (e.g., Pass@1024). This limitation is most visible under Pass@k evaluation: Pass@1 reflects single-rollout effectiveness, whereas Pass@k at large k approximates the model's true capability ceiling by exposing whether successful strategies exist anywhere in its distribution. As previewed in Figure 1, our method substantially expands this ceiling in long-horizon interaction, while RLVR offers little improvement over the base model.

These limitations point to an underused source of supervision in agentic training. Environmental feedback accumulated over long-horizon interaction contains far richer information than a terminal scalar reward. The key challenge is how to internalize this experience during training, as such experience is often heterogeneous, free-form, and structurally misaligned with the policy's action space: agentic tasks ultimately require grounded *think-then-act* interactions with the environment, rather than natural language commentary. Consequently, naively conditioning on or imitating natural-language experience does not by itself enrich the policy's intrinsic action distribution. Our goal is therefore to convert exploration-derived experience into *trainable policy supervision* that can be internalized into model parameters, thereby expanding the policy's exploration space and raising its capability ceiling without requiring explicit experience or heavy test-time scaling at inference.

To this end, we propose **Counterfactual Distillation**, a framework that converts reflective experience into trainable policy supervision through a two-stage pipeline. In Stage 1 (§3.1, *Tree-Structured Exploration for Experience Generation*), the agent reflects on its trajectory, localizes a suboptimal step τ, and synthesizes a brief natural-language experience e that diagnoses the failure and proposes a fix; after restoring the environment back to τ, it re-decides under the experience-augmented context, producing a revised action a'_τ. By consuming the structured feedback in place and propagating diagnoses across attempts through a search tree, this stage turns the rich interaction signal into experience that accumulates across rollouts. In Stage 2 (§3.2, *Experience Distillation*), we strip e from the context and train the model to reproduce a'_τ from the original history h_τ alone. The objective is *counterfactual* in the sense that it uses the experience-guided revision as supervision for the original decision context, training the model to behave as though the information in e had

already been internalized. Each training pair therefore transfers the behavioral change induced by e back to the original decision context, increasing the probability of feedback-grounded alternatives that were less likely under the original policy. In this way, Counterfactual Distillation converts an explicit in-context intervention into an implicit shift of the policy distribution, yielding localized decision-level supervision and expanding the behavioral modes accessible at inference.

We evaluate our method across a diverse suite of long-horizon agentic benchmarks, including CodeContests (Li et al., 2022), WebShop (Yao et al., 2022a), ALFWorld (Shridhar et al., 2020), ScienceWorld (Wang et al., 2022), and Sokoban (Schrader, 2018). Across tasks and model scales, our method consistently outperforms outcome-driven RLVR baselines such as GRPO (Guo et al., 2025) as well as experience-based methods such as Early Experience(Zhang et al., 2025a), with gains of up to 13.94 percentage points over the base model on Pass@128(Tables 1 and 2). Moreover, in the on-policy interleaving experiment (Table 3), Counterfactual Distillation further raises the *optimization ceiling* of outcome-driven RL: after additional GRPO updates have largely saturated, interleaving our distillation cycles continues to improve the performance, improving Pass@1 on Sokoban by 8.00 percentage points over extended GRPO. Together, these results indicate that counterfactually distilling exploration-derived experience provides a complementary axis to scalar-reward optimization: it injects structured, behaviorally diverse supervision that pure RLVR cannot easily access.

To summarize, our contributions can be listed as follows:

- We propose **Counterfactual Distillation**, a framework for internalizing reflective experience into the policy, enabling agents to benefit from experience without heavy test-time scaling at inference.
- We provide a complementary learning signal to outcome-driven RLVR, mitigating its reliance on terminal rewards by reusing rich interaction feedback, transferring experience across attempts, and supplying localized decision-level supervision.
- We conduct extensive experiments across diverse long-horizon agentic benchmarks, showing consistent improvements over RLVR and experience-based baselines, especially in large-k performance, and further gains when interleaved with on-policy GRPO training.

2 Background

2.1 LLMs as Interacting Agents

Following ReAct (Yao et al., 2022b), we model an interactive episode as an interleaved sequence of observations and actions. Given an instruction $q \in \mathcal{Q}$, at step t the environment returns a textual observation o_t, and the LLM policy π_θ generates an executable action

$$a_t \sim \pi_\theta(\cdot \mid h_t, q), \qquad h_t = (o_0, a_0, \ldots, o_{t-1}, a_{t-1}, o_t). \tag{1}$$

A rollout is denoted by $\tau = (q, o_0, a_0, \ldots, o_T)$, whose outcome is evaluated by a task-specific verifier or environment as a success/failure signal.

2.2 Reinforcement Learning with Verifiable Rewards

A common post-training paradigm for agentic LLMs is RL with verifiable rewards (RLVR), where a verifier or environment assigns a scalar reward to a completed trajectory, e.g., success/failure or the number of passing tests. Representative algorithms include PPO (Schulman et al., 2017) and group-based methods such as GRPO (Guo et al., 2025; Zeng et al., 2025b; Shao et al., 2024).

Given G sampled traces for the same problem, GRPO computes normalized advantages within the group and optimizes

$$\ell_i(\theta) = \min\Big(r_i(\theta)A_i, \mathrm{clip}(r_i(\theta), 1-\epsilon, 1+\epsilon)A_i\Big) - \beta\,\mathrm{KL},$$

$$J_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{q,\{o_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\ell_i(\theta)\right],$$

where $\mathrm{KL} \triangleq D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})$. Although effective, outcome-based RLVR provides only trajectory-level supervision, making credit assignment difficult in long-horizon generation and tool use. Since

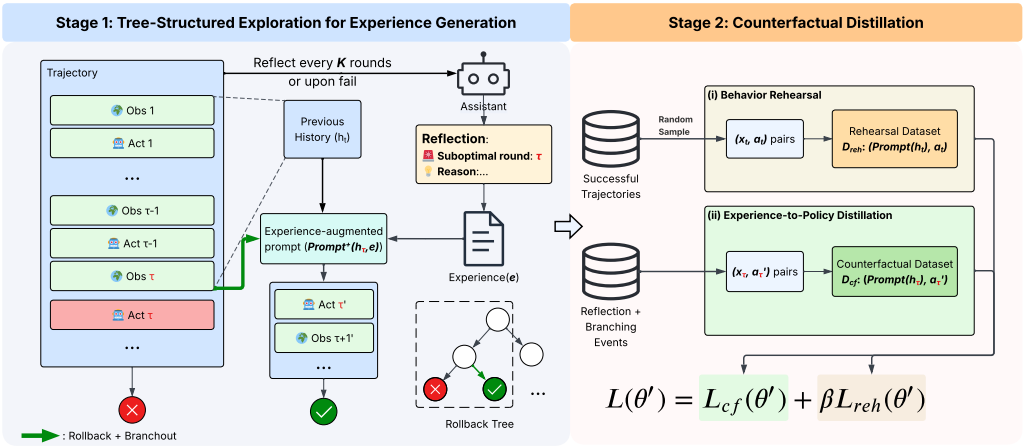


Figure 2: Illustration of the Counterfactual Distillation framework. **Stage 1**: During experience collection, the assistant periodically reviews the current trajectory and identifies a suboptimal round (denoted as red-colored τ). It then produces the actionable experience e, which is concatenated with the restored history to facilitate subsequent attempts. **Stage 2**: During experience distillation, the model optimizes a joint loss using two datasets: randomly sampled rehearsal pairs to maintain capabilities, and counterfactual pairs (original prompts paired with experience-improved actions) to internalize diverse exploration. For simplicity, we depict one branching event from the rollback exploration tree.

updates are driven by rewards on sampled rollouts, they often reweight or sharpen behaviors within the model's existing solution space, rather than reliably discovering qualitatively new solutions. This motivates evaluating capability under larger sampling budgets.

2.3 LLM's Capability Boundary

To quantify a model's capability boundary, we evaluate Pass@K following prior work (Yue et al., 2025). For each problem, we sample K rollouts and assign score 1 if any rollout succeeds under the task-specific verifier/environment, and 0 otherwise. Pass@K is the average score over the dataset.

Pass@1 measures single-rollout effectiveness, which is most relevant for deployment, while large-K Pass@K estimates the model's best-of-K potential, i.e., whether successful strategies exist within its rollout distribution. In our experiments, we use Pass@128 to characterize the empirical capability boundary and additionally report Pass@1 for single-attempt performance.

3 Counterfactual Distillation: Learning From Reflective Experience

This section introduces Counterfactual Distillation, a two-stage framework that converts rollout-time experience into counterfactual supervision for policy learning (see Figure 2). Stage 1 is the rollout stage (§3.1), where the agent interacts with the environment, reflects on trajectories, and performs tree-structured search to collect reflective experience and experience-guided revision actions. The pseudo-code is shown in Algorithm 1. Stage 2 is the training stage (§3.2), where we remove the explicit experience and train the model to reproduce the revised actions from the original decision contexts, thereby internalizing the experience-induced policy shift into model parameters.

3.1 Tree-Structured Exploration for Experience Generation

Setup and Notation. We consider an episodic interactive environment $\mathcal{E}$ with horizon T. For an instruction $q \in \mathcal{Q}$, an episode is a sequence of states $\{E_t\}_{t=0}^{T}$; at each step, the agent observes o_t and emits an executable action a_t. A language-model policy π_θ induces a distribution over actions conditioned on history as in Eq. 1, with environment dynamics $E_{t+1}, o_{t+1} \leftarrow \mathrm{Step}(E_t, a_t)$. A *rollout* executes the policy from an initial state until termination (success, irreversible failure, or step limit).

Reflection and Experience Generation. A fundamental challenge in leveraging experience is that linguistic feedback is qualitative and unstructured, and thus difficult to internalize directly through standard optimization. We address this by treating experience as a *contextual intervention* that induces a *policy shift* via the input context. Concretely, every K steps or upon failure, the agent invokes a reflection procedure. Given the interaction history h_t and a reflection prompt p_{refl}, the policy π_θ produces a target step τ and a natural-language experience e:

$$(\tau, e) \sim \pi_\theta(\cdot \mid h_t, p_{\text{refl}}), \quad \tau \in \{1, \ldots, t\},$$

where τ localizes the step at which the trajectory deviated from the desired path, and e provides a diagnostic with actionable suggestions for re-deciding at τ.

