WHERE SHOULD LLM AGENTS ADAPT AT TEST TIME? IN CONTEXT, IN WEIGHTS, OR BOTH

Anonymous authors
Paper under double-blind review

ABSTRACT

An LLM agent operating repeatedly in the same deployment environment, such as a code repository, a database, or an application suite, can adapt by retaining experience in context, updating its weights, or both. These approaches are usually studied separately, making it difficult to determine their individual contributions and when combining them helps. We compare adaptation through in-context learning (ICL), reinforcement learning with verifiable rewards using GRPO, and self-distillation, individually and in combination, under a common online protocol. Each agent encounters the same task streams, is evaluated on each task before adapting to it, and learns only from its own attempts and environment feedback. We evaluate 14 adaptation configurations and a stateless baseline with Qwen3.5-9B on 109 tasks from four environments drawn from SWE-bench Verified, AppWorld, ScienceWorld, and BIRD-Critic-SQLite, using five task orders per environment. ICL provides the largest standalone gain: its best configuration raises the success rate, averaged across environments, from 28.8% to 49.4%. Adding GRPO to this configuration yields the highest success rate among all configurations at 51.6%. GRPO improves the average success rate under every ICL policy, whereas the effect of self-distillation varies across environments. Further analyses show that GRPO maintains interaction before submission as retained context grows in the database environment, and that adding weight updates to ICL yields larger gains in pass@k at larger k. Finally, GRPO provides larger average gains with an ICL policy that stops admitting new experience once full (ICL-Prefix) than with one that continually replaces the oldest experience (ICL-FIFO). We release a modular framework for evaluating and combining these adaptation mechanisms. Together, our findings provide evidence that context and weight adaptation are complementary and that retaining experience in context does not exhaust the gains available from learning from it.

1 INTRODUCTION

Large language model (LLM) agents are increasingly deployed in persistent roles: a coding agent on one repository, an analyst on one database, or an assistant on one suite of applications (Ouyang et al., 2026; Wei et al., 2026). The tasks of one environment share latent structure specific to it: the idioms of its codebase, the conventions of its schema, or the behavior of its interfaces. An agent that discovers this structure through interactions and feedback from the environment can exploit it on later tasks; one that has learned how a database is organized answers later questions with far fewer queries (Asawa et al., 2026). Agents can retain experience in their context (Allard et al., 2026; Ouyang et al., 2026; Ye et al., 2026a) to condition their behavior, but are limited by context size and computational costs. For a fixed architecture, the LLM's behavior is governed by a policy $\pi_\theta(\cdot \mid c)$, so experience can change it in two places: the context c and the weights θ. This paper asks where an agent should adapt at test time: in context, in weights, or in both.

An agent can specialize to its own deployment environment while the base model stays unchanged for other deployments (Tandon et al., 2025; Hübotter et al., 2026b). We call this *test-time adaptation* of an agent after deployment. At test time, every arriving task counts toward its performance and supplies experience for later tasks at the same time; accumulated state persists across the stream rather than resetting between tasks (Akyürek et al., 2025; Tandon et al., 2025). Learning at test time

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

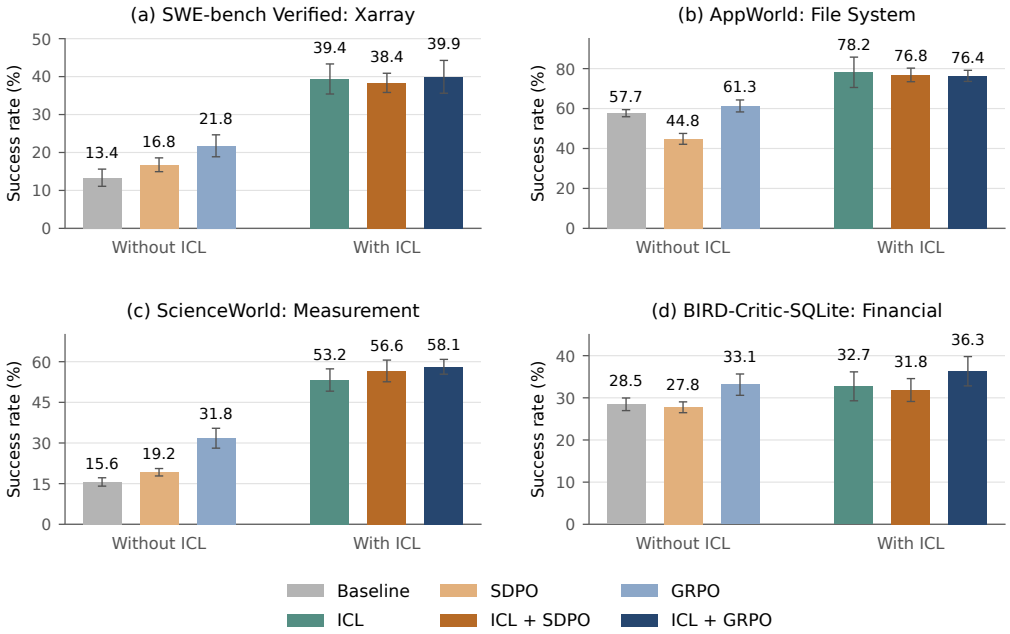

Figure 1: **On average across benchmarks, in-context learning (ICL) gives the largest standalone gain, while ICL with GRPO gives the highest overall gain.** Success rate (%) of the best configuration in each adaptation family on each world with mean and sample standard deviation across five seeds.

can only use the agent's own attempts and environmental feedback, without demonstrations or a stronger teacher (Silver & Sutton, 2025; Hübotter et al., 2026a).

Context and weights have complementary properties as adaptation channels (Tiwari et al., 2026). Context is cheap to change, takes effect at once, and leaves the model weights intact. However, adaptation through context is bounded by the context window, in which different experiences compete, and conditioning the model through it comes with an extra computational cost (Hu et al., 2026). Weights persist without occupying context, but they are costly to update, and an update that helps one task can hurt others (Tiwari et al., 2026; Luo et al., 2025b). Tandon et al. (2025) accordingly read weights updated at test time as long-term memory and the context window as short-term memory. Both adaptation mechanisms are well developed. Memory-based agents keep their weights frozen and retrieve reflections, workflows, cases, or strategies from past attempts into their context (Shinn et al., 2023; Wang et al., 2024; Zhou et al., 2025; Zhang et al., 2025; Ouyang et al., 2026), whereas test-time training, reinforcement learning, and self-distillation write experience into the weights (Akyürek et al., 2025; Yuksekgonul et al., 2026; Shenfeld et al., 2026; Hübotter et al., 2026a). But the two lines of work have developed mostly apart (Jiang et al., 2026; Hebbar et al., 2026), and they are typically evaluated on different benchmarks under different assumptions, so their reported gains cannot be attributed to the mechanism.

Whether these two adaptation channels substitute for or complement each other is open, although systems that optimize a prompt or harness jointly with the weights suggest complementarity (Dorovatas et al., 2026; Tiwari et al., 2026; Hebbar et al., 2026). What is missing is a controlled comparison of the two adaptation channels that answers the following questions:

RESEARCH QUESTIONS

What do context and weights each contribute to test-time adaptation?
Does combining them help? And how should the two be coordinated?

