RESAIL: MITIGATING COLLAPSE IN ITERATIVE AGENT SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Iterative self-distillation enables LLM agents to learn from successive deployments, offering a path toward recursive self-improvement (RSI). Yet our experiments with existing methods reveal a collapse in deployment performance across cycles, while task performance with privileged information (PI) also declines. We address this collapse by prioritizing informative interaction steps for distillation and preserving PI-conditioned behavior as the student becomes the next teacher. We introduce **Re**tentive and **S**elective **A**ugmentation for **I**terative Self-Disti**l**lation (**ReSAIL**), a plug-in augmentation for iterative PI-based self-distillation. ReSAIL selects interaction steps where PI most strongly changes the teacher's predictions and balances the resulting distillation losses across trajectories. It also regularizes the student's PI-conditioned output distributions toward those of the frozen teacher at selected and unselected steps to preserve PI-conditioned behavior for supervision in the next cycle. On ALFWorld and TextCraft, ReSAIL sustains substantial gains across model scales over three cycles, with an average absolute gain of 22.5% in final-cycle success rates when added to self-distillation baselines. Sensitivity-guided selection of offline data also improves action prediction accuracy for multimodal GUI agents on AITZ. These findings provide the first evidence that a more robust learning mechanism can effectively mitigate performance collapse in iterative agent self-distillation over deployment trajectories.

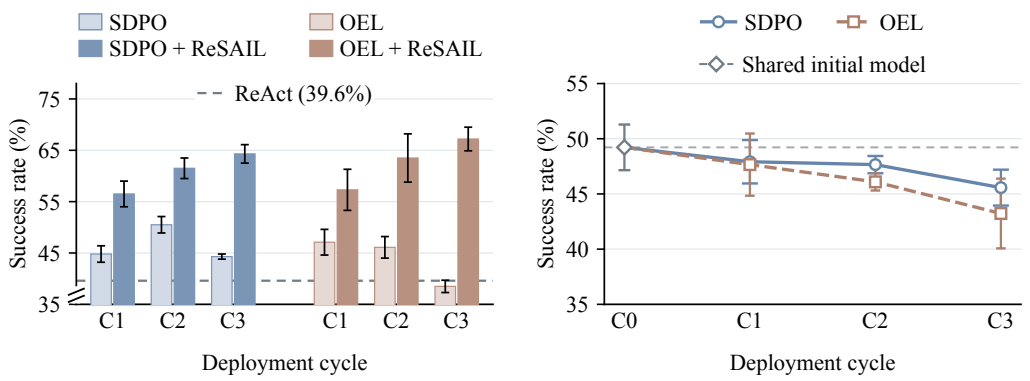
1 INTRODUCTION

The promise of large language model (LLM) agents extends beyond solving complex tasks to improving through experience (Plaat et al., 2025; Chen et al., 2026b). Through multi-step interactions with their environments, agents generate trajectories of actions, observations, and task outcomes (Yao et al., 2023; Wei et al., 2026). These trajectories provide feedback for refining future decisions (Shinn et al., 2023). Learning from this feedback changes agents' behavior and, in turn, the experience they subsequently generate. Iterating this feedback loop offers a path toward recursive self-improvement (RSI) (Ye et al., 2026; Chen et al., 2026b).

Recent work supports this loop by reusing experience as context or incorporating it into model parameters. Reflection and retrieval reuse past experience to guide decisions without updating model parameters (Zhao et al., 2024; Zhang et al., 2026). Self-distillation instead transfers behavior conditioned on training-only *privileged information* (PI) into model parameters (Shenfeld et al., 2026; Hübotter et al., 2026). OEL extends this approach beyond a single training cycle, alternating experience collection and offline self-distillation across successive deployments (Ye et al., 2026). Yet repeating this cycle does not guarantee continued improvement (Chen et al., 2026a). Our experiments with SDPO and OEL show that performance can collapse over successive cycles (Figure 1(a)).

To investigate what may limit continued improvement, we ask two questions about the supervision used in iterative self-distillation. **(Q1) Which interaction steps should be prioritized for distillation?** We hypothesize that interaction steps where PI most strongly changes the teacher's predictions offer particularly informative supervision for transferring PI-conditioned behavior. We quantify this influence using *privileged-information sensitivity* (PI sensitivity), defined as the Jensen–Shannon divergence between the teacher's output distributions with and without PI at the same interaction step. **(Q2) What happens when the student becomes the next teacher?** Each updated model serves two roles: it acts without PI during deployment and provides PI-conditioned supervision in the

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



(a) Task performance

(b) PI-conditioned task competence

Figure 1: **Iterative self-distillation does not guarantee sustained improvement.** On ALFWorld OOD with Qwen3-4B: (a) deployment success rate of SDPO and OEL with and without ReSAIL; (b) success rate of the original methods conditioned on privileged information (PI), with the same tasks and their PI held fixed across cycles. Cycle 0 denotes the shared initial model. Error bars show the standard deviation of success rates across three decoding seeds.

next cycle. The base objective (Equation 2) trains the student only without PI and does not explicitly preserve its PI-conditioned behavior. With evaluation tasks and PI held fixed, both SDPO and OEL show declining mean PI-conditioned task success over successive cycles (Figure 1(b)). These considerations motivate selective distillation within each cycle and preservation of PI-conditioned competence as the updated student becomes the next teacher.

Guided by these considerations, we propose **Re**tentive and **S**elective **A**ugmentation for **I**terative Self-Distil**l**lation (**ReSAIL**), a plug-in augmentation for iterative PI-based self-distillation. The approach combines two complementary modules: **Trajectory-Balanced Selective Distillation** (TBSD) for selective distillation within each cycle and **Privileged Retention** (PR) for retaining PI-conditioned behavior across cycles. TBSD uses Sensitivity-Guided Selection (SGS) to prioritize interaction steps with the highest teacher PI sensitivity and balances the selected losses across trajectories. PR regularizes the student's privileged-view output distributions toward those of the frozen teacher at both selected and unselected interaction steps. This retention objective targets the PI-conditioned behavior used when the updated student becomes the teacher in the next cycle.

Our contributions are summarized as follows: (1) We provide the first evidence that performance collapse in iterative agent self-distillation can be effectively mitigated through a more robust learning mechanism. (2) We propose ReSAIL, a plug-in augmentation that combines trajectory-balanced selective distillation with privileged retention to improve learning within each cycle and preserve PI-conditioned behavior across cycles. (3) Experiments on ALFWorld and TextCraft demonstrate substantial and sustained performance gains across model scales over three deployment cycles. ReSAIL improves final-cycle success rates by an average of 22.5 percentage points over the corresponding SDPO and OEL baselines. We further demonstrate that Sensitivity-Guided Selection improves action prediction accuracy for multimodal GUI agents on AITZ.

2 Related Work

Learning and Adaptation in LLM Agents LLM agents have been widely explored as autonomous systems that combine reasoning, tool use, and interaction with environments to accomplish tasks (Yao et al., 2023; Plaat et al., 2025; Wei et al., 2026). Context-based approaches enable agents to learn from past interactions by reusing reflections and extracted insights to guide subsequent decisions, without updating model parameters (Shinn et al., 2023; Zhao et al., 2024; Zhang et al., 2026; Ouyang et al., 2026). Other approaches instead update model parameters through supervised fine-tuning on interaction trajectories (Dou et al., 2024; Yuan et al., 2025) or reinforcement learning from environmental rewards (Feng et al., 2025; Wang et al., 2025; Xi et al., 2026). Recent work further explores how to reuse accumulated interaction trajectories to support subsequent agent

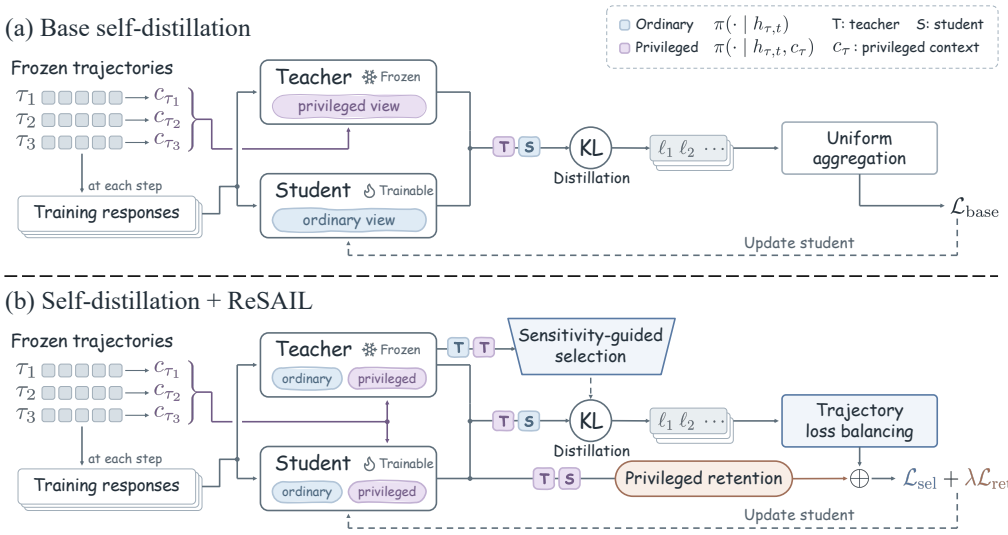


Figure 2: **Overview of ReSAIL**. (a) Base self-distillation matches the ordinary-view student to the privileged-view teacher, uniformly averaging losses over all interaction steps used for training. (b) ReSAIL uses **Sensitivity-Guided Selection** (SGS) to select the most PI-sensitive steps and **Trajectory Loss Balancing** (TLB) to balance the selected losses across trajectories. **Privileged Retention** (PR) regularizes the student's privileged-view predictions toward those of the teacher at both selected and unselected interaction steps within each training batch.

learning and adaptation (Ye et al., 2026; Wu et al., 2026). Within this line of work, we investigate how agent self-distillation can remain effective across successive deployment cycles.