Branching via Replay. To explore an alternative branch from (τ, e), we restore the environment state E_τ and the corresponding history h_τ by resetting $\mathcal{E}$ and replaying the original action sequence $a_{1:\tau-1}$. Conditioned on the synthesized experience e, the policy then produces a revised action and continues the rollout:

$$a'_\tau \sim \pi_\theta(\cdot \mid h_\tau, q, e), \qquad E'_{\tau+1}, o'_{\tau+1} \leftarrow \text{Step}(E_\tau, a'_\tau).$$

The agent proceeds from $E'_{\tau+1}$ until the next reflection or termination. We organize branching requests in a queue and expand them in a breadth-first manner, yielding an implicit *experience tree* whose nodes are decision points and whose edges are experience-guided revisions. Expansion stops when reaching a maximum depth or exhausting the attempt budget.

Figure 5 provides a concrete example from Sokoban illustrating reflection, state restoration via replay, and branch exploration in practice.

3.2 Counterfactual Distillation

Stage 1 yields, for each branching event, a triple (h_τ, e, a'_τ): an original history h_τ, a natural-language experience e, and an experience-guided action a'_τ that is *counterfactual* to the action originally taken at step τ. Stage 2 internalizes these counterfactual revisions into the policy parameters.

(i) Counterfactual Objective. The core supervision is derived from the branching events identified during reflection. Whereas a'_τ was generated under the experience-augmented context (h_τ, q, e), we ask the policy to reproduce a'_τ from the experience-*free* context (h_τ, q) alone:

$$\mathcal{L}_{\text{cf}}(\theta') = -\mathbb{E}_{(h_\tau, a'_\tau) \sim \mathcal{D}_{\text{cf}}} \left[\log \pi_{\theta'}(a'_\tau \mid h_\tau, q) \right].$$

This objective is counterfactual in two complementary senses. At the *decision level*, it commits the policy to a'_τ at h_τ as though the lesson encoded in e were already part of its own knowledge, removing the need for explicit reflection at inference. At the *distribution level*, the in-context experience acts as a lever that lifts a previously low-probability action; training on (h_τ, a'_τ) alone absorbs this lift into the policy parameters as a persistent shift in support. As illustrated in Figure 6, when the trained model later encounters the same situation, it avoids repeating the original mistake without receiving any explicit experience.

(ii) Behavior Rehearsal. To preserve the agent's foundational competence and mitigate catastrophic forgetting under the counterfactual update, we additionally rehearse on successful trajectories. Following rejection-sampling-style imitation (Ahn et al., 2024), we treat successful rollouts—including those discovered through branching—as high-quality demonstrations. For each successful trajectory, we extract state-action pairs (h_t, a_t) and minimize:

$$\mathcal{L}_{\text{reh}}(\theta') = -\mathbb{E}_{(h,a) \sim \mathcal{D}_{\text{reh}}} \left[\log \pi_{\theta'}(a \mid h, q) \right].$$

Training Objective. The two losses are jointly optimized:

$$\mathcal{L}(\theta') = \mathcal{L}_{\text{cf}}(\theta') + \beta \, \mathcal{L}_{\text{reh}}(\theta'), \tag{2}$$

where β balances rehearsal strength. $\mathcal{L}_{\text{cf}}$ injects experience-induced alternatives into the policy's intrinsic action distribution—expanding behavioral coverage and improving large-k metrics such as pass@k—while $\mathcal{L}_{\text{reh}}$ retains the model's existing task competence.

3.3 On-Policy Counterfactual Distillation.

Our method naturally splits into a *rollout* phase (Stage 1) that collects experience under the current policy and an *update* phase (Stage 2) that trains on it, mirroring the structure of standard RL. We exploit this parallel and propose an on-policy variant of Counterfactual Distillation that composes cleanly with policy-gradient methods.

In each on-policy cycle, the rollout phase uses the latest policy π_θ to grow a fresh experience tree, recording both branching events (h_τ, a'_τ) and the reflection acts (h_t, τ, e) that produced them. The update phase then reinforces two kinds of supervision: (a) the revised action a'_τ via $\mathcal{L}_{\text{cf}}$, as before, and (b) reflection acts whose induced branches eventually succeed—i.e., reflections that proved productive in hindsight. Letting $\mathcal{D}^+_{\text{refl}}$ denote such successful reflections, we add:

$$\mathcal{L}_{\text{refl}}(\theta') = -\mathbb{E}_{(h_t,\tau,e)\sim\mathcal{D}^+_{\text{refl}}}[\log \pi_{\theta'}(\tau, e \mid h_t, p_{\text{refl}})]\,,$$

so that the policy learns not only *how to revise* but also *when and how to reflect*. The full on-policy objective is $\mathcal{L}_{\text{cf}} + \beta\,\mathcal{L}_{\text{reh}} + \gamma\,\mathcal{L}_{\text{refl}}$.

On-policy Counterfactual Distillation can be directly interleaved with standard RL: every M GRPO steps, we insert one cycle and then resume RL from the updated checkpoint. As shown in Table 3, under the same extended training budget where standalone GRPO has plateaued, this interleaved schedule continues to deliver gains, breaking through the ceiling of outcome-driven RL.

4 Experiment

We evaluate Counterfactual Distillation (CD) across a diverse range of agentic tasks, spanning competitive programming and multi-step interactive reasoning in simulated physical and digital environments.

Models. We evaluate Counterfactual Distillation using the Qwen2.5 (7B/72B) (Team et al., 2024) and Llama-3/3.1 (8B/70B) (Grattafiori et al., 2024) series. These models are widely adopted as standardized backbones in recent RLVR (Zeng et al., 2025b; Shao et al., 2025) and agentic reasoning (Da et al., 2025; Feng et al., 2025) research.

Datasets. We evaluate on four interactive benchmarks that demand strategic interaction, long-horizon decision making, and mistake recovery: (1) *WebShop* (Yao et al., 2022a), which requires multi-hop search and attribute matching across over one million products; (2) *ALFWorld* (Shridhar et al., 2020), a text-based interactive environment aligned with embodied household tasks; (3) *ScienceWorld* (Wang et al., 2022), a text-based scientific experimentation environment requiring multi-step manipulation under procedural constraints; and (4) *Sokoban* (Schrader, 2018), a planning-intensive puzzle environment requiring precise sequential manipulation and correction of earlier mistakes. We additionally evaluate on *CodeContests* (Li et al., 2022), a competitive programming benchmark with execution-based test-case verification. Together, these benchmarks cover web navigation, embodied interaction, scientific experimentation, and program synthesis.

Baselines & Metrics. We compare Counterfactual Distillation against four baselines: *Base* (instruct model with no task-specific fine-tuning), *GRPO-RLVR* (outcome-supervised RL with verifiable final rewards via GRPO (Guo et al., 2025)), *EarlyExp* (reward-free agent learning that converts early interaction experience into supervision (Zhang et al., 2025a)), and *ACE* (a training-free, prompt-based method that improves agents via evolving playbooks from execution feedback (Zhang et al., 2025b)).

We report Pass@1 and Pass@128 (Sec. 2.3). Pass@1 quantifies single-try pass rate as a measure of exploitation, while Pass@128 estimates best-of-k performance under a much larger sampling budget, serving as a proxy for exploration capacity and the model's fundamental performance boundary. All reported Pass@128 values are computed from *independent* inference-time samples from the trained policy (i.e., without Stage 1 rollback or experience-guided branching). For Qwen2.5-7B-Instruct, Appendix D.6 reports mean $\pm$ SD from 32 complete Pass@1 evaluations and 32 Pass@128 subsamples of fixed rollout pools.

Implementation Details. For the interactive agentic benchmarks, we use *verl-agent* (Feng et al., 2025) as a unified framework for both training and evaluation. We adopt the framework-provided settings and interfaces for *ALFWorld*, *WebShop*, and *Sokoban*, and implement a compatible verl-agent

Table 1: Main results on four agent benchmarks. We report Pass@1(%) and Pass@128(%) for two models per dataset. Counterfactual Distillation consistently outperforms baselines across all tasks in terms of Pass@128.

	WEBSHOP				ALFWORLD			
	QWEN2.5-7B		LLAMA3.1-8B		QWEN2.5-7B		LLAMA3.1-8B	
METHOD	PASS@1	PASS@128	PASS@1	PASS@128	PASS@1	PASS@128	PASS@1	PASS@128
BASE (NO FT)	0.05	5.20	0.00	1.80	26.07	78.57	29.82	74.29
EARLYEXP	61.55	84.60	51.13	77.80	**72.23**	92.86	**74.15**	95.71
GRPO-RLVR	67.45	85.40	54.95	79.40	69.46	91.43	72.50	94.29
ACE	**68.65**	86.80	54.35	79.80	66.34	89.63	70.32	92.50
OURS	66.50	**89.20**	**56.25**	**83.20**	67.50	**94.29**	71.79	**96.43**
	SCIWORLD				SOKOBAN			
	QWEN2.5-7B		LLAMA3.1-8B		QWEN2.5-7B		LLAMA3.1-8B	
METHOD	PASS@1	PASS@128	PASS@1	PASS@128	PASS@1	PASS@128	PASS@1	PASS@128
BASE (NO FT)	7.00	47.33	7.17	48.67	6.90	43.80	17.70	61.40
EARLYEXP	26.17	54.67	24.04	56.00	60.15	71.60	57.32	68.20
GRPO-RLVR	27.17	57.33	24.25	56.00	58.15	68.00	60.43	73.40
ACE	**28.45**	50.00	**25.28**	57.33	61.30	70.80	60.79	73.20
OURS	27.88	**62.00**	22.70	**59.33**	**64.60**	**78.40**	**62.00**	**77.20**

wrapper for *ScienceWorld*. For *CodeContests*, we train with *verl* (Sheng et al., 2025) and follow *CTRL* (Xie et al., 2025) for execution-based evaluation.