We conduct extensive experiments using our framework, in which tasks from an agentic benchmark that share latent structure are arranged into a stream, and every adaptation mechanism processes the stream and is scored on each task as it arrives. The agent learns only from its own attempts and the feedback the environment returns, and it may manage its context, weights, or both from one task to the next. We build four *worlds*, each a set of tasks that share one environment: issues for one repository of SWE-bench Verified (Jimenez et al., 2024), tasks using one application suite of AppWorld (Trivedi et al., 2024), the measurement tasks of ScienceWorld (Wang et al., 2022), and tasks on one database of BIRD-Critic-SQLite (Li et al., 2025). Elaborate systems bundle an adaptation mechanism with many other design choices, such as how experience is summarized, retrieved, or judged. We instead compare the core representatives, alone and in combination: raw experience used in context; reinforcement learning with verifiable rewards (Shao et al., 2024); and self-distillation (Hübotter et al., 2026a). Each mechanism is measured by its gain over the same agent run without adaptation (Asawa et al., 2026).

Our contributions are threefold:

- **Controlled study of adaptation mechanisms.** Under one protocol, which gives every adaptation mechanism the same tasks, order, feedback, and constraints (Section 3), we compare a stateless baseline and adaptation mechanisms that adapt the context, the weights, or both, with Qwen3.5-9B on 109 tasks from four benchmarks, each under five task orders (Section 4).
- **Evidence that context and weights are complementary.** While in-context learning (ICL) gives the larger overall standalone gain, adding GRPO to it raises success rate under every ICL policy, while the effect of self-distillation depends on the benchmark (Figure 1; Table 2). The added gain comes with measurable changes: training keeps the agent querying a database before it submits, raises pass@16 on repository issues, and also on average adds more to an ICL mechanism that stops admitting experience into context once full (ICL-Prefix) than to one with a first-in-first-out eviction policy (ICL-FIFO) (Section 5).
- **Extensible framework for evaluating adaptation mechanisms.** We provide a framework built on verl (Sheng et al., 2024), which registers benchmarks and adaptation mechanisms as modules, where one mechanism can combine in-context experience, reinforcement learning with verifiable rewards, and self-distillation.

2 RELATED WORK

Adaptation through context. Agents retain experience as interaction histories (Park et al., 2023), reflections (Shinn et al., 2023; Zhao et al., 2024), reusable workflows (Wang et al., 2023; 2024), or strategies and playbooks (Ouyang et al., 2026; Zhang et al., 2025). Experience selection can itself be learned (Zhou et al., 2025). These methods change the context while keeping the model fixed. Raw history is already a strong baseline in CL-Bench (Asawa et al., 2026), but bounded context makes old and new experience compete (Hu et al., 2026). We study raw-trajectory ICL policies to separate the contribution of context from additional summarization and retrieval machinery.

Adaptation through weights. Test-time training adapts a model using the test input (Sun et al., 2020). For LLMs, related methods learn from few-shot examples (Akyürek et al., 2025), reinforce solutions to a target problem (Yuksekgonul et al., 2026; Hübotter et al., 2025), or distill a self-teacher conditioned on demonstrations or feedback (Shenfeld et al., 2026; Hübotter et al., 2026a). Such methods often reset between tasks; here the adapter and optimizer state persist across a deployment. This differs from retaining general capability under a changing task distribution, a central concern of continual learning (Shi et al., 2024; Luo et al., 2025b).

Combining context and weights. Context distillation transfers context-conditioned behavior into weights (Snell et al., 2022; Ye et al., 2026b); online experiential learning alternates extracting experience with consolidating it (Ye et al., 2026a). Other systems jointly adapt prompts and weights

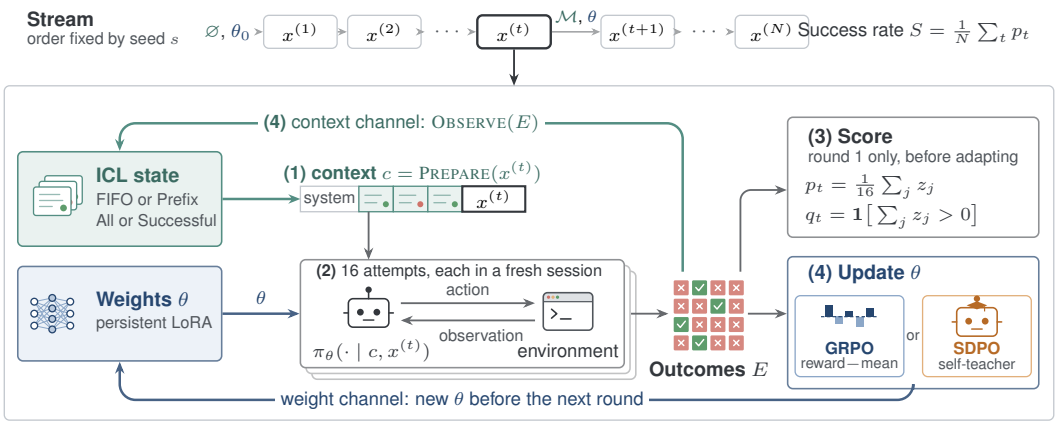

Figure 2: Tasks of one world arrive once each, in an order fixed by seed s, and the retained experience of mechanism $\mathcal{M}$ and the weights θ carry over between tasks. At position t, each round (1) prepares a context c, (2) makes 16 attempts in fresh sessions of the environment, (3) in round 1 only, scores them before any adaptation to $x^{(t)}$, and (4) adapts through the context channel, the weight channel, or both. ICL adapts only in context, GRPO and SDPO only in weights, hybrids in both, and the baseline in neither (Algorithm 1).

(Tiwari et al., 2026), choose between harness and weight updates (Hebbar et al., 2026), or use retained experience during reinforcement learning (Liu et al., 2026). These approaches motivate joint adaptation (Dorovatas et al., 2026). We compare minimal combinations with their constituents under the same online protocol, measuring both their success rates and the behavioral changes that accompany it.

Evaluation and infrastructure. Recent benchmarks evaluate agents over sequential experience (Wu et al., 2024; Wei et al., 2026; He et al., 2026). CL-Bench measures improvement over the same model without accumulated state (Asawa et al., 2026); we adopt this comparison. Closest to our setting, Harrington et al. (2026) compare several update operators under one protocol, but update between stages using curated data and do not combine explicit in-context experience with weight updates. Our agents learn only from their own attempts and environmental feedback. Our framework implements this comparison on verl (Sheng et al., 2024); its mechanism interface exposes context and weight adaptation in the same loop, complementing agent-training and distillation systems (Luo et al., 2025a; Sun et al., 2026; Fallah et al., 2026).