Self-Distillation with Privileged Information Self-distillation with privileged information uses a model's behavior conditioned on additional context to supervise its own learning, without requiring that context at deployment (Shenfeld et al., 2026; Hübotter et al., 2026). Here, privileged information (PI) refers to auxiliary information used during training but omitted from the deployed policy's inputs (Penaloza et al., 2026). PI can include reference solutions or demonstrations (Shenfeld et al., 2026), hindsight feedback (Hübotter et al., 2026), and knowledge extracted from interaction trajectories (Ye et al., 2026; Wang et al., 2026). In each case, the quality of self-distillation supervision depends on how effectively the model can use the supplied PI. We prioritize self-distillation at interaction steps where PI most strongly changes the teacher's predictions.

Iterative Self-Distillation OEL alternates experience collection during deployment with offline self-distillation of knowledge extracted from that experience (Ye et al., 2026). However, repeated self-distillation can degrade agent performance across successive cycles (Chen et al., 2026a). To mitigate this degradation, Chen et al. (2026a) shift to off-policy context-distillation with a forward-KL objective on successful trajectories generated by PI-conditioned teachers. Collecting these teacher trajectories requires additional environment interaction, making this approach inapplicable when the deployment environment cannot be revisited during training. In contrast, ReSAIL addresses this practical and more restrictive setting by strengthening self-distillation on already collected deployment trajectories. It combines selective distillation within each cycle with a retention objective that targets the PI-conditioned behavior used when the updated student becomes the next teacher.

3 METHODOLOGY

In this section, we present **ReSAIL**, a plug-in augmentation for iterative self-distillation with privileged information (PI). We first introduce the offline learning setup and the base self-distillation objective (Section 3.1). We then describe **Trajectory-Balanced Selective Distillation** (Section 3.2) and **Privileged Retention** (Section 3.3), followed by the joint objective and optimization procedure (Section 3.4). Figure 2 provides an overview of ReSAIL.

3.1 SETUP AND BASE SELF-DISTILLATION

In cycle r, a starting policy π_r is paired with a trajectory dataset $\mathcal{D}_r$, collected by π_r or supplied by an external offline source. Training uses this fixed dataset without further environment interaction. Each trajectory is represented as

$$\tau = \big(h_{\tau,t}, \tilde{y}_{\tau,t}\big)_{t=1}^{|\tau|}, \qquad \tau \in \mathcal{D}_r,$$

where $h_{\tau,t}$ is the interaction history at interaction step t and $\tilde{y}_{\tau,t}$ is the logged response.

For each trajectory, the base method supplies PI c_τ, such as the complete trajectory or an extracted experience. A policy π has two views that share parameters:

$$\underbrace{\pi(\cdot \mid h_{\tau,t})}_{\text{ordinary view}} \qquad \text{and} \qquad \underbrace{\pi(\cdot \mid h_{\tau,t}, c_\tau)}_{\text{privileged view}}.$$

For offline training, we initialize the student π^{S} and a frozen teacher π^{T} from π_r. The updated student defines π_{r+1}, acts without PI at deployment, and initializes both models in the next cycle.

During offline training, the base method supplies a training response at each logged step,

$$z_{\tau,t} \sim \mu(\cdot \mid h_{\tau,t}, \tau).$$

In our OEL (Ye et al., 2026) integration, $z_{\tau,t}$ is freshly sampled from the ordinary-view student at each update. In our SDPO-offline (Hübotter et al., 2026) integration, it is generated once by the cycle-start policy and reused throughout the cycle. At each logged step (τ, t), all distributional comparisons use the same prefixes $z_{\tau,t,<k}$ of the training response $z_{\tau,t}$.

For a nonempty trajectory batch $\mathcal{B} \subseteq \mathcal{D}_r$, let $\mathcal{U} \subseteq \{(\tau,t) : \tau \in \mathcal{B},\ 1 \le t \le |\tau|\}$ denote the set of interaction steps used by the base training objective before sensitivity-guided selection. At each step in $\mathcal{U}$, the ordinary-view student matches the privileged-view teacher through token-averaged student-to-teacher KL divergence:

$$\ell_{\mathrm{dist}}(\tau,t) = \frac{1}{|z_{\tau,t}|} \sum_{k=1}^{|z_{\tau,t}|} D_{\mathrm{KL}}\big(\pi^{\mathrm{S}}(\cdot \mid h_{\tau,t}, z_{\tau,t,<k}) \,\big\|\, \pi^{\mathrm{T}}(\cdot \mid h_{\tau,t}, c_\tau, z_{\tau,t,<k})\big)\,. \tag{1}$$

Averaging these losses over steps in $\mathcal{U}$ gives the base objective:

$$\mathcal{L}_{\mathrm{base}} = \frac{1}{|\mathcal{U}|} \sum_{(\tau,t)\in\mathcal{U}} \ell_{\mathrm{dist}}(\tau,t), \qquad \mathcal{U} \neq \emptyset. \tag{2}$$

The influence of PI on the teacher's predictions varies across steps. We therefore concentrate distillation on steps where PI induces the largest prediction changes.

3.2 TRAJECTORY-BALANCED SELECTIVE DISTILLATION (TBSD)

Sensitivity-Guided Selection (SGS). We quantify this prediction shift using PI sensitivity. For any policy π, history h, PI c, and response z, we define *PI sensitivity* as the average divergence between its ordinary and privileged views:

$$s_\pi(h, c; z) = \frac{1}{|z|} \sum_{k=1}^{|z|} \mathrm{JSD}(\pi(\cdot \mid h, z_{<k}), \pi(\cdot \mid h, c, z_{<k}))\,, \tag{3}$$

where JSD denotes the Jensen–Shannon divergence. We score each step in $\mathcal{U}$ using the frozen teacher's PI sensitivity, $s(\tau,t) := s_{\pi^{\mathrm{T}}}(h_{\tau,t}, c_\tau; z_{\tau,t})$. Because both views use the same model parameters and response prefixes, the score measures the prediction change induced by PI. For a selection ratio $\rho \in (0,1]$, we retain the highest-scoring steps across the entire batch:

$$K_\rho = \lceil \rho|\mathcal{U}| \rceil\,, \qquad \mathcal{C}_\rho = \mathrm{TopK}(\mathcal{U}; s, K_\rho). \tag{4}$$

Trajectory Loss Balancing (TLB). A uniform average over selected steps gives greater total weight to trajectories with more selected steps. We therefore average the selected losses within each trajectory and normalize their sum by the full trajectory-batch size. Let $\mathcal{C}_\rho(\tau) = \{t : (\tau, t) \in \mathcal{C}_\rho\}$ and $m_\tau = |\mathcal{C}_\rho(\tau)|$. The selected distillation objective is

$$\mathcal{L}_{\text{sel}} = \frac{1}{|\mathcal{B}|} \sum_{\substack{\tau \in \mathcal{B} \\ m_\tau > 0}} \frac{1}{m_\tau} \sum_{t \in \mathcal{C}_\rho(\tau)} \ell_{\text{dist}}(\tau, t). \tag{5}$$

Each trajectory with selected steps receives total loss coefficient $1/|\mathcal{B}|$; the others contribute zero.

3.3 Privileged Retention (PR)

The updated student also serves as the next cycle's PI-conditioned teacher. Because its two views share parameters, ordinary-view distillation can alter the PI-conditioned behavior used for future supervision. To retain this behavior for the next cycle, PR matches the student's privileged-view output distributions to those of the current frozen teacher:

$$\ell_{\text{ret}}(\tau, t) = \frac{1}{|z_{\tau,t}|} \sum_{k=1}^{|z_{\tau,t}|} D_{\text{KL}}\big(\pi^{\text{S}}(\cdot \mid h_{\tau,t}, c_\tau, z_{\tau,t,<k}) \,\big\|\, \pi^{\text{T}}(\cdot \mid h_{\tau,t}, c_\tau, z_{\tau,t,<k})\big). \tag{6}$$

Retention covers all steps in $\mathcal{U}$, including those not selected for distillation, using the same trajectory-level reduction as TLB. With $\mathcal{U}(\tau) = \{t : (\tau, t) \in \mathcal{U}\}$,

$$\mathcal{L}_{\text{ret}} = \frac{1}{|\mathcal{B}|} \sum_{\substack{\tau \in \mathcal{B} \\ |\mathcal{U}(\tau)| > 0}} \frac{1}{|\mathcal{U}(\tau)|} \sum_{t \in \mathcal{U}(\tau)} \ell_{\text{ret}}(\tau, t). \tag{7}$$

3.4 Joint Objective and Optimization

We combine selective distillation and privileged retention as

$$\mathcal{L} = \mathcal{L}_{\text{sel}} + \lambda \mathcal{L}_{\text{ret}}, \qquad \lambda \geq 0, \tag{8}$$

where λ controls the retention strength. At each update, we recompute the teacher's sensitivity scores and selected set, holding both fixed during backpropagation. Algorithm 1 in Appendix B details one internalization cycle, including tie-breaking and empty-set handling.

4 Experiments

4.1 Experimental Setup

Benchmarks and Models. We evaluate ReSAIL on ALFWorld (Shridhar et al., 2021) for embodied household task planning and TextCraft (Prasad et al., 2024) for compositional crafting. AITZ (Zhang et al., 2024) tests SGS as an offline data filter for Android GUI navigation. We use Qwen3-4B and Qwen3-8B (Yang et al., 2025) with thinking mode disabled for the text-based benchmarks, and Qwen3-VL-4B-Instruct (Bai et al., 2025) for AITZ.

Baselines. On the text benchmarks, we compare against ReAct (Yao et al., 2023), RFT (Yuan et al., 2023), offline GRPO (Shao et al., 2024), EPD (Gou et al., 2026), offline SDPO (Hübotter et al., 2026), and OEL (Ye et al., 2026). All text-task PI methods use experience summaries from their own complete trajectories. We evaluate ReSAIL as a plug-in to SDPO and OEL on text tasks. On AITZ, we compare OEL with OEL + SGS, using SGS only to filter training data before optimization. Implementation details are provided in Appendix B.3.

Training Protocol. Text agents undergo three cycles of 30 updates, each using fresh trajectories and initializing from their own preceding final checkpoint. Update counts and trajectory batch sizes are matched within each text setting. AITZ uses one cycle of 100 updates with 16 source trajectories per update for both methods. OEL + SGS retains a fixed global top 80% of interaction steps before optimization. Data preparation and implementation details are provided in Appendices A and B.