For baselines, *Base* is the instruction-tuned model without task-specific training, and *GRPO-RLVR* applies task-specific GRPO with verifiable rewards. For *EarlyExp*, since no official codebase is available, we re-implement its Implicit World Modeling (IWM) component via the *verl* SFT module followed by GRPO training. We use the IWM variant for the main comparison because EarlyExp-SR relies on expert demonstrations, whereas our setting learns from model-generated interaction and environment feedback; results with EarlyExp-SR are reported separately in Table 9. For *ACE*, we extend its official *AppWorld* playbook-based prompting strategy to our benchmarks and apply it on top of the GRPO-trained model under the same *verl-agent* protocol. Our method is initialized from the GRPO-trained model and distills reflective experience data. More details are in Appendix D.

4.1 Main Results

Large-k capability gains. Table 1 summarizes results on WebShop, ALFWorld, ScienceWorld, and Sokoban across Qwen2.5 and Llama3.1 models. Table 2 further reports results on CodeContests with larger backbones. We report both Pass@1 and Pass@128 to distinguish single-rollout performance from broader capability under increased sampling.

Our advantage is most pronounced at large k. While GRPO sometimes matches or exceeds Counterfactual Distillation on Pass@1, its gains plateau as the sampling budget grows, whereas our method consistently achieves stronger Pass@128, indicating broader behavioral coverage. For example, on WebShop with Qwen2.5-7B, GRPO obtains a higher Pass@1, but Counterfactual Distillation achieves a higher Pass@128. This is consistent with *distribution sharpening*: GRPO increases the likelihood of already-successful trajectories, but provides limited additional coverage as k grows. The contrast is especially clear on CodeContests, where Counterfactual Distillation improves Pass@128 by up to **14 points** over the base model (11 points over GRPO) with Qwen2.5-72B. Overall, these results show that our method better internalizes feedback-grounded agency and expands the model's effective capability in long-horizon interaction.

Raising the training ceiling of GRPO. To study the ceiling of GRPO in long-horizon interactive tasks, we evaluate on-policy Counterfactual Distillation(§3.3) interleaved with RL (RL+CD) on WebShop and Sokoban. Starting from the GRPO-RLVR checkpoint in the main table, we continue training Qwen2.5-7B-Instruct with GRPO for 100 extra steps on WebShop (200 total) and 150 extra steps on Sokoban (300 total). In RL+CD, every 50 GRPO steps we insert 5 on-policy Counterfactual Distillation cycles (data collection and training), then resume GRPO from the updated checkpoint.

Table 2: Main results on CodeContests with larger backbone models. Ours uses GRPO→CD. Counterfactual Distillation yields substantial improvements in Pass@128 over the GRPO-RLVR baseline. *Note:* EarlyExp and ACE are not reported on CodeContests because it is designed for interactive environments and does not directly apply to code execution feedback without a different implementation.

	QWEN2.5-72B		LLAMA3-70B	
METHOD	PASS@1	PASS@128	PASS@1	PASS@128
BASE (NO FT)	10.00	33.94	7.35	24.85
GRPO-RLVR	**20.45**	36.97	13.64	27.88
OURS	17.12	**47.88**	**14.09**	**33.94**

Table 3: Supplementary analysis of the GRPO training ceiling on Qwen2.5-7B-Instruct. We report Pass@1 (%) and Pass@128 (%) on WebShop and Sokoban. Compared with extended GRPO training, the on-policy RL+CD variant achieves higher performance, suggesting that interleaving Counterfactual Distillation with GRPO can further raise the training ceiling of outcome-driven RL.

	WEBSHOP		SOKOBAN	
METHOD	PASS@1	PASS@128	PASS@1	PASS@128
GRPO (MAIN TABLE)	67.45	85.40	58.15	68.00
GRPO (+STEPS)	71.22	88.60	70.70	75.20
RL+CD(INTERLEAVED)	**74.33**	**91.00**	**78.70**	**81.20**

As shown in Table 3, extended GRPO training improves the success rate to 71.22% on WebShop and 70.70% on Sokoban, close to the observed ceiling, as further GRPO updates bring little additional gain. In comparison, RL+CD raises performance to 74.33% and 78.70%, respectively. For Pass@128, extended GRPO reaches 88.60% on WebShop and 75.20% on Sokoban, while RL+CD achieves 91.00% and 81.20%, respectively. These results suggest that late-stage GRPO is limited by sparse rewards and existing decision patterns, making it difficult to induce broader exploration through more RL steps alone. By distilling exploration-derived experience into reflection and revised decisions, on-policy Counterfactual Distillation injects feedback-grounded corrective strategies into the training loop, expanding the effective exploration space and raising the attainable ceiling of outcome-driven RL.

4.2 Capability Scaling: Pass@k Analysis

Figure 3 analyzes Pass@k as the sampling budget increases, with task-specific caps of k=512 for ALFWorld, k=256 for ScienceWorld, and k=1024 for CodeContests. Across all benchmarks, our method consistently performs best in the large-k regime, indicating a higher capability ceiling, with especially clear gains on CodeContests. The scaling curves further show that Counterfactual Distillation reaches comparable success rates with fewer samples and maintains stronger performance as k grows. These results suggest that our method improves not only the attainable upper bound of the policy, but also the efficiency with which additional sampling translates into successful trajectories.

4.3 Effectiveness Analysis

Tree-Structured Exploration for Experience Generation. Table 7 compares three sampling strategies on CodeContests under the same execution budget: independent sampling, iterative refinement using previous execution feedback, and our tree-based rollback branching. Across all models, our Stage 1 strategy achieves the highest Pass@128, showing that feedback-guided branching from critical failure points is more effective than independent or linear refinement.

Counterfactual Distillation. Table 4 compares rehearsal alone ($\mathcal{L}_{reh}$) with rehearsal plus counterfactual distillation ($\mathcal{L}_{cf} + \mathcal{L}_{reh}$) on ScienceWorld. While $\mathcal{L}_{reh}$ preserves successful behaviors through rejection-sampling-style imitation (Ahn et al., 2024), adding $\mathcal{L}_{cf}$ substantially improves Pass@128 with comparable Pass@1 across model sizes, e.g., $59.33\% \rightarrow 62.00\%$ on Qwen2.5-7B

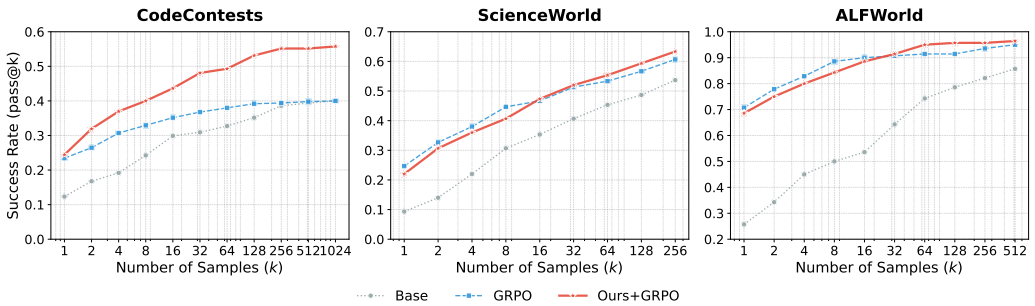


Figure 3: **Scaling results on different benchmarks.** We plot the Pass@k success rate as a function of the number of samples k. Our method (red) consistently achieves higher efficiency and performance ceilings across all tasks compared to the baselines.

Table 4: Ablation of counterfactual distillation on ScienceWorld. Adding $\mathcal{L}_{\text{cf}}$ to rehearsal ($\mathcal{L}_{\text{reh}}$) improves Pass@128 across all three backbones, with comparable Pass@1.

Model	**Pass@1**		**Pass@128**	
	$\mathcal{L}_{\text{reh}}$	$\mathcal{L}_{\text{cf}} + \mathcal{L}_{\text{reh}}$	$\mathcal{L}_{\text{reh}}$	$\mathcal{L}_{\text{cf}} + \mathcal{L}_{\text{reh}}$
Qwen2.5-7B	27.67	27.88	59.33	62.00
Llama3.1-8B	22.54	22.70	57.33	59.33
Qwen2.5-14B	37.17	36.50	67.33	72.00

and $67.33\% \rightarrow 72.00\%$ on Qwen2.5-14B. This indicates that counterfactual distillation is crucial for internalizing corrective revisions, while rehearsal mainly stabilizes training.

4.4 Ablation Study

Counterfactual Context Supervision. Table 5 compares CD (Ours) with CD (keep exp) under matched training conditions (Appendix D.4). CD (keep exp) reinforces revised actions with the hindsight experience e that generated them. CD (Ours) instead learns the same actions under the original context (h_τ, q) without e, achieving higher Pass@128 on both tasks. These results suggest that counterfactual context supervision internalizes hindsight-guided action improvements in model parameters, enabling effective decisions without experience at test time.

Scaling Behavior Across Model Sizes. As shown in Fig. 4, Pass@128 exhibits a consistent scaling trend across benchmarks. Increasing the model size from Qwen2.5-7B to 14B yields gains of +0.6 pp on WebShop and +10.0 pp on ScienceWorld, and the transition from Qwen2.5-32B to 72B yields +13.83 pp on CodeContests. Our method consistently outperforms all baselines across these scales. The robustness of these gains across both Qwen and Llama architectures demonstrates that Counterfactual Distillation effectively leverages increased model capacity to raise the performance ceiling.

Out-of-Distribution Generalization. To evaluate Out-of-Distribution (OOD) generalization, we test models trained on CodeContests on the held-out MBPP benchmark. As shown in Table 6, our method shows superior robustness to distribution shift than the GRPO baseline. While GRPO leads to severe degradation on OOD tasks (e.g., a drop of 4.2 percentage points on Llama3-70B), Counterfactual Distillation not only mitigates this drop but still surpasses the original base model. This suggests that our method helps the model learn fundamental reflective agency rather than overfitting to dataset-specific shortcuts.

We provide a more detailed ablation study in Appendix C.

5 Related Work

Self-Evolving LLM Agents. A growing body of work studies how LLM agents improve through interaction, reflection, and accumulated experience. One line externalizes learning into prompts,

Table 5: Ablation of the training context on Qwen2.5-7B-Instruct. Using the same revision data, CD (Ours) removes experience e from the training input, while CD (keep exp) retains it. CD (Ours) achieves higher Pass@128 on both tasks with counterfactual supervision context.