3 SETUP

3.1 STREAM OF RELATED TASKS

Agentic benchmarks typically score each task on its own, with a fresh agent, yet many are built on a stateful environment, such as a code repository, whose tasks share structure specific to it. Selecting the tasks that share one environment and ordering them as a stream recasts such a benchmark for test-time adaptation. Grouping tasks of one type, such as mathematics problems, would not, since their common structure is a general skill that pretraining already targets. We define a *world* $\mathcal{W}$ as a set of tasks $\mathcal{X} = \{x_1, \ldots, x_N\}$ that share an environment. A seed s fixes the *stream* $(x^{(1)}, \ldots, x^{(N)})$, a random order in which each task appears once. For each task, the agent makes m rounds of n *attempts* (Figure 2). An attempt τ is a multi-turn interaction with a fresh session of the environment that ends when the agent submits a solution or reaches the goal state, or at a limit. The benchmark then returns an *outcome* $o = (r, z, f)$: a reward $r \in [0, 1]$, a success indicator $z \in \{0, 1\}$, and textual feedback f, such as how many tests passed. The adaptation mechanism observes the outcome (See Table 1 for Notations).

We build four worlds. In XARRAY, the agent resolves SWE-bench Verified issues of `pydata/xarray` with shell commands, judged by each issue's tests (22 tasks); in FILE SYS-

Table 1: **Notation.** Rounds are indexed by i and attempts by j; a subscript t marks position t where needed, as in $z_{t,j}$. p_t and q_t use round 1 only.

Symbol	Meaning	Symbol	Meaning
$\mathcal{W}$; $\mathcal{X} = \{x_1, \ldots, x_N\}$	world; its N tasks	$\mathcal{M}$	adaptation mechanism
s; $x^{(t)}$	seed; task at position t	PREPARE; OBSERVE	build context c; admit E
m; n	rounds; attempts per round	θ_0; θ	base; current weights
$\tau \sim \pi_\theta(\cdot \mid c, x)$	attempt	$E = \{(\tau_j, o_j)\}_{j=1}^{n}$	a round's attempts, outcomes
$o = (r, z, f)$	reward, success, feedback	p_t; q_t	success rate; any success
S; pass@16	means of p_t; of q_t	ΔS	gain over the baseline

TEM, it writes Python that calls the APIs of AppWorld's apps (21 tasks whose primary app is the file system); in MEASUREMENT, it issues text commands in ScienceWorld's simulated laboratory to measure a temperature or a melting point (22 tasks); and in FINANCIAL, it fixes faulty SQL for one BIRD-Critic-SQLite database, which it can query before submitting (44 tasks). Following CL-Bench (Asawa et al., 2026), a world's tasks should share structure, leave headroom for the stateless baseline (Table 2), and inform later tasks; gains over that baseline test the last. Unlike CL-Bench, which builds new tasks, we provide a framework for reusing existing benchmarks, environments, and graders.

3.2 ADAPTATION MECHANISMS

We compare the stateless baseline with fourteen adaptation mechanisms, built from four ICL policies and two weight-update rules. **ICL** replays prior attempts as conversations, including environment observations and outcome feedback. Under a bounded token allowance, ICL-FIFO evicts the oldest episodes to admit new ones; ICL-Prefix stops admitting episodes after the first overflow. Each has two admission policies: *All* retains the best attempt from each round, including failures, whereas Successful retains at most one successful attempt per task.

GRPO updates a persistent LoRA adapter (Hu et al., 2022) using a clipped policy-gradient objective (Shao et al., 2024; Schulman et al., 2017). **SDPO** distills a copy of the agent conditioned on a successful sibling attempt or, when unavailable, the attempt's feedback (Hübotter et al., 2026a) into a persistent LoRA adapter. The teacher weights are an exponential moving average of the student. Each **hybrid** combines one ICL with one updater: training uses attempts generated with the experience inside the context, and updated weights generate later experience. In ICL+SDPO, the teacher also sees the student's context; the additional teacher information is hindsight. Appendix A gives the formal interfaces and losses.

3.3 SCORING

The first round's 16 attempts at each task share the same context and weights, and we score them *before* their experience enters context or updates the weights. Further rounds on that task supply adaptation experience but do not enter the main score, so each task tests transfer from earlier tasks. Every mechanism receives the same task order, limits, sampling temperature, and source of feedback, learns only from its own attempts and the environment's outcomes, and completes its updates before the next round. Algorithm 1 gives the full loop; this score-then-update protocol is related to prequential evaluation (Dawid, 1984; Gama et al., 2013).

Metrics. For task t, let $z_{t,j}$ indicate success of first-round attempt j, and let $p_t = 16^{-1} \sum_{j=1}^{16} z_{t,j}$. Across the world's N tasks, we report

$$S = \frac{1}{N} \sum_{t=1}^{N} p_t, \qquad \text{pass@16} = \frac{1}{N} \sum_{t=1}^{N} \mathbf{1}\left[\sum_{j=1}^{16} z_{t,j} > 0 \right]. \tag{1}$$

S denotes the success rate, averaged over tasks. Pass@16 is the fraction of tasks with at least one observed success among 16 attempts. We compare mechanisms on the same stream, with ΔS denoting the difference from the stateless baseline. Formal details are in Appendix A; implementation specifics are in Section A.3.

Algorithm 1 Test-time adaptation on one world. Each task is scored before the agent adapts to it.

Input: tasks $\mathcal{X}$, seed s, mechanism $\mathcal{M}$, base weights θ_0, rounds m, attempts n

$(x^{(1)}, \dots, x^{(N)}) \leftarrow \textsc{Stream}(\mathcal{X}, s)$; $\theta \leftarrow \theta_0$; $\mathcal{M}$ starts with no retained experience
for $t = 1, \dots, N$ **do** ▷ each task once, in stream order
  for $i = 1, \dots, m$ **do** ▷ rounds on the same task
    $c \leftarrow \mathcal{M}.\textsc{Prepare}(x^{(t)})$ ▷ context from retained experience
    $E \leftarrow \{(\tau_j, o_j)\}_{j=1}^{n}$ with $\tau_j \sim \pi_\theta(\cdot \mid c, x^{(t)})$ ▷ n attempts in fresh sessions
    if $i = 1$ **then** ▷ score before adapting
      $p_t \leftarrow \frac{1}{n}\sum_j z_j$, $q_t \leftarrow \mathbf{1}\big[\sum_j z_j > 0\big]$
    $\theta \leftarrow \textsc{Update}(\theta, E)$ ▷ GRPO or SDPO; unchanged for ICL and the baseline
    $\mathcal{M}.\textsc{Observe}(E)$ ▷ retain experience for later rounds
return $S = \frac{1}{N}\sum_t p_t$, $\text{pass@16} = \frac{1}{N}\sum_t q_t$

4 Experiments

4.1 Setup

We evaluate Qwen3.5-9B with thinking mode off on the four worlds of Section 3.1, using five stream seeds (0–4), 16 attempts per round, and temperature 1. Weight updates train a rank-32 LoRA adapter with AdamW at learning rate 10^{-5}. The baseline, ICL, and GRPO configurations take two rounds per task; GRPO takes four optimizer epochs per round. SDPO takes three rounds with one optimizer epoch each.

Table 2: **Success rate for every adaptation configuration.** Percent, mean ± SD across five seeds. ICL, admission, and training variants are reported separately. No ICL and no training is the baseline; "Succ." denotes Successful admission. Bold marks the highest mean in each column.