Table 1: **Main results across three deployment cycles.** Task success rates (%) without PI, reported as mean (SD) over three decoding seeds. SD denotes standard deviation. Shaded rows add ReSAIL to the preceding baseline; bold marks the highest mean in each column within each model size. ReAct is the unchanged initial policy, shown once under Cycle 1 as a fixed reference across cycles.

Model	Method	ALFWorld (ID)			ALFWorld (OOD)			TextCraft		
		Cycle 1	Cycle 2	Cycle 3	Cycle 1	Cycle 2	Cycle 3	Cycle 1	Cycle 2	Cycle 3
Qwen3-4B	ReAct	40.1 (2.3)	–	–	39.6 (1.8)	–	–	61.3 (1.5)	–	–
	RFT	43.2 (2.7)	44.8 (4.4)	41.9 (6.6)	39.3 (4.6)	42.2 (1.4)	43.2 (3.9)	59.7 (0.6)	62.3 (5.5)	60.7 (2.5)
	GRPO	39.1 (2.1)	36.5 (1.2)	37.5 (2.3)	45.1 (1.2)	42.2 (1.6)	50.3 (2.0)	60.3 (1.2)	58.7 (2.3)	56.7 (2.3)
	EPD	45.6 (1.2)	36.7 (3.4)	22.1 (0.5)	43.2 (2.7)	31.5 (3.2)	18.2 (3.5)	63.3 (0.6)	62.0 (2.6)	62.3 (3.5)
	SDPO	52.3 (1.6)	53.4 (2.0)	58.9 (1.6)	44.8 (1.6)	50.5 (1.6)	44.3 (0.5)	62.7 (2.3)	64.7 (0.6)	60.3 (4.5)
	+ ReSAIL	**66.4 (2.1)**	**72.1 (1.6)**	72.4 (0.9)	56.5 (2.5)	61.5 (2.0)	64.3 (1.8)	**64.7 (1.2)**	**66.3 (2.9)**	**67.3 (1.2)**
	OEL	54.9 (4.7)	56.5 (2.0)	47.4 (1.2)	47.1 (2.5)	46.1 (2.1)	38.5 (1.2)	62.0 (3.6)	54.7 (1.5)	43.7 (1.5)
	+ ReSAIL	62.2 (2.0)	70.1 (3.2)	**74.2 (1.4)**	**57.3 (4.0)**	**63.5 (4.7)**	**67.2 (2.3)**	63.3 (1.5)	64.0 (3.6)	64.0 (2.6)
Qwen3-8B	ReAct	49.5 (2.4)	–	–	57.0 (3.9)	–	–	64.0 (1.0)	–	–
	RFT	50.0 (2.8)	52.1 (2.5)	52.1 (2.7)	60.4 (1.2)	61.2 (1.8)	63.8 (5.3)	66.0 (4.4)	62.7 (2.5)	58.7 (2.1)
	GRPO	53.1 (4.9)	47.4 (1.2)	48.2 (2.5)	62.8 (5.0)	65.4 (3.5)	62.2 (1.2)	57.0 (2.6)	64.3 (2.1)	59.3 (2.1)
	EPD	68.0 (2.3)	67.2 (2.8)	53.4 (2.7)	62.0 (1.6)	56.0 (1.2)	46.6 (3.9)	65.7 (2.3)	65.0 (2.6)	64.3 (0.6)
	SDPO	71.4 (1.6)	72.1 (1.2)	61.2 (3.9)	66.1 (2.7)	64.1 (4.1)	49.2 (2.8)	63.0 (1.0)	62.3 (2.3)	51.0 (2.0)
	+ ReSAIL	**75.5 (1.8)**	**80.5 (1.4)**	82.0 (1.4)	**72.1 (5.2)**	73.2 (0.9)	75.3 (1.2)	65.7 (2.3)	70.7 (3.1)	73.3 (1.5)
	OEL	72.4 (0.5)	73.4 (3.9)	58.3 (0.9)	66.7 (2.7)	61.7 (5.5)	41.4 (1.4)	63.0 (2.6)	61.3 (1.5)	51.3 (0.6)
	+ ReSAIL	73.7 (2.7)	79.2 (1.2)	**82.8 (1.6)**	70.8 (3.5)	**73.4 (2.3)**	**78.4 (1.2)**	**69.0 (2.6)**	**72.3 (1.2)**	**73.7 (1.5)**

Evaluation. We evaluate final checkpoints after each text-task cycle using task success rate on ALFWorld (ID/OOD) and TextCraft. ReAct is the unchanged initial policy for text tasks. These final results report means and standard deviations (SD) across three decoding seeds. On AITZ, we monitor offline step-level accuracy every ten updates on a fixed test subset, excluding reference `stop` steps and following the official action-matching rules. Detailed evaluation settings are provided in Appendix A.3. For the retention analysis, we evaluate all checkpoints on the same 128 ALFWorld OOD tasks with matched initial states. The task-specific PI consists of summaries generated by the initial model from separate reference episodes and is held fixed across evaluations (Appendix D.2).

4.2 Main Results

Higher Final-Cycle Performance. After three deployment cycles, both SDPO+ReSAIL and OEL+ReSAIL outperform all unaugmented baselines across ALFWorld ID/OOD and TextCraft at both model sizes (Table 1). Relative to their respective parent methods, ReSAIL improves final-cycle success by 7.0–26.1 percentage points for SDPO and 20.3–37.0 points for OEL. The gains also hold against the strongest alternative baseline. On ALFWorld OOD with Qwen3-8B, SDPO+ReSAIL and OEL+ReSAIL achieve 75.3% and 78.4% success, respectively. The corresponding success rates are 63.8% for RFT, 62.2% for GRPO, and 46.6% for EPD.

Sustained Gains Across Deployment Cycles. Both augmented methods outperform their respective parent methods at every reported cycle and maintain non-decreasing mean success across all three cycles in every evaluated setting. By contrast, OEL finishes below its first-cycle performance in all six settings, and SDPO does so in five. The benefit of ReSAIL therefore extends beyond an initial improvement. Averaged equally across the six model–benchmark settings, the advantage over the corresponding parent method grows from 6.8 to 18.3 percentage points for SDPO and from 5.0 to 26.6 points for OEL between the first and third deployment cycles.

Table 2: **Component ablations** with OEL and Qwen3-4B on ALFWorld. ✓ and × indicate enabled and disabled components; shading marks the full method. Entries report success rates (%) as mean (SD) over three decoding seeds. Bold marks the highest mean in each column. SGS: Sensitivity-Guided Selection; TLB: Trajectory Loss Balancing; PR: Privileged Retention.

Components			ALFWorld (ID)			ALFWorld (OOD)		
SGS	TLB	PR	Cycle 1	Cycle 2	Cycle 3	Cycle 1	Cycle 2	Cycle 3
×	×	×	54.9 (4.7)	56.5 (2.0)	47.4 (1.2)	47.1 (2.5)	46.1 (2.1)	38.5 (1.2)
✓	×	×	**66.4 (4.8)**	69.0 (4.6)	50.5 (3.2)	**61.5 (5.9)**	56.5 (2.5)	37.8 (2.5)
×	✓	×	56.5 (1.6)	60.2 (2.1)	59.9 (2.4)	51.3 (1.6)	50.8 (2.7)	50.8 (3.1)
×	×	✓	52.9 (4.4)	55.2 (0.9)	64.3 (1.6)	49.0 (2.0)	49.7 (1.6)	53.9 (2.1)
✓	✓	×	65.9 (3.0)	68.5 (0.5)	56.5 (3.0)	60.4 (3.0)	58.9 (1.6)	45.6 (1.6)
✓	✓	✓	62.2 (2.0)	**70.1 (3.2)**	**74.2 (1.4)**	57.3 (4.0)	**63.5 (4.7)**	**67.2 (2.3)**

Table 3: **Selection strategy and retention view.** OEL with Qwen3-4B on ALFWorld. (a) Cycle 1 selection at $\rho = 0.05$, fixing TLB and PR. Bottom and Top select the least and most PI-sensitive interaction steps. (b) Cycle 3 retention, fixing SGS and TLB. Entries report success without PI (%) as mean (SD) over three decoding seeds; bold marks the highest mean in each column.

(a) Selection strategy $r = 1$

Selection	ID	OOD
Random	49.7 (3.9)	45.8 (4.4)
Bottom	40.1 (3.2)	40.4 (4.3)
Top (ours)	**62.2 (2.0)**	**57.3 (4.0)**

(b) Retention view $r = 3$

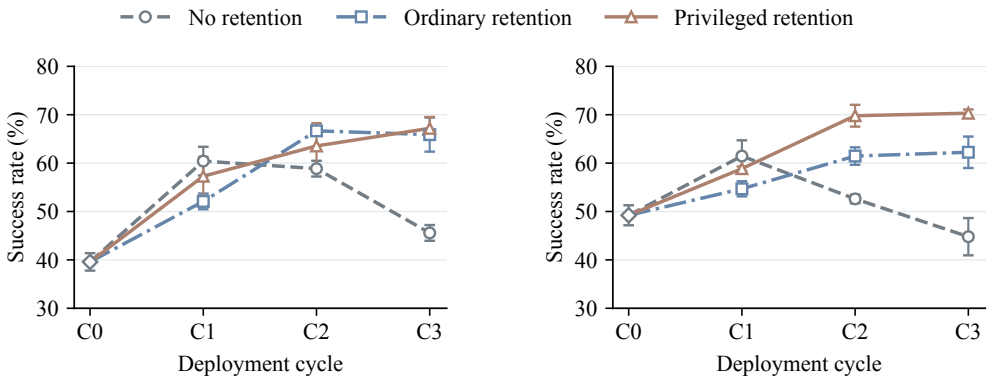
Retention view	ID	OOD
None	56.5 (3.0)	45.6 (1.6)
Ordinary	72.9 (2.4)	65.9 (3.5)
Privileged (ours)	**74.2 (1.4)**	**67.2 (2.3)**

4.3 Ablation Studies

Selection and Retention Contribute at Different Timescales. The component ablation shows complementary roles for selective distillation and privileged-view retention (Table 2). SGS alone achieves the strongest first-cycle performance, improving over OEL by 11.5 and 14.4 percentage points on ID and OOD, respectively. Adding TLB improves third-cycle success over selection alone by 6.0 and 7.8 points, but the resulting TBSD configuration still finishes below its first-cycle performance. Adding PR changes this cross-cycle trajectory: the full method achieves the highest success in Cycles 2 and 3, exceeding TBSD by 17.7 and 21.6 points at Cycle 3. The full method also outperforms PR alone by 9.9 and 13.3 points, showing that selective distillation remains valuable when training already includes a privileged-view retention objective.