	WebShop		Sokoban	
Method	Pass@1	Pass@128	Pass@1	Pass@128
GRPO-RLVR	$\mathbf{67.32} \pm 1.17$	85.40	58.33 ± 0.55	68.00
CD (keep exp)	65.86 ± 0.98	84.80	60.23 ± 0.65	72.00
CD (Ours)	66.38 ± 1.09	**89.20**	$\mathbf{65.32} \pm 0.98$	**78.40**

memories, or reusable experience libraries without directly updating model weights, as in ReasoningBank (Ouyang et al., 2025), FLEX (Cai et al., 2025), and ACE (Zhang et al., 2025b). Another line performs *training-based* self-evolution by iteratively turning interaction data into better policies, skills, or strategic abstractions: SKILLRL (Xia et al., 2026) and EvolveR (Wu et al., 2025) distill trajectories into reusable skills or principles, while Agent0 (Xia et al., 2025), Absolute Zero (Zhao et al., 2025a), and CodeIt (Butt et al., 2024) couple self-improvement with automatic task generation, self-play style curricula, or hindsight relabeling. A third line focuses on *test-time* self-improvement, either through lightweight test-time fine-tuning on synthesized data (Acikgoz et al., 2025) or through repeated evolution of prompts, memory, and tool-use configurations across attempts (He et al., 2025b). Our work emphasizes internalizing feedback-grounded exploratory experience into the policy for long-horizon interaction.

Learning from Experience. Learning from experience is a long-standing goal in agentic systems. Before LLMs, model-based RL such as World Models (Ha and Schmidhuber, 2018) and Dreamer (Hafner et al., 2023, 2020) learned latent dynamics to imagine future outcomes for planning. For LLM agents, Early Experience (EarlyExp) (Zhang et al., 2025a) learns from self-generated traces via implicit world modeling and self-reflection; Agent Q (Putta et al., 2024) combines guided MCTS and self-critique with off-policy DPO to learn from successes and failures; and SPA (Chen et al., 2025a) internalizes world models via self-play finetuning. Related rollback or hindsight-based methods further exploit failed trajectories: HOPE (Wang and Choudhury) uses hindsight counterfactual actions to drive multi-turn RL exploration, while GA-Rollback (Li et al., 2025b) triggers stepwise rollback to prevent error propagation. In contrast, Counterfactual Distillation explicitly localizes critical failure points during tree-structured interaction and distills the resulting experience-guided corrective actions under the original decision contexts without the experience text, thereby internalizing the revisions into a single-rollout policy without requiring external memory or heavy test-time search.

Additional background on LLM agents, RLVR, and Pass@k optimization is provided in Appendix B.

6 Limitations and Conclusion

We presented Counterfactual Distillation, a framework that internalizes reflective experience into LLM agents by converting rollout-time feedback into counterfactual policy supervision. Through tree-structured exploration, the agent localizes failure points, generates experience-guided revisions, and distills these revisions under the original decision contexts. Across long-horizon agentic benchmarks, this improves Pass@k, especially at large k, indicating broader behavioral coverage and a higher capability ceiling than outcome-driven RLVR.

The main limitation is its reliance on informative feedback and replayable environment states. When feedback is sparse, delayed, or ambiguous, reflected experience may be unreliable; when environments are highly stochastic or stateful, exact rollback may be impractical. Extending the method to such settings may require additional state tracking or approximate replay mechanisms.

Acknowledgements

We sincerely thank the anonymous reviewers for their insightful and constructive feedback. We gratefully acknowledge MBZUAI and NVIDIA for providing the computational resources that made this work possible.

References

Emre Can Acikgoz, Cheng Qian, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. Self-improving llm agents at test-time. *arXiv preprint arXiv:2510.07841*, 2025.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*, 2024.

Natasha Butt, Blazej Manczak, Auke Wiggers, Corrado Rainone, David W Zhang, Michaël Defferrard, and Taco Cohen. Codeit: Self-improving language models with prioritized hindsight replay. *arXiv preprint arXiv:2402.04858*, 2024.

Zhicheng Cai, Xinyuan Guo, Yu Pei, Jiangtao Feng, Jinsong Su, Jiangjie Chen, Ya-Qin Zhang, Wei-Ying Ma, Mingxuan Wang, and Hao Zhou. Flex: Continuous agent evolution via forward learning from experience. *arXiv preprint arXiv:2511.06449*, 2025.

Shiqi Chen, Tongyao Zhu, Zian Wang, Jinghan Zhang, Kangrui Wang, Siyang Gao, Teng Xiao, Yee Whye Teh, Junxian He, and Manling Li. Internalizing world models via self-play finetuning for agentic rl. *arXiv preprint arXiv:2510.15047*, 2025a.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@ k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025b.

Mingyue Cheng, Jie Ouyang, Shuo Yu, Ruiran Yan, Yucong Luo, Zirui Liu, Daoyu Wang, Qi Liu, and Enhong Chen. Agent-r1: Training powerful llm agents with end-to-end reinforcement learning. *arXiv preprint arXiv:2511.14460*, 2025.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Jeff Da, Clinton Wang, Xiang Deng, Yuntao Ma, Nikhil Barhate, and Sean Hendryx. Agent-rlvr: Training software engineering agents via guidance and environment rewards. *arXiv preprint arXiv:2506.11425*, 2025.

Xingyu Dang, Christina Baek, Kaiyue Wen, Zico Kolter, and Aditi Raghunathan. Weight ensembling improves reasoning in language models. *arXiv preprint arXiv:2504.10478*, 2025.

Jia Deng, Jie Chen, Zhipeng Chen, Daixuan Cheng, Fei Bai, Beichen Zhang, Yinqian Min, Yanzipeng Gao, Wayne Xin Zhao, and Ji-Rong Wen. From trial-and-error to improvement: A systematic analysis of llm exploration mechanisms in rlvr. *arXiv preprint arXiv:2508.07534*, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025.

Jingchu Gai, Guanning Zeng, Huaqing Zhang, and Aditi Raghunathan. Differential smoothing mitigates sharpening and improves llm reasoning. *arXiv preprint arXiv:2511.19942*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3), 2018.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*, 2025a.

Yufei He, Juncheng Liu, Yue Liu, Yibo Li, Tri Cao, Zhiyuan Hu, Xinxing Xu, and Bryan Hooi. Evotest: Evolutionary test-time learning for self-improving agentic systems. *arXiv preprint arXiv:2510.13220*, 2025b.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Xingguang Ji, Yahui Liu, Qi Wang, Jingyuan Zhang, Yang Yue, Rui Shi, Chenxi Sun, Fuzheng Zhang, Guorui Zhou, and Kun Gai. Leanabell-prover-v2: Verifier-integrated reasoning for formal theorem proving via reinforcement learning. *arXiv preprint arXiv:2507.08649*, 2025.

Long Li, Zhijian Zhou, Jiaran Hao, Jason Klein Liu, Yanting Miao, Wei Pang, Xiaoyu Tan, Wei Chu, Zhe Wang, Shirui Pan, et al. The choice of divergence: A neglected key to mitigating diversity collapse in reinforcement learning with verifiable reward. *arXiv preprint arXiv:2509.07430*, 2025a.

Xingzuo Li, Kehai Chen, Yunfei Long, Xuefeng Bai, Yong Xu, and Min Zhang. Generator-assistant stepwise rollback framework for large language model agent. *arXiv preprint arXiv:2503.02519*, 2025b.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022. doi: 10.1126/science.abq1158. URL https://www.science.org/doi/abs/10.1126/science.abq1158.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion Blog.

MiniMax. Minimax m2 & agent: Ingenious in simplicity. https://www.minimax.io/news/minimax-m2, October 2025. Accessed: 2026-01-28.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*, 2024.

Penghui Qi, Zichen Liu, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Optimizing anytime reasoning via budget relative policy optimization. *arXiv preprint arXiv:2505.13438*, 2025.

Max-Philipp B. Schrader. gym-sokoban. https://github.com/mpSchrader/gym-sokoban, 2018.

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

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

Shijie Shang, Ruosi Wan, Yue Peng, Yutong Wu, Xiong-hui Chen, Jie Yan, and Xiangyu Zhang. Stepfun-prover preview: Let's think and verify step by step. *arXiv preprint arXiv:2507.20199*, 2025.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, et al. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*, 2025.

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.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Yuda Song, Julia Kempe, and Remi Munos. Outcome-based exploration for llm reasoning. *arXiv preprint arXiv:2509.06941*, 2025.

Sijun Tan, Michael Luo, Colin Cai, Tarun Venkat, Kyle Montgomery, Aaron Hao, Tianhao Wu, Arnav Balyan, Manan Roongta, Chenguang Wang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. rllm: A framework for post-training language agents, 2025. URL https://pretty-radio-b75.notion.site/rLLM-A-Framework-for-Post-Training-Language-Agents-21b81902c146819db63cd98a54ba5f31. Notion Blog.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Qwen Team et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3), 2024.

Christian Walder and Deep Karkhanis. Pass@ k policy optimization: Solving harder reinforcement learning problems. *arXiv preprint arXiv:2505.15201*, 2025.

Huaxiaoyue Wang and Sanjiban Choudhury. The road not taken: Hindsight exploration for llms in multi-turn rl. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*, 2022.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. *arXiv preprint arXiv:2511.16043*, 2025.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Zhihui Xie, Jie Chen, Liyu Chen, Weichao Mao, Jingjing Xu, and Lingpeng Kong. Teaching language models to critique via reinforcement learning. *arXiv preprint arXiv:2502.03492*, 2025.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022b.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

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. *arXiv preprint arXiv:2503.14476*, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Z.ai. Glm-4.7: Advancing the coding capability, 2025. URL `https://z.ai/blog/glm-4.7`.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*, 2025a.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025b.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2025a.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025b.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025a.

Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: Rl post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*, 2025b.