ICL	Admission	Training	Xarray	File System	Measurement	Financial	Mean
None	–	None	13.4 ± 2.3	57.7 ± 1.8	15.6 ± 1.5	28.5 ± 1.5	28.8 ± 0.9
FIFO	All	None	39.4 ± 4.0	66.9 ± 7.4	50.3 ± 5.3	28.5 ± 1.6	46.3 ± 2.4
FIFO	Succ.	None	35.1 ± 2.1	77.9 ± 6.8	46.1 ± 13.2	32.6 ± 1.9	47.9 ± 4.7
Prefix	All	None	35.6 ± 4.2	62.6 ± 11.1	48.0 ± 8.0	27.9 ± 1.6	43.5 ± 4.3
Prefix	Succ.	None	33.4 ± 1.3	**78.2** ± 7.6	53.2 ± 4.1	32.7 ± 3.4	49.4 ± 2.3
None	–	SDPO	16.8 ± 1.8	44.8 ± 2.7	19.2 ± 1.4	27.8 ± 1.3	27.1 ± 0.6
FIFO	All	SDPO	38.4 ± 2.5	53.7 ± 9.5	56.6 ± 4.0	31.1 ± 2.6	44.9 ± 1.8
FIFO	Succ.	SDPO	36.2 ± 3.6	75.2 ± 7.8	52.8 ± 14.4	31.8 ± 2.7	49.0 ± 3.2
Prefix	All	SDPO	33.7 ± 8.9	59.0 ± 8.4	55.5 ± 11.3	29.8 ± 2.6	44.5 ± 2.3
Prefix	Succ.	SDPO	35.2 ± 3.2	76.8 ± 3.4	50.2 ± 9.7	30.6 ± 1.7	48.2 ± 3.5
None	–	GRPO	21.8 ± 2.9	61.3 ± 3.0	31.8 ± 3.7	33.1 ± 2.5	37.0 ± 1.5
FIFO	All	GRPO	38.9 ± 3.9	66.1 ± 3.3	50.1 ± 5.5	34.4 ± 4.6	47.4 ± 1.4
FIFO	Succ.	GRPO	36.6 ± 4.1	76.4 ± 2.8	55.2 ± 5.3	33.0 ± 0.9	50.3 ± 1.4
Prefix	All	GRPO	**39.9** ± 4.3	69.4 ± 7.9	55.9 ± 5.9	34.0 ± 3.6	49.8 ± 2.2
Prefix	Succ.	GRPO	37.6 ± 3.2	74.5 ± 9.1	**58.1** ± 2.7	**36.3** ± 3.5	**51.6** ± 3.0

4.2 Context provides the larger gain; training still contributes

Context is the stronger single channel. ICL gains more than either weight update alone on Xarray, File System, and Measurement (Figure 1; Table 2); Financial is the exception, where GRPO alone outperforms ICL. Adding GRPO to ICL raises the four-world mean success rate under every ICL policy (Table 2, Mean column). With the best configuration of each family on that average, ICL (Prefix, Successful admission) raises the success rate from 28.8% to 49.4%, and adding GRPO to the same ICL policy raises it to 51.6%. The added gain is largest on Measurement and negative on File System. Thus the larger contribution from context does not exhaust the benefit of adapting the weights.

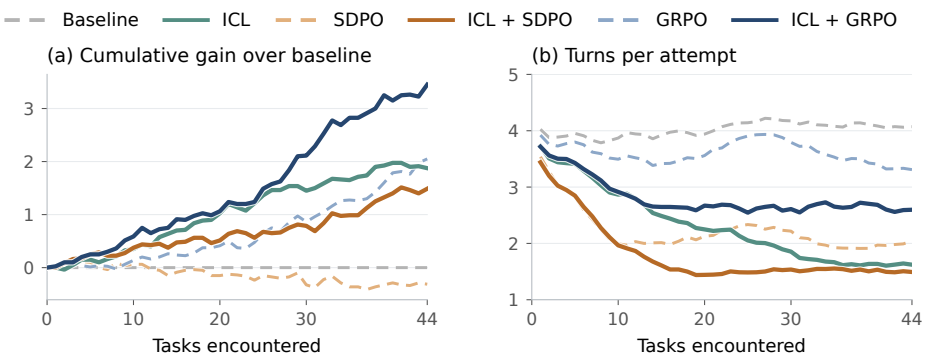

Figure 3: **On FINANCIAL, GRPO keeps the agent querying the database as the retained context grows.** (a) Cumulative gain over the baseline: the running sum of per-task success rate (the fraction of successful first-round attempts) minus the baseline's, averaged over five seeds. (b) Mean turns per first-round attempt (turn limit five), five-seed mean with a centered nine-task moving average. Each family uses its best configuration on FINANCIAL (Table 2); ICL and ICL+GRPO share the same ICL policy (Prefix, Successful admission).

The benefit depends on the update rule and environment. SDPO alone trails GRPO on success rate in every world, and adding it to ICL helps on MEASUREMENT but hurts on FILE SYSTEM (Table 2).

5 WHAT DOES TRAINING ADD TO ICL?

5.1 TRAINING KEEPS THE AGENT QUERYING THE DATABASE

FINANCIAL shows where the gain of ICL+GRPO over ICL comes from. ICL+GRPO accumulates a larger gain in success rate over the baseline, with the gap opening in the second half of the stream (Figure 3a). An attempt may run SQL queries for up to four turns before it submits a fix. As the context fills, ICL attempts become shorter in every seed, whereas with GRPO they level off (Figure 3b). The same holds for all four ICL policies, and adding GRPO raises the success rate under each of them (Appendix Table 8).

Short attempts are the ones that fail. A one-turn attempt submits without querying the database and succeeds about half as often as an attempt of two to four turns (Appendix Table 7). The baseline errs in the other direction: over a quarter of its attempts use all five turns without submitting, which ICL largely removes. As retained context grows, however, ICL increasingly submits without querying. GRPO's objective favors the middle ground. Within each scored round, one-turn attempts receive below-average reward in every family and two- to four-turn attempts above-average reward (Appendix Table 7), so its updates move probability toward querying before submitting. SDPO, which distills toward a successful sibling, shortens attempts even further and gains little on FINANCIAL.

This observation qualifies an efficiency-centered account of adaptation. ICL greatly reduces turns in FILE SYSTEM and FINANCIAL, but weight updates do not uniformly compress trajectories. Conversely, SDPO can shorten attempts without improving the success rate.

5.2 TRAINING IMPROVES PASS@k ON CODE ISSUE FIXING

Success rate is pass@1, the probability of success for a single attempt, whereas an agent allowed several attempts benefits from pass@k at larger k. On XARRAY, adding GRPO or SDPO to ICL barely changes the success rate, but its gain in pass@k grows steadily with k (Figure 4), so the update adds more to pass@16 than to success rate. This holds under every ICL policy (Appendix Table 10). The gain at pass@16 exceeds the gain at pass@1 in four of five seeds for each update (Appendix Figure 6).