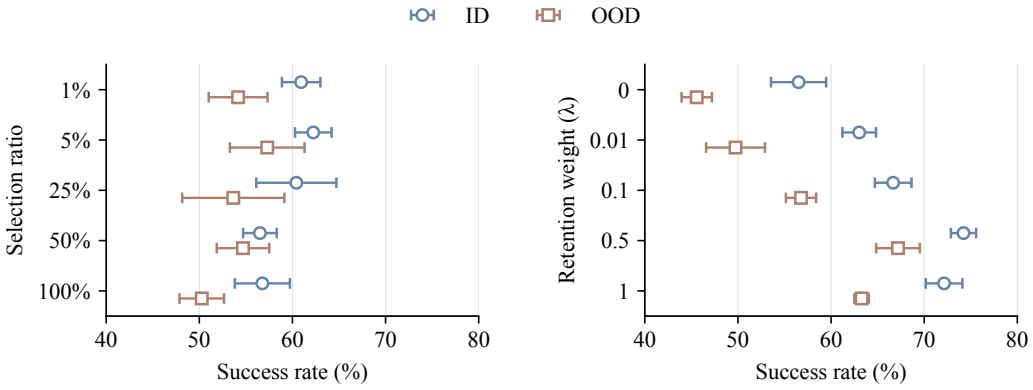
PI Sensitivity Identifies Useful Interaction Steps. To test whether the selection criterion matters beyond sparsity, we compare Random, Bottom, and Top selection at the same ratio $\rho = 0.05$, with TLB and PR fixed. Top selection, corresponding to SGS, improves first-cycle success over Random by 12.5 and 11.5 percentage points on ID and OOD, respectively, while Bottom selection performs worse than Random on both splits (Table 3(a)). Thus, the benefit depends on prioritizing PI-sensitive steps, rather than selecting an arbitrary subset of the same size.

Retention Improves Repeated-Deployment Performance. We compare no retention, ordinary-view retention, and privileged-view retention while fixing SGS and TLB. The two retention objectives use the same set of interaction steps, averaging scheme, and coefficient. Both improve third-cycle deployment success substantially over no retention, with privileged-view retention achieving the highest results: 74.2% on ID and 67.2% on OOD (Table 3(b)). We next examine whether retaining the privileged view also better preserves the PI-conditioned competence needed for the updated student to supervise learning in subsequent self-distillation cycles.



(a) Ordinary view (b) Privileged view

Figure 3: **Privileged retention sustains PI-conditioned task competence.** OEL+TBSD with Qwen3-4B on ALFWorld OOD tasks, evaluated (a) without PI and (b) with PI. Task initial states and the PI in (b) are fixed across checkpoints and retention variants. C0 denotes the shared initial model. Points show means; error bars show ± 1 SD over three decoding seeds.

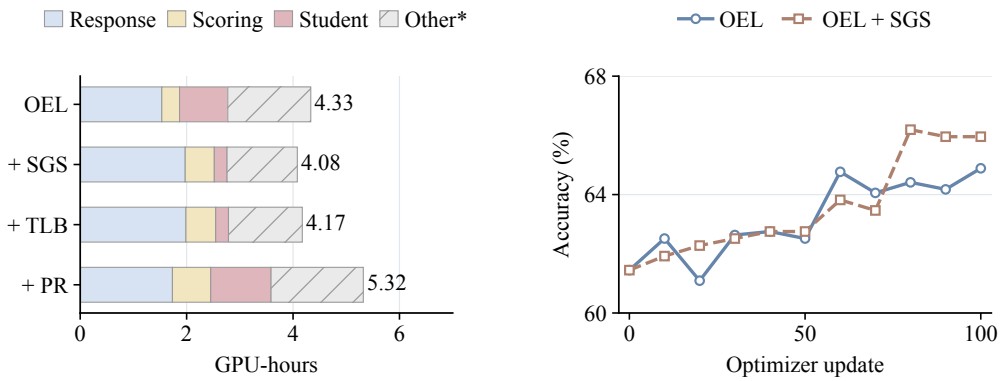


(a) Selection ratio (Cycle 1) (b) Retention weight (Cycle 3)

Figure 4: **Selection ratio and retention weight.** OEL+ReSAIL with Qwen3-4B on ALFWorld. (a) Cycle 1 success at each selection ratio ρ, fixing $\lambda = 0.5$. (b) Cycle 3 success at each retention weight λ, fixing $\rho = 0.05$; each weight remains constant across cycles. TLB is enabled in both sweeps. Points show means; horizontal error bars show ± 1 SD over three decoding seeds.

Privileged Retention Preserves PI-Conditioned Competence. Without retention, PI-conditioned success falls from 61.5% at Cycle 1 to 44.8% at Cycle 3. With PR, it instead increases from 58.9% to 70.3% (Figure 3). Ordinary-view retention reaches 62.2% at Cycle 3, leaving an 8.1-percentage-point advantage for PR under PI, compared with a 1.3-point advantage without PI. The larger separation in PI-conditioned performance supports retaining the view in which the model provides supervision for subsequent cycles.

Selection and Retention Hyperparameters. We evaluate the selection ratio after one cycle to measure within-cycle learning, and the retention weight after three cycles to measure its effect under repeated deployment. With TLB and PR enabled and $\lambda = 0.5$, selecting the top 1%, 5%, or 25% of steps outperforms using all steps in $\mathcal{U}$ ($\rho = 1$) on both splits (Figure 4(a)). The default $\rho = 0.05$ achieves the strongest first-cycle results, improving over the $\rho = 1$ configuration by 5.4 and 7.0 percentage points on ID and OOD. With $\rho = 0.05$, third-cycle success increases as λ rises from 0 to 0.5, while $\lambda = 1$ yields lower performance (Figure 4(b)). These sweeps favor selective distillation with a moderate retention weight, supporting the default settings used in our experiments.



(a) Training cost (b) AITZ accuracy

Figure 5: **Training cost and AITZ accuracy.** (a) Approximate Cycle 1 training-loop costs on ALFWorld using eight H800 GPUs. Components are added cumulatively to OEL; the final bar is OEL+ReSAIL. Other* denotes residual training-loop cost. (b) AITZ step-level action accuracy on a fixed subset. OEL+SGS retains a fixed top-80% training subset.

4.4 Efficiency and Generalization

Training Cost–Performance Tradeoff. For Qwen3-4B on ALFWorld, SGS improves the Cycle 1 training-loop cost–performance tradeoff (Figure 5(a)). It reduces student optimization cost by approximately 74% and total training-loop cost from 4.33 to 4.08 GPU-hours, while improving first-cycle task success. The total training-loop reduction is approximately 6%. Response generation and sensitivity scoring account for most of the remaining cost. Adding TLB increases total cost only slightly, from 4.08 to 4.17 GPU-hours, while student optimization remains at 0.24 GPU-hours. The full ReSAIL configuration additionally retains privileged-view predictions at all interaction steps in $\mathcal{U}$, bringing the measured cost to 5.32 GPU-hours, or 1.23 times that of OEL. Thus, selection offers a lower-cost improvement, while the complete method preserves PI-conditioned competence to sustain task performance over multiple deployment cycles.

PI Sensitivity as an Offline Multimodal Data Filter. We further apply SGS to an existing dataset of multimodal agent trajectories on AITZ. Using PI sensitivity computed by the initial model, we retain the global top 80% of interaction steps and keep this subset fixed throughout training. Both methods share the same initialization, 100-update schedule, and OEL objective.

OEL+SGS achieves higher action accuracy at the final three evaluations and finishes at 65.95%, compared with 64.89% for OEL. This result extends PI-sensitive selection beyond iterative text-agent training to static offline filtering for a multimodal agent, improving final action accuracy by 1.06 percentage points. These gains are obtained with the same update budget and a subset fixed before training, supporting PI sensitivity as a criterion for filtering offline agent trajectories.

5 Conclusion

We introduced ReSAIL, a plug-in augmentation for iterative experience-based self-distillation. It combines PI-sensitive selection and trajectory-balanced distillation with privileged retention to improve learning within each cycle and preserve PI-conditioned behavior across cycles. Experiments on ALFWorld and TextCraft show that ReSAIL improves SDPO and OEL across model scales and sustains gains over three deployment cycles. Ablations and evaluations with fixed privileged information support the complementary roles of selection and retention. Sensitivity-guided offline filtering also improves action prediction for multimodal agents on AITZ. These findings highlight the importance of improving the deployed policy while preserving the PI-conditioned competence it will use as the teacher in subsequent self-distillation cycles.

AI USE STATEMENT

We used LLM to assist with manuscript polishing, literature retrieval and discovery, and code implementation. The authors take full responsibility for the final manuscript and for the methods, analyses, and experimental results presented in this work.

ETHICS STATEMENT

This work uses publicly available benchmarks and trajectory datasets. To the best of our knowledge, the experiments reported in this paper raise no additional ethical concerns.

REPRODUCIBILITY STATEMENT

Data preparation and evaluation protocols are detailed in Appendix A. Appendix B provides training configurations, baseline adaptations, and pseudocode; Appendix C reports detailed experimental results. Full prompt templates are provided in Appendix E.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Jingwen Chen, Wenkai Yang, Shengda Fan, Wenbo Nie, Chenxing Sun, Shaodong Zheng, Yangen Hu, Lu Pan, Ke Zeng, and Yankai Lin. Rethinking continual experience internalization for self-evolving llm agents. *arXiv preprint arXiv:2606.04703*, 2026a.