Yaoyu Zhu, Di Huang, Hanqi Lyu, Xiaoyun Zhang, Chongxiao Li, Wenxuan Shi, Yutong Wu, Jianan Mu, Jinghua Wang, Pengwei Jin, et al. Qimeng-codev-r1: Reasoning-enhanced verilog generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

A Algorithm Detail

Algorithm 1 COUNTERFACTUAL DISTILLATION Stage 1: Tree-Structured Experience Search

Input: environment $\mathcal{E}$, policy π_θ, instruction q, period K, max steps T, budget B
Initialize $\mathcal{W} \leftarrow \{(1, \emptyset, \emptyset)\}$; $\mathcal{D}_{\text{traj}} \leftarrow \emptyset$; $\mathcal{D}_{\text{refl}} \leftarrow \emptyset$
$count \leftarrow 0$
while $\mathcal{W} \neq \emptyset$ **and** $count < B$ **do**
  Dequeue $(\tau, e, a_{1:\tau-1})$ from $\mathcal{W}$; $count \leftarrow count + 1$
  $\text{Reset}(\mathcal{E})$; $\text{Replay}(\mathcal{E}, a_{1:\tau-1})$ {Restore E_τ and h_τ}
  $t \leftarrow \tau$
  while $t \leq T$ **and** $\mathcal{E}$ not terminal **do**
    if $t = \tau$ **and** $e \neq \emptyset$ **then**
      $a_t \sim \pi_\theta(\cdot \mid h_t, q, e)$ {Generate improved action a'_τ}
      $\mathcal{D}_{\text{refl}} \leftarrow \mathcal{D}_{\text{refl}} \cup \{(h_t, e, a_t)\}$ {Collect experience-guided sample}
    else
      $a_t \sim \pi_\theta(\cdot \mid h_t, q)$
    end if
    $E_{t+1}, o_{t+1} \leftarrow \text{Step}(E_t, a_t)$; $h_{t+1} \leftarrow h_t \cup (a_t, o_{t+1})$
    if $t \bmod K = 0$ **or** $\mathcal{E}$ fails **then**
      $(\hat{\tau}, \hat{e}) \sim \pi_\theta(\cdot \mid h_{t+1}, p_{\text{refl}})$ {Identify failure and provide guidance}
      Enqueue $(\hat{\tau}, \hat{e}, a_{1:\hat{\tau}-1})$ into $\mathcal{W}$
    end if
    if $\mathcal{E}$ succeeds **then**
      $\mathcal{D}_{\text{traj}} \leftarrow \mathcal{D}_{\text{traj}} \cup \{h_{t+1}\}$
    end if
    $t \leftarrow t + 1$
  end while
end while

B Related Work

LLM-based Agents. Large Language Models (LLMs) have evolved from passive generators into autonomous agents capable of tool-use and multi-step reasoning. Early work largely relies on prompting to elicit agent behaviors without weight updates, including ReAct (Yao et al., 2022b), Reflexion (Shinn et al., 2023), and Tree of Thoughts (Yao et al., 2023). Recently, reinforcement-learning (often RLVR) has become a major route to improve reasoning robustness and long-horizon decision making, exemplified by DeepSeek-R1 (Guo et al., 2025) and OpenAI o1 (Jaech et al., 2024). Building on this direction, systems such as verl-agent/GiGPO (Feng et al., 2025), rLLM (Tan et al., 2025), and Agent-R1 (Cheng et al., 2025) scale multi-turn RL for tool-using agents, while open-weight releases (e.g., Kimi K2 (Team et al., 2025), GLM-4.5/4.7 (Zeng et al., 2025a; Z.ai, 2025), MiniMax-M2 (MiniMax, 2025)) further accelerate reproducible agent research.

Reinforcement Learning with Verifiable Rewards (RLVR). RLVR post-trains LLMs with automatically verifiable signals (e.g., exact-match answers, unit tests, theorem provers, or environment states), and has become a widely adopted recipe for improving reasoning on mathematics and programming-style tasks (Guo et al., 2025; Yu et al., 2025). PPO (Schulman et al., 2017) is a standard baseline, while critic-free group-based optimizers such as GRPO (Shao et al., 2024; Guo et al., 2025) and step-aware variants like GiGPO (Feng et al., 2025) improve stability and credit assignment; these recipes have been reproduced and scaled in open settings (e.g., SimpleRL-Zoo (Zeng et al., 2025b), ORZ (Hu et al., 2025), and Skywork-OR1 (He et al., 2025a)). Beyond sparse outcome rewards, recent work explores denser step-wise supervision via process reward modeling and automated process verifiers (Setlur et al., 2024) as well as implicit/dense reward construction (Cui et al., 2025). RLVR is also being scaled and extended through efficient small-model RL scaling (Luo et al., 2025), budget-aware optimization (Qi et al., 2025), and system-level acceleration (e.g., asynchronous rollouts) (Fu et al., 2025), and has expanded to new verifiable domains such as Verilog generation (Zhu et al., 2025) and Lean-based theorem proving (Ji et al., 2025; Shang et al., 2025).

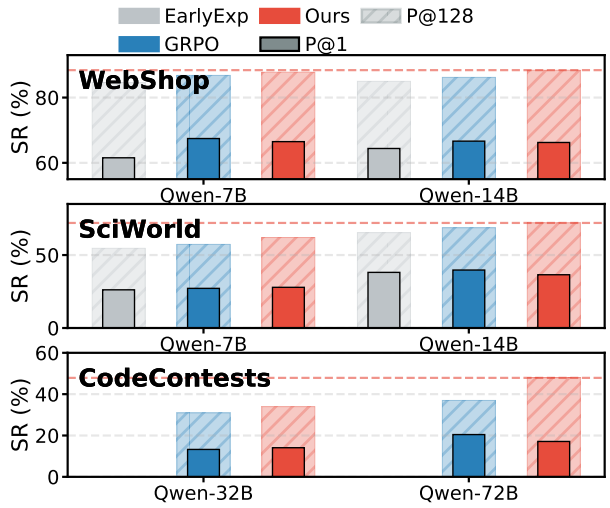


Figure 4: Main results on WebShop and SciWorld. Bars represent **Pass@1** (solid) and **Pass@k** (hatched) (%). **Ours** consistently outperforms GRPO and other baselines across different model architectures and scales.

Table 6: OOD generalization on MBPP after training on CodeContests. Values are Pass@128 (%); superscripts show absolute changes from the base model, with red for drops and blue for gains. Counterfactual Distillation preserves OOD performance better than GRPO and can surpass the base model.

Method	**Qwen2.5-32B**	**Qwen2.5-72B**	**Llama3-70B**
Base	85.45	83.33	78.31
GRPO	$81.22^{-4.2}$	$81.22^{-2.1}$	$74.07^{-4.2}$
Counterfactual Distillation	$\mathbf{85.45}^{+0.0}$	$\mathbf{85.13}^{+1.8}$	$\mathbf{79.63}^{+1.3}$

Pass@k Optimization Methods. Recent work has studied Pass@k not only as an evaluation metric, but also as an objective or diagnostic for balancing exploration and exploitation in reasoning models. Several studies observe that RLVR can improve Pass@1 while providing limited gains, or even causing degradation, at large k, suggesting that outcome-driven training may sharpen existing solution modes rather than expand the model's capability boundary (Yue et al., 2025; Dang et al., 2025). To address this, Pass@k Training (Chen et al., 2025b) directly optimizes a Pass@k-oriented objective to adaptively balance exploration and exploitation, while Pass@k Policy Optimization (Walder and Karkhanis, 2025) formulates policy optimization around the utility of multiple sampled outputs. Other works improve large-k behavior by encouraging broader exploration or mitigating diversity collapse, such as outcome-based exploration (Song et al., 2025), divergence choices that better preserve policy support (Li et al., 2025a), differential smoothing (Gai et al., 2025), and systematic analyses of exploration mechanisms in RLVR (Deng et al., 2025). Unlike these methods, which mainly focus on final-answer diversity or sample-set optimization, Counterfactual Distillation targets long-horizon interactive agents: it uses environment feedback to identify critical failure points and distills experience-guided corrective decisions into the policy, thereby improving Pass@k through decision-level behavioral coverage.

C Ablation Study Results

Synergy and Trade-offs with Auxiliary Training Target. We investigated the synergy of Counterfactual Distillation with EarlyExp and GRPO. For agentic tasks, GRPO first builds basic interaction skills; coding models already have basic coding skills, so we also evaluate CD alone on CodeContests. CodeContests does not use EE. As presented in Table 8, integrating these techniques consistently improves Pass@1 success rates across all evaluated benchmarks: For instance, adding EarlyExp

Table 7: CodeContests Pass@128 under different sampling strategies with Independent Sampling (IS), Iterative Refinement (IR), and our stage 1.

Model	IS	IR	Ours (Stage 1)
Qwen2.5-32B	48.92	51.48	**55.52**
Qwen2.5-72B	48.65	49.52	**54.30**
Llama3-70B	30.20	38.10	**42.50**

Table 8: Ablation study on the impact of auxiliary training stages (i.e., EarlyExp (EE) and GRPO (RL)). For ALFWorld, Ours is GRPO→CD, and w/ EE adds EE before GRPO. For CodeContests, Ours is CD, and w/ RL adds GRPO before CD. Results are reported as Pass@1 / Pass@128 (%). While these additional stages consistently enhance single-rollout performance (Pass@1), they may occasionally constrain the exploration ceiling (Pass@128).

Task	Model	Ours	Ours + Enhancement
ALFWorld	Qwen2.5-7B	67.50 / 94.29	**67.86 / 95.71** (w/ EE)
	Llama3.1-8B	71.79 / **96.43**	**74.46** / 93.57 (w/ EE)
CodeContests	Qwen2.5-32B	9.34 / **34.35**	**15.38** / 34.05 (w/ RL)
	Qwen2.5-72B	11.15 / 40.00	**17.12 / 47.88** (w/ RL)
	Llama3-70B	9.33 / 30.30	**14.09 / 33.94** (w/ RL)

before GRPO for Llama3.1-8B on ALFWorld increases accuracy from 71.79% to 74.46%, while adding GRPO before CD boosts Qwen2.5-72B on CodeContests from 11.15% to 17.12%. However, these gains in exploitation do not always translate to enhanced model capability at a higher sampling budget. We observe that Pass@128 performance can sometimes decline with these additional training stages, notably for Llama3.1-8B (96.43% → 93.57%) and Qwen2.5-32B (34.35% → 34.05%), suggesting that while EarlyExp and GRPO effectively align the primary training target, they may narrow the exploration space. In contrast, Counterfactual Distillation consistently improves the model's capacity as analyzed in §4.1.