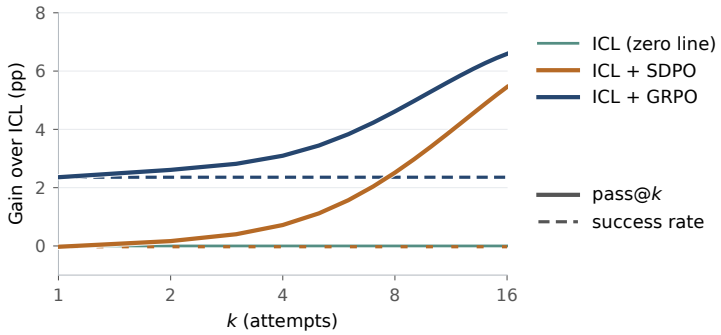

Figure 4: **On XARRAY, training adds more to pass@16 than to mean success rate.** Gain of each hybrid over ICL in pass@k (solid) and in success rate (dashed), averaged over five seeds; ICL is the zero line. Success rate, the expected fraction of successful attempts, does not depend on k and equals pass@1. Pass@k is the unbiased estimate from each task's 16 first-round attempts.

5.3 TRAINING ADDS MORE TO ICL THAT STOPS ADMITTING EXPERIENCE

Once the context allowance fills, ICL-FIFO evicts the oldest episodes to admit new ones, whereas ICL-Prefix stops admitting episodes. Figure 5 compares the success rate gain from adding GRPO under each rule. With All admission, averaged over the four worlds, GRPO raises the success rate with an ICL-Prefix policy in every seed but with an ICL-FIFO policy in only three of five, and ICL-Prefix's gain exceeds ICL-FIFO's in every seed (Appendix Table 12). One explanation could be that since the policy optimization is directly training the policy conditioned on the context, a fixed context could result in more stable optimization.

6 DISCUSSION AND LIMITATIONS

Our results show that retaining experience in context leaves room for further improvement through weight adaptation. ICL provides the largest standalone gain, while adding GRPO improves average success rate across the four environments under every ICL policy. The benefit nevertheless depends on the environment and update rule: some configurations lose performance, and self-distillation does not reproduce GRPO's consistent average gains. These findings motivate treating context and weight updates as coupled design choices, with their contribution assessed jointly for the deployment environment.

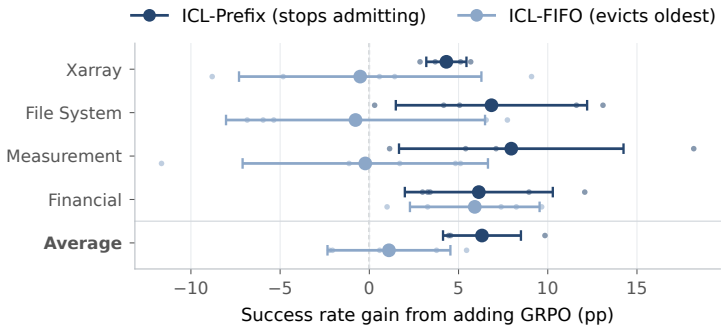

Figure 5: **GRPO adds more to ICL-Prefix than to ICL-FIFO.** All admission. Success rate gain from adding GRPO to each ICL policy: mean and sample SD over five seeds, with individual seeds as small dots. Once the context allowance fills, ICL-Prefix stops admitting episodes, whereas ICL-FIFO evicts the oldest ones.

The behavioral analyses help characterize this complementarity. On FINANCIAL, GRPO maintains database interaction before submission as retained context grows, alongside a higher success rate. On XARRAY, weight updates yield larger gains in pass@k at larger sampling budgets than in single-attempt success rate. Thus, adaptation can change both how an agent interacts and which tasks it solves across multiple attempts. Shorter trajectories alone are insufficient evidence of better adaptation, and single-attempt success rate does not fully capture the benefit. The association between querying and success, however, does not establish that longer interaction causes the improvement.

The retention comparison further suggests that ICL policy affects the value of training. With All admission, GRPO yields a larger average increase in success rate with ICL-Prefix, which stops admitting experience once full, than with ICL-FIFO, which continually refreshes its history. This motivates studying how context refresh and weight updates should be coordinated. The comparison does not establish that weights preserve knowledge from excluded or evicted episodes, or that fixed context stabilizes optimization. Each run develops its own trajectories and retained context; gains observed after ICL-Prefix freezes may also reflect updates made before that point. Controlled interventions that hold retained context fixed or stop weight updates at the freeze point would help distinguish these explanations.

We compare simple raw-trajectory ICL policies and specific update recipes; stronger retrieval, summarization, or training methods may change their relative contributions. GRPO and SDPO also use different adaptation schedules, so their comparison does not establish superiority at equal compute.

Our focus is post-deployment adaptation to a particular environment, where the objective is to improve performance on subsequent tasks in that environment rather than to produce a single adapted model that performs well across all possible deployments. Accordingly, we do not evaluate final recall, performance after removing context, or retention of general capabilities. These remain relevant limitations if the agent's responsibilities change, its context becomes unavailable, or the adapted model is reused elsewhere.

7 CONCLUSION

We study how LLM agents should adapt after deployment by comparing learning through context, weights, and their combination under a common online protocol. Across four environments, retaining experience in context provides the largest standalone gain, while adding GRPO achieves the highest average success rate among the evaluated configurations. The benefits depend on the environment, update rule, and ICL policy, motivating joint consideration of how experience is retained and learned from. We also provide a modular framework that supports registering new benchmarks and adaptation mechanisms to extend these comparisons. Our findings point toward agents that improve within their deployment environments by jointly retaining experience in context and learning from it through weight updates.

AI USE STATEMENT

We used ChatGPT and Claude Code to assist with research execution, including code development and implementation of experiments, and to draft and revise portions of the manuscript for clarity and readability. We did not use these tools for research ideation. Synthetic dataset generation and mathematical proof development are not applicable to this work. The authors carefully reviewed and verified the AI-assisted code and manuscript content for correctness and consistency with the experimental results. We take full responsibility for the final paper, including all text, claims, code, and reported results.

REFERENCES

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The Surprising Effectiveness of Test-Time Training for Few-Shot Learning, March 2025. URL http://arxiv.org/abs/2411.07279. arXiv:2411.07279 [cs].

Marc-Antoine Allard, Arnaud Teinturier, Victor Xing, and Gautier Viaud. Experiential Reflective Learning for Self-Improving LLM Agents, March 2026. URL http://arxiv.org/abs/2603.24639. arXiv:2603.24639 [cs] version: 2.

Parth Asawa, Christopher M. Glaze, Gabriel Orlanski, Ramya Ramakrishnan, Benji Xu, Asim Biswal, Vincent Sunn Chen, Frederic Sala, Matei Zaharia, and Joseph E. Gonzalez. Continual Learning Bench: Evaluating Frontier AI Systems in Real-World Stateful Environments, June 2026. URL http://arxiv.org/abs/2606.05661. arXiv:2606.05661 [cs.AI].

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, et al. Evaluating large language models trained on code, 2021. URL http://arxiv.org/abs/2107.03374.

A. P. Dawid. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society. Series A (General)*, 147(2): 278–290, 1984. doi: 10.2307/2981683.