Mingguang Chen, Licheng Wang, and Bo Qu. Recursive self-improvement in ai: From bounded self-refinement to autonomous research loops, 2026b. URL https://arxiv.org/abs/2607.07663.

Zi-Yi Dou, Cheng-Fu Yang, Xueqing Wu, Kai-Wei Chang, and Nanyun Peng. Re-rest: Reflection-reinforced self-training for language agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15394–15411, 2024.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 46375–46408. Curran Associates, Inc., 2025. doi: 10.52202/085713-1544. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/420c9f777c0b4f78d515e53cf74d58b2-Paper-Conference.pdf.

Chenhui Gou, Haoqin Tu, Yunhao Fang, Jianfei Cai, and Hamid Rezatofighi. Sample-efficient learning from agent experience. *arXiv preprint arXiv:2607.21051*, 2026.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Siru Ouyang, Jun Yan, I-Hung 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, 2026. URL https://arxiv.org/abs/2509.25140.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models, 2026. URL https://arxiv.org/abs/2602.04942.

Aske Plaat, Max van Duijn, Niki van Stein, Mike Preuss, Peter van der Putten, and Kees Joost Batenburg. Agentic large language models, a survey, 2025. URL https://arxiv.org/abs/2503.23037.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. ADaPT: As-needed decomposition and planning with language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 4226–4252, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.264. URL https://aclanthology.org/2024.findings-naacl.264/.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

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. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03768.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-sd: Skill-conditioned self-distillation for multi-turn llm agents, 2026. URL https://arxiv.org/abs/2604.10674.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, et al. Agentic reasoning for large language models, 2026. URL https://arxiv.org/abs/2601.12538.

Jie Wu, Zhenru Zhang, Beichen Zhang, Xuwu Wang, Yuhui Su, Mouxiang Chen, Peng Wang, Zhihai Wang, Que Shen, Hao Zhou, et al. Terminal-universe: Turning agent trajectories into scalable terminal environments, 2026. URL https://arxiv.org/abs/2609.04148.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Jiaqi Liu, Honglin Guo, yajie yang, Rui Zheng, Junjie Ye, Jiazheng Zhang, et al. Agentgym-rl: An open-source framework to train llm agents for long-horizon decision making via multi-turn rl. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 12685–12727, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1571ce1ff735be4551db50887043726e-Paper-Conference.pdf.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous GUI interaction. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 69772–69805. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/xu25ae.html.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Tianzhu Ye, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Online experiential learning for language models. *arXiv preprint arXiv:2603.16856*, 2026.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training, 2025. URL https://arxiv.org/abs/2501.11425.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023. URL https://arxiv.org/abs/2308.01825.

Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for GUI agents. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12016–12031, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.702. URL https://aclanthology.org/2024.findings-emnlp.702/.

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, 2026. URL https://arxiv.org/abs/2510.04618.

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

APPENDIX CONTENTS

A Experimental Setup **14**
A.1 Benchmarks and Data Splits 14
A.2 Training Data and Cycles 14
A.3 Evaluation and Reporting 14

B Implementation and Baseline Details **15**
B.1 Inputs and Experience Construction 15
B.2 Selection, Retention, and Optimization 15
B.3 Baseline Adaptations 16

C Detailed Experimental Results **18**
C.1 Hyperparameter Ablations 18
C.2 AITZ Training Dynamics 19
C.3 Computational Cost 19

D Analysis and Discussion **20**
D.1 Privileged Retention versus Reference KL Regularization 20
D.2 Dual-View Evaluation Details 20
D.3 PI-Sensitivity Measurement and Dynamics 21

E Prompt Templates **23**
E.1 Text Agent Prompts 23
E.2 Text Experience Extraction 27
E.3 GUI Agent Prompt 29
E.4 GUI Experience Extraction 31

A Experimental Setup

A.1 Benchmarks and Data Splits

Text experiments use Qwen3-4B and Qwen3-8B with thinking disabled; AITZ uses Qwen3-VL-4B-Instruct. Table 4 summarizes the test sets, held fixed across methods and cycles.

Table 4: Evaluation scope per decoding seed.

Benchmark	Test set	Metrics
ALFWorld	128 ID and 128 OOD episodes	Success rate
TextCraft	Full official test set: 100 episodes	Success rate
AITZ	Fixed monitoring subset: 101 episodes, 843 non-`stop` steps	Accuracy

A.2 Training Data and Cycles

Text agents use three cycles with the budgets in Table 5, carrying forward final model parameters and resetting optimizer and random states. Each cycle samples task IDs without replacement and collects one trajectory per task; recurring tasks receive fresh trajectories. Methods of the same model size share the Cycle 1 corpus and summaries; later cycles use each method's own preceding final checkpoint to collect a fresh corpus, replacing the previous one.

AITZ uses 1,728 externally collected, complete CoAT training trajectories from Aguvis Stage-2 (Xu et al., 2025), each with at most 20 steps and no unsupported actions. We use their screenshots and logged actions, without the original GPT-generated semantic annotations. A fixed seed samples and orders 1,600 trajectories without replacement, shared by all training runs. The initial policy prepares diagnostic attempts and trajectory summaries before optimization. OEL and OEL + SGS follow the same single-cycle, 100-update schedule, with 16 source trajectories per update. The filtered run retains a fixed global top 80% of steps before training. Only retained steps contribute to its loss; their original histories and summaries remain available.

A.3 Evaluation and Reporting

Text tasks evaluate each cycle's final checkpoint, without intermediate validation or checkpoint selection; ReAct is the unchanged initial policy. Success rate uses a 30-step interaction limit. Invalid actions consume a step, and the agent continues from environment feedback without regenerating the response. ALFWorld tasks and initial states are fixed across decoding seeds. Final results for the text benchmarks report mean and sample standard deviation across three decoding seeds, measuring decoding variability.

For AITZ, we monitor both methods on a fixed random subset of 101 test episodes every ten updates using one fixed decoding seed. Monitoring does not guide hyperparameter changes, early stopping, or checkpoint selection.

Within the fixed monitoring subset, AITZ evaluation excludes steps whose reference action is `stop`. At each remaining logged step, the agent receives the task instruction, current screenshot, and ground-truth prior actions. Predictions are scored against reference actions without execution or insertion into later histories. Exact-match (EM) accuracy follows the official screen-wise rule for the category and its associated arguments: target, scroll direction, text, or control command, as applicable. Matching follows the scorer's criteria rather than literal equality of raw arguments. Accuracy averages over all non-`stop` evaluation steps; invalid predictions score zero. Complete inputs are in Appendix E.

Evaluation uses temperature 0.4, top-$p = 1.0$, and no top-k truncation ($k = -1$). Generation is capped at 1,024 tokens for ALFWorld and 512 for TextCraft and AITZ. Generation settings and the seeds for each evaluation protocol are matched across methods and checkpoints.

B Implementation and Baseline Details

This section specifies experience construction, the optimization procedure, and baseline adaptations under the protocol in Appendix A.

Training infrastructure. Our training framework is built on slime,[1] with Megatron-LM[2] for training and SGLang[3] for generation. We use Python 3.12.3 and PyTorch 2.11.0 with CUDA 12.9. Model parameters use BF16, while optimizer states and main gradients use FP32. Megatron uses data parallelism of degree 8 and tensor, pipeline, and context parallelism of degree 1. Each SGLang engine uses tensor parallelism of degree 1.

B.1 Inputs and Experience Construction

The ordinary input contains the task and interaction history. PI adds a summary c_τ extracted from the associated complete trajectory. Text-task extraction uses a task snapshot, executed actions, resulting observations, and the trajectory outcome; AITZ additionally uses the cycle-initial student's diagnostic attempts. The trajectory itself is not appended to the teacher input.

Each PI-based method uses its frozen starting policy to extract one accepted summary per trajectory, paired with its source histories and cached throughout training. Text-task summaries refresh each cycle; AITZ prepares them once. Extraction rules and templates are shared, although different policies may produce different summaries. Appendices A.2 and E specify data sharing and complete templates. Deployment evaluation uses ordinary input.

B.2 Selection, Retention, and Optimization

Algorithm 1 summarizes the full text-task integration using the objectives in Section 3.4. Selection operates over the full update batch, with a fixed tie order and no trajectory-coverage completion. Scoring and both losses use the same response prefixes and valid-token masks; responses, scores, and selected indices are fixed during differentiation. We compute teacher JSD and both distillation and retention KL using a Top-20-plus-residual-tail approximation. At each response prefix, the current ordinary-view student selects the top 20 token IDs; all compared distributions use these same IDs and pool their remaining probability mass into one tail category.

Distillation covers selected interaction steps; retention covers all steps in $\mathcal{U}$. Each loss averages within trajectories and then over the full sampled batch, with empty supports contributing zero. We neither renormalize over surviving trajectories nor adjust λ with coverage. Thus, $\rho = 1$ need not recover the baseline's uniform average over steps.

All trained methods use AdamW with constant learning rate 10^{-6} and no warmup. Table 5 gives the trajectory budgets and ReSAIL hyperparameters.

Table 5: Training settings per cycle (one cycle for AITZ, three for text tasks). Batch size counts source trajectories before step filtering; ρ and λ denote selection ratio and retention weight.

Benchmark	Updates	Batch size	Trajectories	ρ	λ
ALFWorld	30	32	960	0.05	0.5
TextCraft	30	8	240	0.25	1.0
AITZ (OEL)	100	16	1,600	–	–
AITZ (OEL + SGS)	100	16	1,600	0.80	–

Offline filtering on AITZ. We score initial-policy ordinary responses using Equation 3 and retain the fixed global top 80% of steps for OEL + SGS. The filtered run preserves the OEL loss, without TLB or PR. OEL (Ye et al., 2026) distills the frozen privileged teacher along fresh ordinary-student responses at each update, without changing this selection.