Comparison with Early Experience-SR. Early Experience is an important baseline for learning from agent interaction. In the main experiments, we primarily compare with its Implicit World Modeling (IWM) variant, since the Self-Reflection (SR) variant relies on expert demonstrations and is therefore less aligned with our main comparison setting, where supervision is obtained from model-generated interaction and environment feedback. To provide a more complete comparison, we additionally reproduce Early Experience-SR on ScienceWorld with Qwen2.5-7B-Instruct. Following the SR setup, we first train an Imitation Learning baseline using expert demonstrations, and then train Early Experience-SR with a mixed IL+SR dataset. In our reproduction, the Imitation Learning data scale is about 14k samples, the mixed IL+SR training set is about 28k samples, and the Counterfactual Distillation training data contains about 4k samples. As shown in Table 9, EarlyExperience-SR achieves the best Pass@1 score of 40.87%, slightly higher than Counterfactual Distillation at 40.50%. However, Counterfactual Distillation achieves a substantially higher Pass@128 score, improving from 82.67% to 86.00%. This result suggests that Counterfactual Distillation more effectively expands the policy's exploration space and achieves a higher capability ceiling under large sampling budgets.

Intermediate Pass@k Results. Table 10 extends the experience-removal ablation with intermediate sampling budgets on WebShop and Sokoban. The benefit is clearest at larger sampling budgets; CD does not lead at every intermediate k (e.g., WebShop at $k = 8$).

D Implementation Details

D.1 Summary

Table 11 summarizes the data scale used across all benchmarks. For each dataset, we report the exact number of instances in the task split (training set and test set). In addition, we list the average amount of auxiliary SFT data (EarlyExp and our Counterfactual Distillation) distilled from rollouts; these

Table 9: Comparison with Early Experience-SR on ScienceWorld using Qwen2.5-7B-Instruct. We report Pass@1 (%) and Pass@128 (%). Counterfactual Distillation achieves the best Pass@128, indicating stronger exploration capability under large sampling budgets.

METHOD	PASS@1	PASS@128
BASE	7.00	47.33
IMITATION LEARNING	37.33	80.67
EARLYEXPERIENCE-SR	**40.87**	82.67
COUNTERFACTUAL DISTILLATION	40.50	**86.00**

Table 10: Intermediate Pass@k results (%) on Qwen2.5-7B-Instruct under independent sampling. These point estimates extend the Pass@128 results in Table 5.

Task	Method	$k=8$	$k=16$	$k=32$	$k=64$	$k=128$
WebShop	GRPO-RLVR	76.20	77.80	82.00	84.20	85.40
	CD (keep exp)	**76.60**	78.40	81.20	83.40	84.80
	CD	75.80	**80.80**	**84.40**	**86.40**	**89.20**
Sokoban	GRPO-RLVR	63.20	64.60	66.00	67.40	68.00
	CD (keep exp)	69.20	70.40	71.20	71.80	72.00
	CD	**75.20**	**77.40**	**78.00**	**78.40**	**78.40**

values are approximate and meant to indicate the typical order of magnitude (in thousands) rather than an exact count, since the final number can vary slightly across runs.

Table 11: Dataset statistics and approximate numbers of training samples used by auxiliary supervision. Training/test set sizes are exact instance counts, while EarlyExp and Counterfactual Distillation refer to the average scale (order of magnitude) of distilled samples used for SFT across runs.

Dataset	**#Training Set**	**#Test Set**	**#EarlyExp samples**	**#Counterfactual Distillation samples**
ALFWorld	3553	140	10k	12k
ScienceWorld	2000	150	6k	12k
WebShop	5000	500	15k	10k
Sokoban	5000	500	10k	10k
CodeContests	5000	165	N/A	10k

D.2 Computational Resources

All experiments are conducted on NVIDIA GB200 nodes. Each node contains 4 GPUs. For the GRPO training on ScienceWorld and the large-model experiments on CodeContests with 70B-scale models, we use 2 nodes, corresponding to 8 GPUs in total. All other experiments are conducted on 1 node with 4 GPUs.

Table 12 summarizes the approximate computational cost of different training or data-construction procedures. For GRPO, the runtime varies substantially across datasets: ScienceWorld takes about 20 hours, CodeContests takes about 15 hours, and the remaining datasets are generally completed within 10 hours. For Early Experience and our method, the reported time includes both data collection and supervised training. For ACE, the reported time mainly corresponds to collecting interaction data and constructing the knowledge base. The number of steps denotes the approximate amount of data used in each procedure, measured in GRPO-equivalent training steps.

D.3 Implementation of Environment

ALFWorld. ALFWorld is a household embodied task suite built on text-based environments. At each step, the environment executes the agent's action and returns a textual observation describing the execution result and the updated scene state. In addition, ALFWorld provides a list of *admissible actions* for the next step, which we expose to the model as action candidates.

Table 12: Approximate computational cost of different methods. The runtime is reported in hours. The number of steps denotes the approximate amount of data used by each method, measured in GRPO-equivalent training steps.

	GRPO	Early Experience	ACE	Ours
Time (hr)	6–20	2	3	3
Steps	100–150	10	5	10

ScienceWorld. ScienceWorld is an interactive scientific reasoning environment in which the agent solves multi-step tasks by manipulating objects and instruments. After executing an action, the environment returns a textual observation describing the action outcome and the resulting world state. Unlike ALFWorld, ScienceWorld exposes a set of currently *available objects*. The agent must compose valid actions by selecting an action template and instantiating it with objects from this list (e.g., `use [object] on [object]`), following the environment's action syntax.

WebShop. WebShop simulates goal-directed online shopping as a browser-like interaction loop. At each step, the environment renders the current webpage content (including product lists, descriptions, and navigation elements). The agent can either issue a *search* action by providing a keyword query, or perform a *click* action by selecting a clickable element (e.g., a product entry, category link, or a back button) to navigate across pages toward the target purchase.

Sokoban. Sokoban is a grid-based puzzle environment in which the agent must push boxes onto designated target locations through sequential movement decisions. At each step, the environment executes one of four directional actions (*up*, *down*, *left*, *right*) and returns the updated grid state. In our setup, we use a *text-mode* representation of the environment, where each grid cell is rendered as a symbolic token: walls, floor tiles, targets, boxes, the player are represented by symbols such as `#`, `_`, `O`, `X`, `P`. This textual grid is exposed to the model as the observation, and the agent must infer spatial structure, object interactions, and long-horizon box-pushing plans directly from these symbolic layouts.

CodeContests. CodeContests is a competitive programming benchmark where each episode corresponds to a single coding problem. In our setup, at every round, the model produces a *complete* candidate solution (full code) for the problem. The environment compiles and executes the submitted code against test cases and returns execution feedback. On the public test set, the environment provides concrete runtime outputs (e.g., failing case outputs, errors, or mismatched results), whereas on the private set it reports only an aggregate correctness score. The model uses this feedback to generate a revised full-program submission in the next round.

D.4 Implementation of Our Method Counterfactual Distillation

Stage 1: Tree-Structured Exploration for Experience Generation. During rollout, we trigger *rollback reflection* either (i) every fixed interval of K interaction rounds, or (ii) immediately when an episode terminates in failure. At each reflection, we provide the model with the full interaction history up to the current step and prompt it to (a) select a specific *suboptimal round* τ to roll back to, and (b) summarize a concise *experience* e (diagnosis-and-fix instruction) grounded in the observed feedback. The chosen τ is intended to correspond to the decision that caused the trajectory to enter an incorrect branch or become unproductive. A concrete prompt example is shown in Appendix D.7. In our implementation, we set $K{=}10$ for ALFWorld and ScienceWorld, $K{=}5$ for WebShop and Sokoban, and $K{=}4$ for CodeContests. Notably, CodeContests has a maximum horizon of 4, so rollback is only invoked when the attempt budget is exhausted.

Stage 2: Counterfactual Distillation. We construct two types of supervised data for standard next-token likelihood training. First, *Behavior Rehearsal* samples successful trajectories to preserve the agent's base competence. We randomly select 20% of successful rollouts to form the rehearsal set $\mathcal{D}_{\mathrm{reh}}$. Second, *Experience-to-Policy Distillation* builds a *counterfactual* dataset $\mathcal{D}_{\mathrm{cf}}$ from branching points: when an experience e injected at round τ yields an improved action a'_τ, we treat a'_τ as the counterfactual target for the *original* history without providing e at training time. In practice, we

sample 3 counterfactual (branching) instances per task to populate $\mathcal{D}_{\mathrm{cf}}$. We then perform SFT on the union of $\mathcal{D}_{\mathrm{reh}}$ and $\mathcal{D}_{\mathrm{cf}}$ with batch size 128 and learning rate 1×10^{-6} for 2–3 epochs, depending on the dataset and model scale.

Protocol for the Experience-Removal Ablation. Both CD variants start from the same GRPO-trained Qwen2.5-7B-Instruct checkpoint and use the same reflection-generated revision data, training budget, and evaluation protocol. CD (keep exp) predicts a'_τ from (h_τ, q, e), whereas CD predicts the same target from (h_τ, q); both are evaluated without e, rollback, or experience-guided branching. The Pass@1 values in Table 5 are mean $\pm$ SD over 32 complete stochastic evaluations of each fixed checkpoint, with one rollout per test instance in each evaluation. These are repeated evaluations, not independent training runs, and need not equal the original point estimates in Table 1. The Pass@128 values in Table 5 are point estimates under independent sampling; the supplementary variability estimates are reported separately in Appendix D.6. Table 10 reports the intermediate Pass@k results.