Vaggelis Dorovatas, Malte Schwerin, Andrew D. Bagdanov, Lucas Caccia, Antonio Carta, Laurent Charlin, Barbara Hammer, Tyler L. Hayes, Timm Hess, Christopher Kanan, Dhireesha Kudithipudi, Xialei Liu, Vincenzo Lomonaco, Jorge Mendez-Mendez, Darshan Patil, Ameya Prabhu, Elisa Ricci, Tinne Tuytelaars, Gido M. van de Ven, Liyuan Wang, Joost van de Weijer, Jonghyun Choi, Martin Mundt, and Rahaf Aljundi. Modular Memory is the Key to Continual Learning Agents, March 2026. URL http://arxiv.org/abs/2603.01761. arXiv:2603.01761 [cs.LG].

Kion Fallah, Silen Naihin, Barak Widawsky, and Qingqing Mao. CLaaS: Continual learning as a service for sample efficient online learning, June 2026. URL http://arxiv.org/abs/2606.05559. arXiv:2606.05559 [cs.LG].

João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013. doi: 10.1007/s10994-012-5320-9.

Anne Harrington, Nayan Saxena, Michael Murphy, Anastasia Borovykh, Zeyu Yun, Sridhar Kamath, Ara Eindra Kyi, Trevor Darrell, Jitendra Malik, and Yutong Bai. When Does Continual Learning Require Learning, July 2026. URL http://arxiv.org/abs/2607.07847. arXiv:2607.07847 [cs.LG].

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, Jiaxin Pei, Julian McAuley, Yejin Choi, and Alex Pentland. MemoryArena: Benchmarking Agent Memory in Interdependent Multi-Session Agentic Tasks, February 2026. URL http://arxiv.org/abs/2602.16313. arXiv:2602.16313 [cs].

Prannay Hebbar, Yogendra Manawat, Samuel Verboomen, Alesia Ivanova, Selvam Palanimalai, Kunal Bhatia, and Vignesh Baskaran. SIA: Self Improving AI with Harness & Weight Updates, May 2026. URL http://arxiv.org/abs/2605.27276. arXiv:2605.27276 [cs.AI].

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Qisheng Hu, Quanyu Long, and Wenya Wang. When Continual Learning Moves to Memory: A Study of Experience Reuse in LLM Agents, April 2026. URL http://arxiv.org/abs/2604.27003. arXiv:2604.27003 [cs.LG].

Jonas Hübotter, Leander Diaz-Bone, Ido Hakimi, Andreas Krause, and Moritz Hardt. Learning on the Job: Test-Time Curricula for Targeted Reinforcement Learning, October 2025. URL http://arxiv.org/abs/2510.04786. arXiv:2510.04786 [cs].

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement Learning via Self-Distillation, February 2026a. URL http://arxiv.org/abs/2601.20802. arXiv:2601.20802 [cs].

Jonas Hübotter, Patrik Wolf, Alexander Shevchenko, Dennis Jüni, Andreas Krause, and Gil Kur. Specialization after Generalization: Towards Understanding Test-Time Training in Foundation Models, February 2026b. URL http://arxiv.org/abs/2509.24510. arXiv:2509.24510 [cs].

Pengcheng Jiang, Jiacheng Lin, Zhiyi Shi, Zifeng Wang, Luxi He, Yichen Wu, Ming Zhong, Peiyang Song, Qizheng Zhang, Heng Wang, Xueqiang Xu, Hanwen Xu, Pengrui Han, Dylan Zhang, Jiashuo Sun, Chaoqi Yang, Kun Qian, Tian Wang, Changran Hu, Manling Li, Quanzheng Li, Hao Peng, Sheng Wang, Jingbo Shang, Chao Zhang, Jiaxuan You, Liyuan Liu, Pan Lu, Yu Zhang, Heng Ji, Yejin Choi, Dawn Song, Jimeng Sun, and Jiawei Han. Adaptation of Agentic AI: A Survey of Post-Training, Memory, and Skills, March 2026. URL http://arxiv.org/abs/2512.16301. arXiv:2512.16301 [cs] version: 3.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?, November 2024. URL http://arxiv.org/abs/2310.06770. arXiv:2310.06770 [cs].

Jinyang Li, Xiaolong Li, Ge Qu, Per Jacobsson, Bowen Qin, Binyuan Hui, Shuzheng Si, Nan Huo, Xiaohan Xu, Yue Zhang, Ziwei Tang, Yuanshuai Li, Florensia Widjaja, Xintong Zhu, Feige Zhou, Yongfeng Huang, Yannis Papakonstantinou, Fatma Ozcan, Chenhao Ma, and Reynold Cheng. SWE-SQL: Illuminating LLM pathways to solve user SQL issues in real-world applications, June 2025. URL https://arxiv.org/abs/2506.18951. arXiv:2506.18951 [cs.DB]; NeurIPS 2025.

Zeyuan Liu, Jeonghye Kim, Xufang Luo, Dongsheng Li, and Yuqing Yang. Exploratory Memory-Augmented LLM Agent via Hybrid On- and Off-Policy Optimization, February 2026. URL https://arxiv.org/abs/2602.23008v2.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent Lightning: Train ANY AI agents with reinforcement learning, August 2025a. URL https://arxiv.org/abs/2508.03680. arXiv:2508.03680 [cs.AI].

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning, January 2025b. URL http://arxiv.org/abs/2308.08747. arXiv:2308.08747 [cs].

Siru Ouyang, Jun Yan, I.-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory, March 2026. URL http://arxiv.org/abs/2509.25140. arXiv:2509.25140 [cs].

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, pp. 1–22, New York, NY, USA, October 2023. Association for Computing Machinery. ISBN 979-8-4007-0132-0. doi: 10.1145/3586183.3606763. URL https://dl.acm.org/doi/10.1145/3586183.3606763.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-Distillation Enables Continual Learning, January 2026. URL http://arxiv.org/abs/2601.19897. arXiv:2601.19897 [cs].

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. *arXiv preprint arXiv:2409.19256*, 2024. doi: 10.48550/arXiv.2409.19256. URL https://arxiv.org/abs/2409.19256.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual Learning of Large Language Models: A Comprehensive Survey, June 2024. URL http://arxiv.org/abs/2404.16789. arXiv:2404.16789 [cs] version: 2.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

David Silver and Richard S. Sutton. Welcome to the era of experience, 2025. Preprint of a chapter in *Designing an Intelligence*, MIT Press.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, September 2022. URL https://arxiv.org/abs/2209.15189. arXiv:2209.15189 [cs.CL].

Jie Sun, Mao Zheng, Mingyang Song, Qiyong Zhong, Gengsheng Li, Zhepei Hong, Chang Wu, Pengfei Liu, Junfeng Fang, and Xiang Wang. EasyOPD: An Easy-to-use On-Policy Distillation Framework for Large Language Models, July 2026. URL http://arxiv.org/abs/2607.11012. arXiv:2607.11012 [cs.CL].

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 2020.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-End Test-Time Training for Long Context, 2025. URL https://arxiv.org/abs/2512.23675.

Rishabh Tiwari, Kusha Sareen, Lakshya A. Agrawal, Joseph E. Gonzalez, Matei Zaharia, Kurt Keutzer, Inderjit S. Dhillon, Rishabh Agarwal, and Devvrit Khatri. Learning, Fast and Slow: Towards LLMs That Adapt Continually, May 2026. URL http://arxiv.org/abs/2605.12484. arXiv:2605.12484 [cs.LG].