[1] https://github.com/THUDM/slime

[2] https://github.com/NVIDIA/Megatron-LM

[3] https://github.com/sgl-project/sglang

Algorithm 1 One internalization cycle of ReSAIL

Input: Trajectories $\mathcal{D}_r$, starting policy π_r, selection ratio ρ, retention weight λ, updates M
Output: Final student π_{r+1}, evaluated with ordinary input.
Initialize student π^{S} and teacher π^{T} from π_r; freeze teacher
Prepare per-trajectory summaries c_τ using the base-method interface
for $u = 1, \ldots, M$ **do**
  Take the next trajectory batch $\mathcal{B}$ from the cycle's data schedule
  Obtain base-method responses $z_{\tau,t}$; form $\mathcal{U}$
  Initialize $\mathcal{C}_\rho \leftarrow \emptyset$, $\mathcal{L}_{\mathrm{sel}} \leftarrow 0$, $\mathcal{L}_{\mathrm{ret}} \leftarrow 0$
  if $\mathcal{U} \neq \emptyset$ **then**
    Score $\mathcal{U}$ with frozen-teacher JSD (Equation 3)
    $\mathcal{C}_\rho \leftarrow \mathrm{TopK}(\mathcal{U}; s, \lceil \rho |\mathcal{U}| \rceil)$, using fixed tie order
    Compute $\mathcal{L}_{\mathrm{sel}}$ via Equation 5, applying base-method token corrections before reduction
    Compute $\mathcal{L}_{\mathrm{ret}}$ via Equation 7
  end if
  Optimize only student with $\mathcal{L}_{\mathrm{sel}} + \lambda \mathcal{L}_{\mathrm{ret}}$
end for
return Final student π_{r+1}, evaluated with ordinary input

Ordinary-view retention control. The Ordinary control in Table 3(b) matches the ordinary student to the ordinary frozen cycle-start teacher. KL direction, set of interaction steps, token and trajectory averaging, and coefficient match privileged retention; only the conditioning views change.

B.3 Baseline Adaptations

Table 6 summarizes the baselines. AITZ uses the OEL filtering comparison above. ReAct (Yao et al., 2023) is the unchanged initial policy. All training uses fixed logged histories without executing new environment actions.

Table 6: Baseline response sources and training objectives. All methods use ordinary input at evaluation. SDPO in the result tables refers to the offline adaptation below.

Method	Response source	Training objective
ReAct	Initial policy	No update
RFT	Successful logged responses	Cross-entropy
GRPO	Policy-generated candidate groups	Clipped objective + KL
EPD	Cached privileged-teacher responses	Cross-entropy
Offline SDPO	Cached cycle-start student responses	Distillation with correction
OEL	Fresh current-student responses	Distillation

RFT and success filtering. RFT (Yuan et al., 2023) fits successful logged responses by cross-entropy under ordinary histories. RFT and GRPO reuse successes from the current cycle's corpus as needed to match the update budget, without additional environment attempts.

GRPO: offline action matching. GRPO (Shao et al., 2024) samples eight responses per successful logged step. Reward is one if the normalized action type and all arguments match the log, and zero otherwise, including invalid actions. Reasoning text is ignored. Group advantages are

$$A_i = \frac{r_i - \bar{r}}{s_r + 10^{-6}}, \qquad s_r^2 = \frac{1}{7}\sum_{j=1}^{8}(r_j - \bar{r})^2.$$

No additional batch/token advantage whitening is applied. PPO ratios are clipped to $[0.8, 1.28]$. The frozen base Qwen3-4B or Qwen3-8B reference, matching the trained model's size, supplies a `low_var_kl` auxiliary loss with $\beta = 0.01$, excluded from rewards and advantages. Entropy has zero weight.

EPD. EPD (Gou et al., 2026) caches the frozen cycle-start teacher's complete, summary-conditioned responses before training, then fits these targets by ordinary-view cross-entropy.

Offline SDPO. Offline SDPO (Hübotter et al., 2026) uses newly generated cycle-start ordinary responses, one per logged history, as cached KL prefixes throughout the cycle. Its baseline averages valid-token KL within steps, then uniformly across steps in $\mathcal{U}$.

Following SDPO (Hübotter et al., 2026, Appendix A.4), token-level truncated importance sampling (TIS) corrects distillation. Both current-student/old-actor and old-actor/cycle-start-policy ratios are capped at 2. The old actor refreshes before each update; the teacher stays frozen for the cycle. Baseline and augmented SDPO share this correction.

C Detailed Experimental Results

The main results and primary ablations are reported in Tables 1–3. This appendix complements the main experiments with full hyperparameter results, AITZ training dynamics, and a breakdown of training-loop costs.

C.1 Hyperparameter Ablations

Table 7 provides the numerical results for the two sweeps in Figure 4. Both use OEL+ReSAIL with Qwen3-4B on ALFWorld, changing one parameter at a time from the default $\rho = 0.05, \lambda = 0.5$. Training budgets and optimization settings follow Table 5; evaluation follows Appendix A.3.

Table 7: **Complete hyperparameter ablation results.** Success rates (%) are reported as mean (sample standard deviation) over three decoding seeds, with 128 episodes per split in each evaluation. Shading marks the default parameter in each sweep. The two panels report different deployment cycles.

(a) Selection ratio $r = 1$

Top-ρ	**ID**	**OOD**
1%	60.9 (2.1)	54.2 (3.2)
5%	62.2 (2.0)	57.3 (4.0)
25%	60.4 (4.3)	53.6 (5.5)
50%	56.5 (1.8)	54.7 (2.8)
100%	56.8 (3.0)	50.3 (2.4)

(b) Retention weight $r = 3$

λ	**ID**	**OOD**
0	56.5 (3.0)	45.6 (1.6)
0.01	63.0 (1.8)	49.7 (3.2)
0.1	66.7 (2.0)	56.8 (1.6)
0.5	74.2 (1.4)	67.2 (2.3)
1	72.1 (2.0)	63.3 (0.8)

Selection ratio. With $\lambda = 0.5$, the sweep changes only the fraction of interaction steps in $\mathcal{U}$ used for distillation. TLB remains enabled, and PR covers all steps in $\mathcal{U}$ at the fixed weight. Consequently, Top-100% retains both trajectory balancing and retention; it is not the unaugmented OEL baseline.

Retention weight. With $\rho = 0.05$, each setting starts from the same base model and keeps its weight fixed across three cycles, carrying forward its own final student between cycles. The table reports the third-cycle final checkpoints. At $\lambda = 0$, retention is removed while selection and TLB remain enabled. The error bars in the main figure and the standard deviations here describe decoding variability, rather than variation across independent training runs.

C.2 AITZ Training Dynamics

Figure 5(b) reports accuracy every ten updates on 843 non-`stop` steps from a fixed random subset of 101 test episodes, using one decoding seed. At update 100, OEL reaches 64.89% accuracy and OEL + SGS (Top-80% steps) reaches 65.95%. The monitoring subset is not used for model selection. Appendix A.3 specifies the shared scoring rules.

C.3 Computational Cost

Table 8 provides the numerical breakdown for Figure 5(a): approximate Cycle 1 training-loop costs on ALFWorld with the default OEL integration and eight NVIDIA H800 GPUs. GPU-hours are elapsed hours multiplied by eight, counting all allocated GPUs without utilization weighting. Auxiliary JSD diagnostics are excluded; required SGS scoring and privileged-view forward and backward passes for PR over all steps in $\mathcal{U}$ are included. Other is the residual between total loop cost and the three compute components, including model switching, synchronization, and scheduling waits. It is not separately profiled; the measured scope is the training loop, not the full deployment cycle.

Table 8: Training-loop cost on ALFWorld (Cycle 1, GPU-hours on eight H800 GPUs). Scoring includes selection; Other is computed as the residual. Values are approximate; totals are computed before rounding.

Configuration	**Response**	**Scoring**	**Optimization**	**Other**	**Total**
OEL	1.53	0.33	0.91	1.56	4.33
OEL + SGS	1.97	0.55	0.24	1.32	4.08
OEL + SGS + TLB	1.99	0.56	0.24	1.39	4.17
ReSAIL (+ PR)	1.73	0.72	1.13	1.73	5.32

D Analysis and Discussion

This appendix discusses privileged retention and provides the protocols for dual-view evaluation and PI-sensitivity diagnostics. All empirical analyses use ALFWorld and Qwen3-4B. Here, PI consists of trajectory-derived experience summaries.

D.1 Privileged Retention versus Reference KL Regularization

Regularization target. PR applies KL regularization to preserve behavior conditioned on privileged information (PI). Reference KL in GRPO penalizes deviation from a reference policy during reward optimization (Shao et al., 2024). Our objective targets the model's ability to use PI: the updated student becomes the next cycle's teacher and supplies privileged-view supervision.

Conditioning view. Our ordinary-view control and PR match the KL direction, set of interaction steps, loss averaging, and coefficient ($\lambda = 0.5$). Each configuration freezes its own cycle-start policy as the teacher. Omitting response prefixes, the two penalties compare

$$\begin{aligned} \text{Ordinary:} \quad & D_{\mathrm{KL}}\big(\pi^{\mathrm{S}}(\cdot \mid h) \,\big\|\, \pi^{\mathrm{T}}(\cdot \mid h)\big)\,, \\ \text{Privileged:} \quad & D_{\mathrm{KL}}\big(\pi^{\mathrm{S}}(\cdot \mid h, c) \,\big\|\, \pi^{\mathrm{T}}(\cdot \mid h, c)\big)\,, \end{aligned}$$

where h is the interaction history and c is PI. Although the views share parameters, constraining predictions given h does not directly constrain predictions given (h, c). PR applies the latter constraint at all steps in $\mathcal{U}$ (Equation 6).

Functional evidence. Figure 3 separates ordinary-view performance from the ability to benefit from Experience. For OEL+TBSD at Cycle 3 on OOD tasks, ordinary-view retention improves both input-view means over no retention, but its success decreases when Experience is supplied. Under PR, Experience instead raises mean success by 3.1 percentage points. This contrast supports explicitly retaining Experience-conditioned behavior.

Retention strength. The PR weight sweep in Figure 4(b) fixes $\rho = 0.05$ throughout three cycles. Increasing λ from 0.01 to 0.5 raises Cycle 3 mean success from 63.0% to 74.2% on ID and from 49.7% to 67.2% on OOD; increasing it to 1 lowers both means (Appendix C.1). Our GRPO baseline also uses a KL coefficient of 0.01, but coefficient values depend on the primary objective and loss normalization; their numerical sizes alone do not determine the strength of regularization.