Rollback-Tree Hyperparameters We use a small set of task-dependent hyperparameters for rollback-tree exploration. Across all main experiments, we set the maximum rollback-tree depth to $d = 3$. The reflection frequency f is chosen according to the typical interaction horizon of each environment: $f = 10$ for ALFWorld and ScienceWorld, $f = 5$ for WebShop, and $f = 4$ for CodeContests. Intuitively, environments with longer horizons can tolerate less frequent reflection, while shorter-horizon or faster-feedback tasks benefit from more frequent opportunities to summarize experience and branch from earlier decisions.

To examine whether our method is sensitive to these choices, we conduct a small hyperparameter ablation on ScienceWorld. As shown in Table 13, increasing the tree depth from $d = 3$ to $d = 5$ yields a small improvement, while making reflection overly frequent ($f = 5$) slightly hurts performance. This is likely because frequent reflection consumes the limited rollout budget more quickly, leaving fewer effective attempts for completing the task. Overall, the results suggest that our method is not highly sensitive to the rollback-tree settings within a reasonable range.

Table 13: Hyperparameter ablation of rollback-tree depth d and reflection frequency f on ScienceWorld(Based on imitation learning). The results suggest that our method is not highly sensitive to these settings: larger depth gives a small gain, while overly frequent reflection slightly degrades performance.

Metric	$d = 3, f = 10$	$d = 5, f = 10$	$d = 3, f = 5$
Pass@1	38.83	40.50	38.00
Pass@128	85.33	86.00	83.33

D.5 Implementation of Baselines

D.5.1 GRPO-RLVR

Interactive agentic benchmarks (ALFWorld, ScienceWorld, WebShop, Sokoban). For ALFWorld, ScienceWorld, WebShop, and Sokoban, we train the GRPO-RLVR baseline using the VERL-AGENT framework. ALFWorld, WebShop, and Sokoban are officially supported in the repository, and we directly follow the provided environment setup, default training configurations, and reward definitions. ScienceWorld is not officially implemented in VERL-AGENT; we therefore implement a compatible environment wrapper following the official interfaces and guidelines, while keeping the observation/action formatting and data structures consistent with existing tasks in the framework. Unless stated otherwise, we use the same default training setup and reward scheme as in VERL-AGENT. The only ScienceWorld-specific modification is in the memory module: we keep at most the most recent 10 interaction turns in the context.

Competitive programming benchmark (CodeContests). For CodeContests, we train GRPO-RLVR using the VERL framework. We follow the environment setup and evaluation protocol of CTRL: a solution receives reward of 1 if and only if the generated program passes *all* test cases, including both public and private tests; otherwise the reward is 0. We set the GRPO learning rate to

1×10^{-6}, the training batch size to 128, and use $n = 8$ rollouts per prompt. We train for 3 epochs on the CodeContests training set.

D.5.2 Early Experience

In this section, we describe our reproduction of the Early Experience baseline, with a focus on its key component, *Implicit World Modeling* (IWM). In short, IWM trains the agent to *predict the next environment observation* conditioned on the current interaction context and a candidate action, thereby implicitly capturing environment dynamics via supervised next-token prediction.

EarlyExp constructs IWM supervision using expert demonstrations. In our setting, we focus on interaction-driven learning: both our method and the GRPO-RLVR baseline obtain supervision from environment feedback rather than ground-truth trajectories. Therefore, for a fair comparison, we implement IWM supervision purely from model-generated rollouts: we rollout the *base model* on the training set and collect action–observation pairs, using the environment-returned observation as the prediction target. For each task instance, we perform 4 independent rollouts, select the highest-scoring attempt, and uniformly sample 3 action–observation pairs from that attempt to form the IWM supervision set. We then perform SFT on it with batch size 64 and learning rate 1×10^{-6} for 1 epoch. An example prompt is shown in D.7.

D.5.3 ACE

ACE proposes a prompt-based self-evolving agent framework in which experience is accumulated in an external *playbook* rather than internalized through parameter updates. The playbook is a structured context memory that stores reusable task-solving strategies, heuristics, and reflective insights extracted from past interactions. Its implementation is organized around three main roles: a *Generator* that interacts with the environment to solve tasks, a *Reflector* that analyzes trajectories and summarizes useful takeaways, and a *Curator* that updates the playbook by incorporating new entries.

The open-source implementation of ACE is built on AppWorld, which serves as a representative interactive agent benchmark. In our experiments, we extend the official implementation to ALFWorld, WebShop, ScienceWorld, and Sokoban by following the same overall ACE pipeline. Specifically, we reuse the official prompts for the Generator, Reflector, and Curator roles, while keeping the task execution environments and interaction formatting consistent with the VERL-AGENT setup used in our other interactive benchmarks.

Since the current ACE implementation only appends new playbook entries and does not perform effective pruning or consolidation, the playbook can grow quickly and lead to excessive context length. To control context size, we cap the number of playbook entries during training at 100. At test time, we use the playbook obtained after training as part of the model context for downstream evaluation.

D.6 Evaluation Protocol and Variability

We evaluate Qwen2.5-7B-Instruct on the four agent benchmarks using the same test sets, decoding settings, and success criteria as in the main experiments. All rollouts are sampled directly from fixed checkpoints, without Stage 1 rollback or experience-guided branching.

Pass@1. We perform 32 complete independent stochastic evaluations, with exactly one rollout per test instance in each evaluation. Table 14 reports the mean and sample SD of the resulting dataset-level success rates.

Table 14: Pass@1 variability on Qwen2.5-7B-Instruct. We report mean ± SD (%) over 32 complete stochastic evaluations of each fixed checkpoint.

Method	WebShop	ALFWorld	ScienceWorld	Sokoban
Base	0.05 ± 0.09	26.35 ± 1.78	7.37 ± 0.87	7.35 ± 1.13
GRPO-RLVR	$\mathbf{67.32 \pm 1.17}$	$\mathbf{69.72 \pm 2.32}$	27.56 ± 1.45	58.33 ± 0.55
CD (keep exp)	65.86 ± 0.98	68.12 ± 1.89	27.20 ± 1.03	60.23 ± 0.65
CD (Ours)	66.38 ± 1.09	67.58 ± 2.05	$\mathbf{28.14 \pm 2.35}$	$\mathbf{65.32 \pm 0.98}$

Pass@128. For each test instance and checkpoint, we generate a fixed pool of 256 independent rollouts. We randomly select 128 rollouts per instance and compute Pass@128 as defined in Section 2.3, repeating the subsampling with 32 random seeds. Table 15 reports the mean and sample SD across these 32 estimates, which reuse the same pools rather than generating new rollouts for every seed.

Table 15: Pass@128 variability on Qwen2.5-7B-Instruct. We report mean ± SD (%) over 32 subsamples from fixed pools of 256 rollouts per test instance.

Method	WebShop	ALFWorld	ScienceWorld	Sokoban
Base	5.33 ± 0.75	78.33 ± 0.56	46.13 ± 1.08	45.12 ± 0.79
GRPO-RLVR	85.29 ± 0.32	91.22 ± 0.35	56.88 ± 0.80	68.94 ± 0.81
CD (Ours)	$\mathbf{88.98 \pm 0.22}$	$\mathbf{94.10 \pm 0.31}$	$\mathbf{62.75 \pm 0.88}$	$\mathbf{78.83 \pm 0.37}$

Statistical scope and original results. For both tables, sample SD is computed across 32 scores with denominator 31. These SDs describe evaluation-time variation at fixed checkpoints, not variation across independent training runs; the Pass@128 SD is conditional on the generated rollout pools. The original results in Tables 1 and 3 remain point estimates and are not replaced by these supplementary means. In the original Pass@128 evaluation, we sample up to 128 rollouts per instance and stop at the first success, which does not change its binary Pass@128 outcome. The stored rollouts therefore do not form 128 complete Pass@1 evaluations; the 32 Pass@1 evaluations and the full 256-rollout pools above are collected separately.

D.7 Example Prompts

EarlyExp Example Prompt

User:
You are an expert agent operating in the ALFRED Embodied Environment. Your task is to: put two cellphone in bed. Prior to this step, you have already taken 7 step(s). Below are the most recent 5 observations and the corresponding actions you took:
[Observation 3: 'You arrive at desk 1...', Action 3: 'take cellphone 2 from desk 1']
[Observation 4: 'You pick up the cellphone 2 from the desk 1.', Action 4: 'go to bed 1']
[Observation 5: 'You arrive at bed 1. On the bed 1, you see ...', Action 5: 'go to desk 1']
[Observation 6: 'You arrive at desk 1. On the desk 1, you see ...', Action 6: 'take cellphone 1 from desk 1']
[Observation 7: 'Nothing happens.', Action 7: 'move cellphone 2 to bed 1']
You are now at step 8 and your current observation is: Nothing happens. Your admissible actions of the current situation are: ['examine cellphone 2', 'examine desk 1', 'go to bed 1', 'go to drawer 1', 'go to drawer 2', 'go to garbagecan 1', 'go to shelf 1', 'go to shelf 2', 'go to shelf 3', 'go to shelf 4', 'go to shelf 5', 'go to shelf 6', 'go to shelf 7', 'go to shelf 8', 'go to sidetable 1', 'inventory', 'look', 'move cellphone 2 to desk 1'].

The next action is **move cellphone 2 to desk 1**. Your task is to predict the environment observation after executing this action. Output only the observation.

Assistant:
You move the cellphone 2 to the desk 1...

Counterfactual Distillation Example Prompt

User:
[STEP 1] Observation: This room is called the bathroom. In it, you see... Action: go kitchen
[STEP 2] Observation: You move to the kitchen... Action: pick up apple
[STEP 3] Observation: You move the apple to the inventory... Action: go to the counter
[STEP 4] Observation: It's not clear how to go to/through a counter... Action: go stove
[STEP 5] Observation: It's not clear how to go to/through a stove... Action: go counter
[STEP 6] Observation: It's not clear how to go to/through a counter... Action: focus on apple
...
You are an expert agent playing ScienceWorld, a text-based science simulation game. Your task is to boil apple juice. For compounds without a boiling point, combusting the substance is also acceptable.
You have already taken 6 step(s), but the task failed without completing the objective.
FAILURE REASON: The environment closed early, but the task was not properly completed.