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents, July 2024. URL http://arxiv.org/abs/2407.18901. arXiv:2407.18901 [cs].

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory, September 2024. URL http://arxiv.org/abs/2409.07429. arXiv:2409.07429 [cs].

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. Evo-Memory: Benchmarking LLM Agent Test-time Learning with Self-Evolving Memory, May 2026. URL http://arxiv.org/abs/2511.20857. arXiv:2511.20857 [cs.CL].

Cheng-Kuang Wu, Zhi Rui Tam, Chieh-Yen Lin, Yun-Nung Chen, and Hung-yi Lee. StreamBench: Towards Benchmarking Continuous Improvement of Language Agents, October 2024. URL http://arxiv.org/abs/2406.08747. arXiv:2406.08747 [cs].

Tianzhu Ye, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Online Experiential Learning for Language Models, June 2026a. URL http://arxiv.org/abs/2603.16856. arXiv:2603.16856 [cs.CL].

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-Policy Context Distillation for Language Models, March 2026b. URL http://arxiv.org/abs/2602.12275. arXiv:2602.12275 [cs.CL].

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to Discover at Test Time, January 2026. URL https://arxiv.org/abs/2601.16175v2.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models, October 2025. URL https://arxiv.org/abs/2510.04618v3.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners, 2024. URL https://arxiv.org/abs/2308.10144.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning LLM Agents without Fine-tuning LLMs, August 2025. URL http://arxiv.org/abs/2508.16153. arXiv:2508.16153 [cs].

A Experimental and Framework Details

This appendix specifies the adaptation mechanisms, benchmark settings, and implementation choices needed to reproduce the comparisons. The online protocol and scoring rule are given in Section 3 and Algorithm 1.

A.1 Interfaces and adaptation mechanisms

A benchmark registers four contracts: a *world* defines the tasks and shared interaction rules; a *stream* orders those tasks; a *task* supplies its prompt and grading specification; and a *session* executes one isolated attempt. A mechanism implements Prepare to return context messages, Observe to receive completed attempts and outcomes, and, for self-distillation, TeacherContexts to construct hindsight contexts. Save and restore hooks preserve its state across checkpoints. Context preparation and teacher-context construction use a token-budget predicate supplied by the trainer.

For a round's experience $E = \{(\tau_j, o_j)\}_{j=1}^{n}$, weight updates minimize

$$\mathcal{L}(\theta; E) = \lambda_{\mathrm{RL}}\mathcal{L}_{\mathrm{RL}}(\theta; E) + \lambda_{\mathrm{SD}}\mathcal{L}_{\mathrm{SD}}\big(\theta; E, \textsc{TeacherContexts}(E)\big). \tag{2}$$

Both coefficients are zero for the stateless and ICL configurations. GRPO sets $(\lambda_{\mathrm{RL}}, \lambda_{\mathrm{SD}}) = (1, 0)$; SDPO sets them to $(0, 1)$. The LoRA adapter and optimizer state persist across tasks. In hybrids, updates use attempts generated with retained experience in context, and the updated policy generates later context candidates. All learning uses the agent's own attempts and outcomes.

ICL policies. Episodes contain actions, environment observations, and final outcome feedback, replayed oldest first between the system prompt and the current task. All admission selects the highest-reward attempt from each round, including failures. Successful admission selects at most one successful attempt per task, from the first round containing a success. It therefore changes both outcome filtering and the number of episodes a task may contribute. FIFO removes oldest whole episodes until the context fits; Prefix permanently stops admitting episodes at the first overflow. Neither uses summarization, retrieval, or an external judge. Reward ties, successful-attempt selection, and SDPO donor selection follow the trainer's attempt order, which verl rearranges to balance sequence lengths.

GRPO. The policy-gradient objective uses each attempt's reward minus its round's mean reward as the advantage, without standard-deviation normalization. The loss is averaged over generated tokens and uses clipping without a KL penalty toward the base model. Table 4 gives clipping and optimization settings.

SDPO. The self-teacher starts as a copy of the student. After each step it is updated as $\bar{\theta} \leftarrow (1-\beta)\bar{\theta} + \beta\theta$, with $\beta = 0.01$; the two LoRA factors are averaged separately and the teacher is checkpointed. For each attempt, hindsight is a successful sibling from the same round, excluding the attempt itself, or, if none exists and the attempt failed, its own feedback. Attempts with neither contribute zero loss. The teacher sees the student's original prompt, including retained context in hybrids, with hindsight added to the task message. Hindsight is shortened with an explicit marker if necessary; the original prompt is preserved.

The loss is reverse KL, $\mathrm{KL}(\pi_\theta \,\|\, \pi_{\bar{\theta}})$, on the student's generated positions, evaluated over its top 20 tokens and one bucket for the remaining probability mass. Each token's divergence is multiplied by two detached importance weights, each capped at 2: the student-to-generating policy probability ratio and verl's inference-to-trainer probability correction. The loss averages over generated tokens within an attempt and then over all n attempts, including zero contributions. Support and teacher scores are computed once per round and are not refreshed after an update, so SDPO takes one optimizer step per round. This implementation uses no policy-gradient term.

A.2 Worlds, feedback, and task limits

Each world is shuffled once per seed, and each task appears once in the stream. Multiple rounds on that task supply adaptation experience; only the first round contributes to reported success rate and pass@k.

Table 3: The four worlds. Success is each benchmark's verdict (for ScienceWorld, AgentBoard's subgoal annotations); reward is the learning signal of GRPO and the selection rule of the All-admission ICL policies, whereas Successful admission selects by success.

World	Tasks	Action	Turns	Success, reward, feedback
SWE-bench Verified: XARRAY	22 instances from `pydata/xarray`	one block of shell commands per turn, then a submission	40	issue resolved; binary reward; feedback reports how many of the required tests pass
AppWorld: FILE SYSTEM	21 train and dev tasks whose first required app is the file system (7 templates, 3 variants each)	Python code calling app APIs in a persistent interpreter	40	all task requirements pass; binary reward; feedback reports how many requirements passed and the first failure
ScienceWorld: MEASUREMENT	22 AgentBoard instances of the thermometer and melting-point tasks	one text command	30	all annotated subgoals reached; reward is the fraction reached; feedback reports it
BIRD-Critic-SQLite: FINANCIAL	44 tasks on the `financial` SQLite database	SQL queries, then a submitted fix of one or more statements	5	all test cases pass; reward is the fraction passed; feedback reports it and any error

SWE-bench reference patches are never sent to rollout workers; BIRD reference queries are available only to the grader; and ScienceWorld reference action sequences are not generated during rollouts. The SWE-bench turn limit is reduced from the configuration's default of 250 to 40. Attempts also have time limits of 1,800 s in ScienceWorld and 7,200 s in SWE-bench; three consecutive malformed actions end a SWE-bench attempt. One ScienceWorld task, `use-thermometer_7`, has an annotated subgoal that cannot match the environment's output, making full success unattainable on that task.