D.2 Dual-View Evaluation Details

Evaluation protocol. We evaluate ten checkpoints: the shared initial model and the final models from Cycles 1–3 of OEL+TBSD with no retention, ordinary-view retention, or PR. Each is evaluated with and without Experience on the same 128 OOD tasks, using matched initial states and three decoding seeds. Thinking is disabled; decoding uses temperature 0.4, top-$p = 1$, no top-k truncation, and at most 1,024 tokens per response, with a 30-step interaction limit (Appendix A.3). Table 9 reports the complete results.

Fixed Experience. To hold guidance constant, the frozen initial model collects one reference episode per task in the ordinary view and summarizes it using the success/failure prompts in Appendix E. Both successful and unsuccessful episodes are retained. The resulting 128 task-specific Experiences remain fixed across configurations, checkpoints, and decoding seeds, rather than being refreshed at each training cycle. Each evaluation episode restarts from its task-specific initial state and builds its own action history; only the privileged view receives the fixed summary at every interaction step. This protocol compares checkpoint competence under fixed guidance; it does not assess summary generation or subsequent teaching effects.

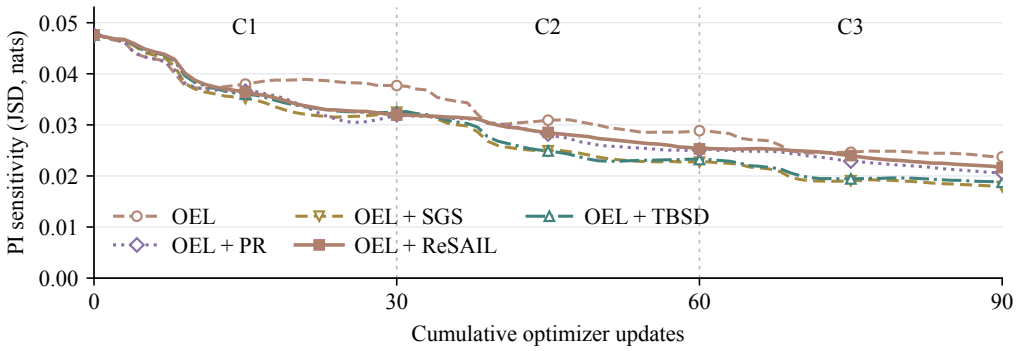
Table 9: **Task success under both input views.** Mean success rates (%) and sample standard deviations over three decoding seeds on 128 ALFWorld OOD tasks. All trained configurations use OEL+TBSD and differ only in retention; the shared initial checkpoint is listed once.

Retention	Cycle	Ordinary view	Privileged view
Shared initial model	0	39.6 (1.8)	49.2 (2.1)
None	1	60.4 (3.0)	61.5 (3.3)
	2	58.9 (1.6)	52.6 (0.9)
	3	45.6 (1.6)	44.8 (3.9)
Ordinary	1	52.1 (1.6)	54.7 (1.6)
	2	66.7 (0.9)	61.5 (1.8)
	3	65.9 (3.5)	62.2 (3.3)
Privileged (PR)	1	57.3 (4.0)	58.9 (0.5)
	2	63.5 (4.7)	69.8 (2.3)
	3	67.2 (2.3)	70.3 (0.8)

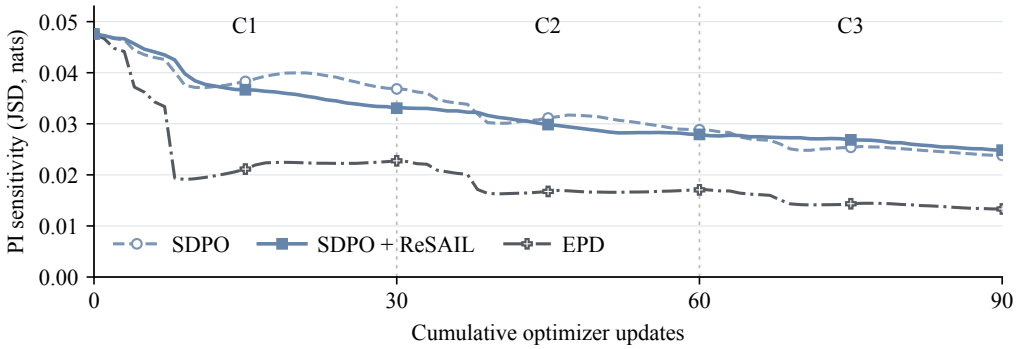
D.3 PI-Sensitivity Measurement and Dynamics

Figure 6 shows the complete, unsmoothed PI-sensitivity trajectories for all eight configurations.

Fixed probe. We use 16 logged trajectories comprising 315 interaction steps. At each cycle's initialization and after every update, we evaluate both views of the current student on the same logged responses. At each response prefix, vocabulary support is fixed to the initial model's ordinary-view Top-20 tokens plus one residual-tail category. This fixed support keeps diagnostic comparisons consistent across checkpoints; training uses the current-student support described in Appendix B.2. PI sensitivity is the JSD between the ordinary- and privileged-view distributions, reported in nats and averaged over all response tokens; longer responses therefore receive more weight. The probe is a fixed diagnostic sample, not an independent task-held-out set, and all curves describe one training seed.



(a) OEL component combinations



(b) SDPO integration and EPD

Figure 6: **Full PI-sensitivity dynamics across three deployment cycles.** Current-student JSD on the fixed ALFWorld/Qwen3-4B probe, using response-token averaging. (a) OEL and component combinations. (b) SDPO with and without ReSAIL, plus EPD. All observations are shown without smoothing across 90 updates; vertical lines mark cycle boundaries and markers are spaced for readability.

Sensitivity generally decreases, with local reversals less pronounced under ReSAIL than under OEL and SDPO. Adding PR to OEL+TBSD does not uniformly raise endpoint JSD across cycles. These curves describe distributional responses to Experience; functional competence is evaluated separately in Section 4.3 and Appendix D.2.

E PROMPT TEMPLATES

Each box assembles a complete model call, including messages, tools, and image positions. Runtime fields such as <CURRENT_OBSERVATION> are placeholders; tags such as <thinking> and <action> are literal. <IMAGE: ...> marks image positions, and square-bracketed role and conditional labels are display annotations. Ellipses denote repeated records or runtime fields, not omitted instructions. Ordinary inputs omit the privileged insertion; text privileged inputs include one outcome-matched summary. Native thinking is disabled in the chat template; the explicitly requested response formats still apply.

E.1 TEXT AGENT PROMPTS

All previous observations and executed actions are serialized into the current user message, separated by spaces. Earlier reasoning is omitted. The full episode history is retained, up to the 30-step text-task limit. For ALFWorld, the task description is extracted from the initial observation, and help is removed from the admissible actions. For TextCraft, episode-level crafting commands are reinserted when available.

ALFWorld: first-step input

[USER MESSAGE]
You are an expert agent operating in the ALFRED Embodied Environment.
Your current observation is: <CURRENT_OBSERVATION>
Your admissible actions of the current situation are: [
'<ACTION_1>'
'<ACTION_2>'
...
'<ACTION_N>'].

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

[IF PRIVILEGED VIEW AND SUCCESSFUL TRAJECTORY]
A successful trajectory for the current task:

Guidance summary:
− Minimal plan: <...>
− Critical actions: <...>
− Checks: <...>
− Avoid: <...>

Use this information as a reference and continue solving the original task.

[ELSE IF PRIVILEGED VIEW AND FAILED TRAJECTORY]
A failed trajectory for the current task:

Failure analysis:
− Failure diagnosis: <...>
− Useful evidence: <...>
− Corrected plan: <...>
− Avoid: <...>

Use this information as a reference and continue solving the original task.
[END CONDITIONAL INSERT]

ALFWorld: later-step input

[USER MESSAGE]
You are an expert agent operating in the ALFRED Embodied Environment. Your task is to: <TASK_DESCRIPTION>
Prior to this step, you have already taken <STEP_COUNT> step(s). Below are the most recent <HISTORY_LENGTH> observations and the corresponding actions you took: [Observation 1: '<OBSERVATION_1>', Action 1: '<ACTION_1>'] [Observation 2: '<OBSERVATION_2>', Action 2: '<ACTION_2>'] ... [Observation K: '<OBSERVATION_K>', Action K: '<ACTION_K>']
You are now at step <CURRENT_STEP> and your current observation is: <CURRENT_OBSERVATION>
Your admissible actions of the current situation are: [
'<ACTION_1>'
'<ACTION_2>'
...
'<ACTION_N>'].

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

[IF PRIVILEGED VIEW AND SUCCESSFUL TRAJECTORY]
A successful trajectory for the current task:

Guidance summary:
− Minimal plan: <...>
− Critical actions: <...>
− Checks: <...>
− Avoid: <...>

Use this information as a reference and continue solving the original task.

[ELSE IF PRIVILEGED VIEW AND FAILED TRAJECTORY]
A failed trajectory for the current task:

Failure analysis:
− Failure diagnosis: <...>
− Useful evidence: <...>
− Corrected plan: <...>
− Avoid: <...>

Use this information as a reference and continue solving the original task.
[END CONDITIONAL INSERT]

TextCraft: first-step input

[USER MESSAGE]
You are an expert agent operating in TextCraft, a text−only Minecraft crafting environment.
Your current observation is:
<CURRENT_OBSERVATION>

Valid actions are:
– get N item
– inventory
– craft N target item using N ingredient, N ingredient

Use only the crafting commands listed in the observation. Always include quantities for get and craft actions.
You should first reason step−by−step. This reasoning process MUST be enclosed within <thinking> </thinking> tags.
Then choose exactly one action and present it within <action> </action> tags.

[IF PRIVILEGED VIEW AND SUCCESSFUL TRAJECTORY]
A successful trajectory for the current task:

Guidance summary:
– Minimal plan: <...>
– Critical actions: <...>
– Checks: <...>
– Avoid: <...>

Use this information as a reference and continue solving the original task.