IMPORTANT: Analyze your failed attempt and identify where things went wrong. You have a chance to go back to a specific step and try a different approach, with the benefit of your current experience.
First, carefully analyze the trajectory within `<think> </think>` tags.
Then, provide your final decision in the following format: `<step>` ... `</step>` `<experience>` ... `</experience>`

Assistant:
`<think>` The trajectory initially seemed promising as I picked up the apple and moved towards the counter, but I became confused about how to proceed... `</think>`
`<step>2</step>`
`<experience>` Plan the full procedure before acting: focus on the target substance, use valid appliances/containers, and avoid actions that the environment rejects (e.g., going "to/through" non-locations). `</experience>`

E Examples

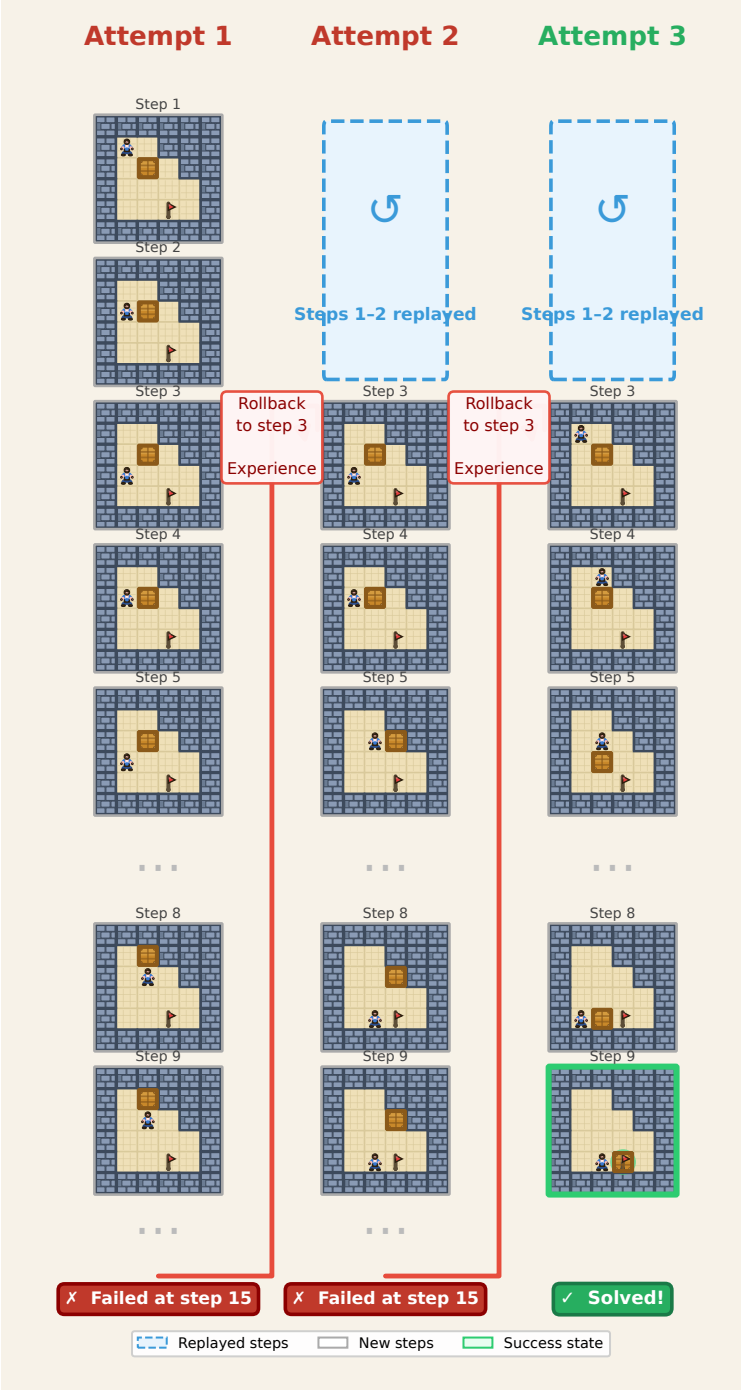


Figure 5: An example on Sokoban illustrating Stage 1 of Counterfactual Distillation. Starting from a failed trajectory, the agent reflects on the interaction history, identifies an earlier suboptimal decision(step 3), and generates a compact experience summary for rollback-based revision. The environment is then reset to the selected step, the prior history is replayed(step 1-2), and a new branch is explored under the guidance of the reflected experience. Repeating this failure $\rightarrow$ reflection $\rightarrow$ rollback $\rightarrow$ correction process enables the agent to recover from early mistakes and eventually reach a successful solution.

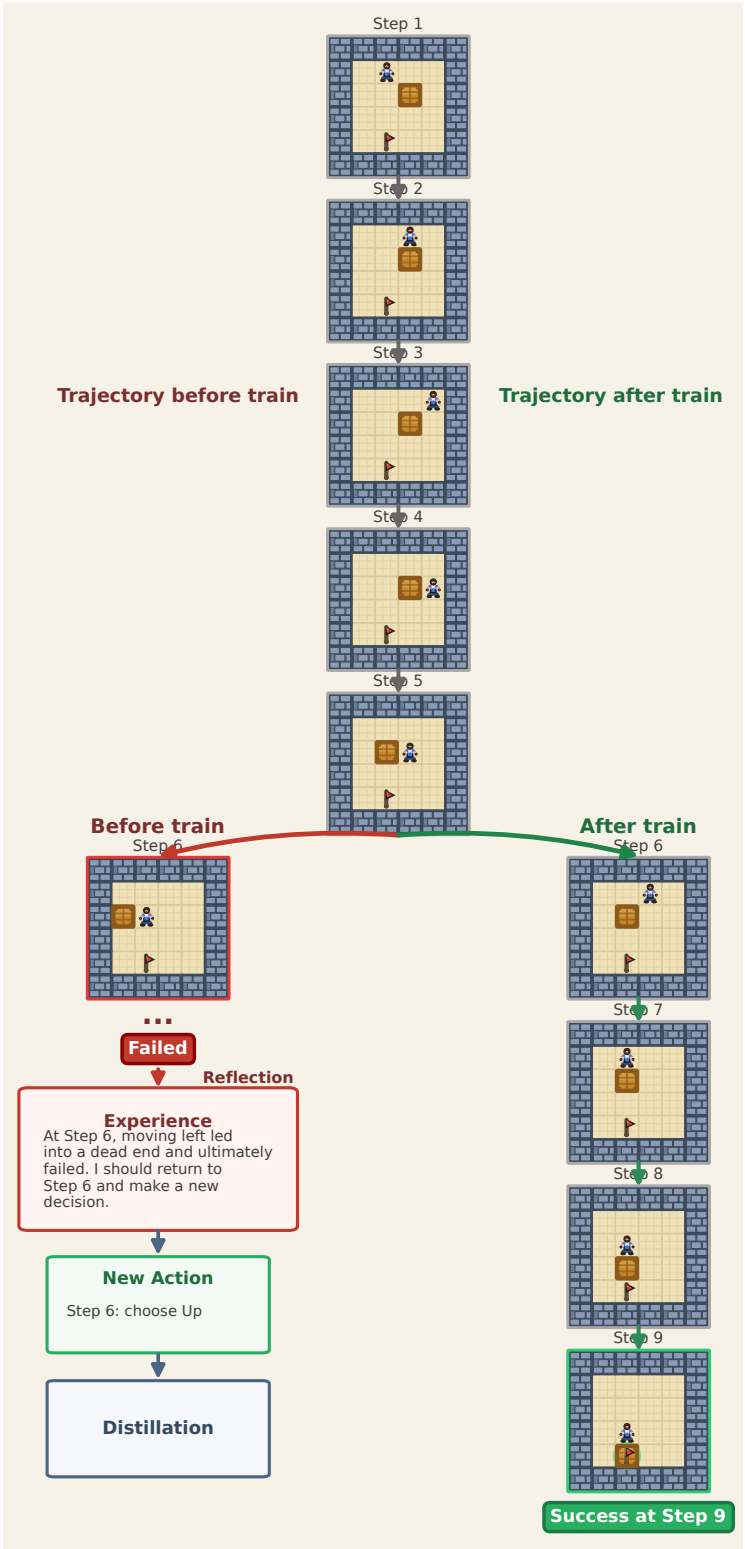


Figure 6: An example showing how Counterfactual Distillation distills experience-guided corrections into the model. Before training, the agent makes a suboptimal move at Step 6 and fails. Reflection identifies the error and suggests an alternative action, which is used as distillation supervision. After training, the model avoids the same mistake even without explicit experience at inference time, producing a successful trajectory at Step 9.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's motivation, scope, and main contributions. These claims are supported by the proposed framework and the empirical results in Sections 3–4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes. The paper includes a dedicated Limitations section that explicitly discusses the limitations of our method.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Yes. Section 3 presents the formal setup, assumptions, notation, and training objectives of our method, providing the theoretical formulation of our proposal.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sections 3–4 and Appendix D describe our method and experimental protocols. Our implementation is available as open-source code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code is openly available, and Section 4 and Appendix D describe the datasets and experimental protocols.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Section 4 and Appendix D describe the training and evaluation settings; our implementation is also available as open-source code.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Appendix D.6 (Tables 14 and 15) reports mean $\pm$ sample SD on four agent benchmarks with Qwen2.5-7B-Instruct, using 32 complete Pass@1 evaluations and 32 Pass@128 subsamples of fixed rollout pools. These SDs quantify evaluation-time variation at fixed checkpoints, rather than variation across training runs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes. We provide the necessary compute resource details in the appendix, including the type of compute workers, memory requirements, and execution time/cost for reproducing the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes. We have reviewed the NeurIPS Code of Ethics, and the research conforms to it; our experiments use standard benchmark environments and do not involve human subjects or private data collection.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: This work is a foundational method study evaluated on standard benchmark environments, and does not collect private data, or release high-risk data or models with direct societal impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work is a foundational method study evaluated on standard benchmark environments, and does not collect private data, or release high-risk data or models with direct societal impact.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the datasets, models, and codebases used in this work and respect their licenses and terms of use. Our implementation is available as open-source code.

    Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release open-source code for our method; its data construction, training, and evaluation procedures are described in Sections 3–4 and Appendix D.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are core experimental agents/policies; no private data; models listed in §4

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.