[ELSE IF PRIVILEGED VIEW AND FAILED TRAJECTORY]
A failed trajectory for the current task:

Failure analysis:
– Failure diagnosis: <...>
– Useful evidence: <...>
– Corrected plan: <...>
– Avoid: <...>

Use this information as a reference and continue solving the original task.
[END CONDITIONAL INSERT]

TextCraft: later-step input

[USER MESSAGE]
You are an expert agent operating in TextCraft. Your goal is: <GOAL_TEXT>
Prior to this step, you have already taken <STEP_COUNT> step(s). Recent history:
[Observation 1: '<OBSERVATION_1>', Action 1: '<ACTION_1>'] [Observation 2:
'<OBSERVATION_2>', Action 2: '<ACTION_2>'] ... [Observation K:
'<OBSERVATION_K>', Action K: '<ACTION_K>']
[IF EPISODE−LEVEL CRAFTING COMMANDS ARE AVAILABLE]
Available crafting commands for this episode:
<CRAFTING_COMMANDS>
[END CONDITIONAL COMMAND CONTEXT]
Your current observation is:
<CURRENT_OBSERVATION>

Valid actions are: get N item, inventory, or craft N target item using N ingredient, N ingredient.
Use only the crafting commands listed above, in the observation, or already visible in the recent history.
You should first reason step−by−step within <thinking> </thinking> tags.
Then choose exactly one action within <action> </action> tags.

[IF PRIVILEGED VIEW AND SUCCESSFUL TRAJECTORY]
A successful trajectory for the current task:

Guidance summary:
− Minimal plan: <...>
− Critical actions: <...>
− Checks: <...>
− Avoid: <...>

Use this information as a reference and continue solving the original task.

[ELSE IF PRIVILEGED VIEW AND FAILED TRAJECTORY]
A failed trajectory for the current task:

Failure analysis:
− Failure diagnosis: <...>
− Useful evidence: <...>
− Corrected plan: <...>
− Avoid: <...>

Use this information as a reference and continue solving the original task.
[END CONDITIONAL INSERT]

E.2 TEXT EXPERIENCE EXTRACTION

Both tasks extract a summary from the task snapshot and the complete sequence of executed actions and resulting observations, using the outcome-specific prompts below. After removing the `<thinking>` block, accepted summaries must have the matching outcome heading and four nonempty fields, without extra headings or a copied trajectory. The accepted summary is cached for the cycle.

Evidence limits. Each action and resulting observation is capped at 240 and 480 characters, respectively. The task snapshot contains at most the first four nonempty lines, each capped at 240 characters.

Text summary: successful trajectory

[USER MESSAGE]
You are writing concise guidance from one completed trajectory.
This guidance will be inserted into a future prompt for the same interactive task.

Output ONLY this format:
<thinking>
...
</thinking>

Guidance summary:
– Minimal plan: ...
– Critical actions: ...
– Checks: ...
– Avoid: ...

Rules:
– Use the complete trajectory as evidence, but do not copy the full task, observations, or command list.
– Use exact action strings in backticks when they are critical.
– Keep it concise and do not invent facts absent from the evidence.
– Think briefly inside <thinking> before writing the guidance summary.

Task snapshot:
<TASK_SNAPSHOT>

Successful trajectory evidence:
1. Action: ‘<ACTION_1>‘
  Result: <OBSERVATION_AFTER_ACTION_1>
2. Action: ‘<ACTION_2>‘
  Result: <OBSERVATION_AFTER_ACTION_2>
...
T. Action: ‘<ACTION_T>‘
  Result: <OBSERVATION_AFTER_ACTION_T>

Text summary: failed trajectory

[USER MESSAGE]
You are writing concise guidance from one failed trajectory.
This guidance will be inserted into a future prompt for the same interactive task.

Output ONLY this format:
<thinking>
...
</thinking>

Failure analysis:
– Failure diagnosis: ...
– Useful evidence: ...
– Corrected plan: ...
– Avoid: ...

Rules:
– Use the complete failed trajectory as evidence, but do not copy the full task, observations, or command list.
– Distinguish useful partial progress from the action pattern that caused failure.
– Use exact action strings in backticks when they are critical.
– Keep it concise and do not invent facts absent from the evidence.
– Think briefly inside <thinking> before writing the failure analysis.

Task snapshot:
<TASK_SNAPSHOT>

Unsuccessful trajectory evidence:
1. Action: ‘<ACTION_1>‘
  Result: <OBSERVATION_AFTER_ACTION_1>
2. Action: ‘<ACTION_2>‘
  Result: <OBSERVATION_AFTER_ACTION_2>
...
T. Action: ‘<ACTION_T>‘
  Result: <OBSERVATION_AFTER_ACTION_T>

E.3 GUI AGENT PROMPT

The complete call comprises a system message, the `mobile_use` tool definition, and a multimodal user message. Ordinary and privileged views differ only in the inserted guideline. Both include all ground-truth prior actions (`None` at the first step) and one current screenshot; earlier screenshots and model predictions are excluded.

GUI action input: system, tool, and user

[SYSTEM MESSAGE]
You are a GUI agent. Use the mobile_use tool to complete the user's task from the current screenshot and prior action history.

Response format

Response format for every step:
1) Thought: one concise sentence explaining the next move (no multi−step reasoning).
2) Action: a short imperative describing what to do in the UI.
3) A single <tool_call>...</tool_call> block containing only the JSON object.

Rules:
− Output exactly in the order: Thought, Action, <tool_call>.
− Be brief: one sentence for Thought, one for Action.
− Use normalized integer coordinates from 0 to 999.
− Do not output anything else outside those three parts.
− If finishing, use action=terminate in the tool call.

[TOOL DEFINITION]

```
{
  "type": "function",
  "function": {
    "name": "mobile_use",
    "description": "Use a touchscreen to interact with a mobile device. You can click, type,
swipe, press system buttons, wait, and finish the task. Some applications take time to load, so
wait when the UI has not settled. Coordinates are normalized integers from 0 to 999, where
(0, 0) is the top−left corner; target the center of an element.",
    "parameters": {
      "type": "object",
      "properties": {
        "action": {
          "type": "string",
          "description": "Action to perform: click a point; long_press a point; swipe from
coordinate to coordinate2; type into the active field; answer the user; press a system_button;
open an app by name; wait; or terminate with status.",
          "enum": ["click", "long_press", "swipe", "type", "answer",
                   "system_button", "open", "wait", "terminate"]
        },
        "coordinate": {
          "type": "array", "items": {"type": "integer"},
          "description": "Required for click, long_press, and swipe: normalized [x,y]."
        },
        "coordinate2": {
          "type": "array", "items": {"type": "integer"},
          "description": "Required for swipe: normalized ending [x,y]."
        },
        "text": {
          "type": "string", "description": "Required for type, answer, and open."
        },
```

GUI action input: system, tool, and user (continued)

```
      "time": {
        "type": "number", "description": "Required for long_press and wait."
      },
      "button": {
        "type": "string", "enum": ["Back", "Home", "Menu", "Enter"],
        "description": "Required for system_button."
      },
      "status": {
        "type": "string", "enum": ["success", "failure"],
        "description": "Required for terminate."
      }
    },
    "required": ["action"]
  }
 }
}

[USER MESSAGE]
The user query: <USER_INSTRUCTION>

[IF PRIVILEGED VIEW]
Use this task guideline when choosing the next action:
<remark>
How to do:
1) <...>

Watch out:
- <...>
</remark>
[END CONDITIONAL INSERT]

Task progress (operations already completed on the current device):
[IF PREVIOUS ACTIONS EXIST]
Step 1: <GROUND_TRUTH_PREVIOUS_ACTION_1>
Step 2: <GROUND_TRUTH_PREVIOUS_ACTION_2>
...
Step t-1: <GROUND_TRUTH_PREVIOUS_ACTION_t-1>
[OTHERWISE]
None
[END CONDITIONAL HISTORY]
Choose the next action from the current screenshot.
<IMAGE: CURRENT_SCREENSHOT>
```

E.4 GUI Experience Extraction

The initial student independently attempts each logged step under ordinary input; correctness follows AITZ EM (Appendix A.3). One accepted <remark> per trajectory is cached throughout training. OEL and OEL + SGS sample fresh ordinary-view responses at each update (Appendix B.2).

GUI summary input: system and complete user message

[SYSTEM MESSAGE]
You are a conservative GUI error−correction summarizer. Correct−step actions and lines marked CORRECT action are the only ground truth. Lines marked WRONG attempt are prohibited negative evidence. Never turn a wrong attempt into a recommended or prerequisite operation, and never infer extra operations from the user query, screenshots, or task−world knowledge. If the demonstrated route appears incomplete or unusual, summarize only its shown correct actions. Keep the required output format exactly.

[USER MESSAGE]
Summarize this successful GUI trajectory into a short guideline for the student model.

User query: <USER_INSTRUCTION>

The student was tested independently at every step. Correct steps are text−only route context. A screenshot is shown only for a failed step, together with the student's attempted action and the correct action. Focus on the failed steps and the difference between the attempt and the correct action.

For failed steps, "WRONG attempt — DO NOT DO" is negative evidence shown only to diagnose the mistake. "CORRECT action — DO THIS" is the authority. Build the route only from correct−step actions and CORRECT actions. A wrong attempt may appear only as an explicit "do not" warning followed by the correct alternative.

[FOR EACH STEP, IN TRAJECTORY ORDER]
[IF THE STUDENT ACTION IS CORRECT]
Step <INDEX> — student handled correctly
Action: <CANONICAL_CORRECT_ACTION>
[OTHERWISE]
Step <INDEX> — student failed
<IMAGE: SCREENSHOT_AT_THIS_STEP>

CORRECT action — DO THIS: <CANONICAL_CORRECT_ACTION>
WRONG attempt — DO NOT DO: <STUDENT_ATTEMPT>
[END STEP BRANCH; REPEAT FOR ALL STEPS]

Briefly cover the demonstrated route only; do not invent intermediate actions. Spend most detail on failed steps. Preserve text that must be typed, distinguish taps, swipes, system buttons, waiting, and completion, and state what visible UI cue should trigger each uncertain action. Include one useful fallback. Before answering, verify that every failed step's CORRECT action is represented and no WRONG attempt is written as a positive instruction. Do not include coordinates or tool calls.

Output only:
<remark>
How to do:
1) ...

Watch out:
− ...
</remark>