A.3 CONFIGURATIONS AND IMPLEMENTATION

Per task, GRPO receives 32 attempts and takes eight optimizer steps, whereas SDPO receives 48 attempts and takes three steps. The stateless and ICL-only configurations use the same two-round schedule as GRPO. SDPO's extra round also changes context admission: All can contribute up to three episodes per task, compared with two for ICL and ICL+GRPO. Thus comparisons between GRPO and SDPO concern complete adaptation procedures, not objectives at equal compute or identical experience budgets.

Execution and reproducibility. Our framework extends verl v0.9.0 with a synchronous trainer and a common agent loop. The stream is fixed before training, dataloader shuffling is disabled, and every round is validated before any update or context admission. All expected attempts must have matching identities, finite rewards, and valid endings; otherwise the run stops. Failed tasks are not replaced. Updated weights are synchronized before the next round, and all attempts within a round share context and weights.

Benchmark outcomes become available only after an attempt ends. An attempt terminated by a turn, token, or time limit cannot succeed and receives zero reward, except in ScienceWorld, where it retains partial subgoal reward. Tolerated infrastructure faults are recorded as ungraded zero-reward outcomes; a run is refused when at least four have occurred and they exceed 5% of attempts. A grading failure that leaves a submission without a verdict stops the run. Oversized task prompts are rejected rather than silently truncated.

The self-teacher is a separate inference-only model in the actor worker. Long-context scoring projects only the generated positions needed by the loss, in chunks, avoiding a full sequence-by-vocabulary tensor. The chunked path was checked against full-logit computation, and the SDPO objective and gradients were checked against the reference implementation on CPU.

Table 4: Settings shared by all mechanisms, and settings specific to each family.

Setting	Value
Model	Qwen3.5-9B, non-thinking generation
Attempts per round n	16, sampled at temperature 1.0
Rounds of attempts per task m	2 (stateless control, ICL and GRPO families); 3 (SDPO family)
Token allowance	task prompt 4,096; retained context 75,776; each attempt, including its observations, 24,576; teacher prompt 106,496; model sequence limit 131,072
Turn limits	FILE SYSTEM 40; MEASUREMENT 30; FINANCIAL 5; XARRAY 40
ICL context	replayed conversation with the agent's actions, observations, and the outcome's feedback; no limit on the number of episodes beyond the context allowance
Adapter and optimizer	LoRA rank 32, $\alpha = 32$, every linear layer except the output head; AdamW, learning rate 10^{-5}, constant schedule, weight decay 0.01, gradient clipping 1.0; no KL penalty toward the base model
GRPO family	2 rounds per task; 4 epochs over each round's 16 attempts (4 optimizer steps per round, 8 per task); clip range $[0.8, 1.28]$, with verl's dual-clip bound of 3 for negative advantages; token-mean loss; no standard-deviation normalization of advantages; no correction for the difference between the inference engine's and the trainer's probabilities
SDPO family	3 rounds per task; 1 epoch (1 optimizer step per round, 3 per task); at most one hindsight per attempt, including the donor's observations and shortened if needed to fit the teacher allowance; teacher moving average rate 0.01; reverse KL over the student's top 20 tokens plus a remainder bucket, with per-token importance weights capped at 2; no policy-gradient term

A run seed controls both stream order and generation seeds, although execution is not bitwise reproducible. The recorded order is checked against the loaded stream. Run configurations and input manifests record the model, benchmark, source, and shared budget settings; the auditor checks these and compares logged metrics with recorded outcomes. Best configurations in the reported summaries are selected retrospectively by their five-seed mean score. During a run, adaptation has access only to experience already generated, and each arriving task is scored before learning from its outcomes.

B Additional Results and Uncertainty

The 300 runs cover four worlds, 15 configurations, and five seeds (0–4). Table 2 reports success rate; Table 5 adds pass@16. Both use first-round outcomes, averaged over tasks within a run before summarizing across seeds. Per-seed data are in `results/five_seed_data/`.

Table 5: **Pass@16 for every configuration.** Percentage of tasks with at least one success among 16 first-round attempts, mean ± sample SD across five seeds. Each entry is computed from task-level outcomes before averaging.

Configuration	File System	Measurement	Financial	Xarray
Baseline	89.5 ± 6.2	52.7 ± 6.1	64.5 ± 1.2	40.9 ± 4.5
ICL-FIFO	86.7 ± 9.2	69.1 ± 6.7	63.6 ± 1.6	71.8 ± 5.9
ICL-FIFO (Succ.)	96.2 ± 6.2	69.1 ± 14.2	70.5 ± 3.6	63.6 ± 3.2
ICL-Prefix	89.5 ± 12.8	68.2 ± 6.4	61.4 ± 3.6	68.2 ± 6.4
ICL-Prefix (Succ.)	94.3 ± 2.1	72.7 ± 5.6	67.3 ± 2.0	62.7 ± 8.7
GRPO	87.6 ± 5.4	60.9 ± 5.2	67.3 ± 4.7	50.0 ± 9.1
ICL-FIFO + GRPO	86.7 ± 2.1	60.9 ± 9.4	67.7 ± 3.7	75.5 ± 5.2
ICL-FIFO (Succ.) + GRPO	94.3 ± 6.2	73.6 ± 5.9	67.7 ± 4.1	70.0 ± 7.6
ICL-Prefix + GRPO	90.5 ± 6.7	68.2 ± 9.1	67.3 ± 2.6	74.5 ± 6.9
ICL-Prefix (Succ.) + GRPO	93.3 ± 5.4	79.1 ± 2.5	70.0 ± 4.9	72.7 ± 6.4
SDPO	83.8 ± 8.7	61.8 ± 4.1	68.6 ± 5.4	47.3 ± 5.2
ICL-FIFO + SDPO	77.1 ± 14.0	74.5 ± 6.1	67.7 ± 6.9	73.6 ± 3.8
ICL-FIFO (Succ.) + SDPO	95.2 ± 4.8	77.3 ± 10.7	69.1 ± 2.6	71.8 ± 7.5
ICL-Prefix + SDPO	85.7 ± 8.2	78.2 ± 8.7	62.3 ± 3.4	70.9 ± 12.3
ICL-Prefix (Succ.) + SDPO	94.3 ± 4.0	75.5 ± 5.2	65.0 ± 1.2	71.8 ± 7.5

Aggregation and paired comparisons. Table 6 and the pass@k analysis average the same four ICL policies within each seed; Average weights worlds equally. This differs from the selected configurations in Figure 1. Differences are paired by seeded task order. The five seeds are the independent replicates; policies and attempts are not additional replicates. Seed variation includes both task order and sampling.

Table 6: **Paired hybrid gains over ICL.** Differences in percentage points, mean ± sample SD across five seeds, after averaging the four ICL policies within each seed. Seeds > 0 counts positive differences. Two-sided paired t-tests use $df = 4$; bold denotes uncorrected $p < 0.05$.

	ICL + GRPO − ICL			ICL + SDPO − ICL		
World	Δ (pp)	seeds > 0	p	Δ (pp)	seeds > 0	p
Success rate						
Xarray	$+2.4 \pm 2.1$	4/5	0.065	0.0 ± 3.5	3/5	0.987
File System	$+0.2 \pm 3.4$	4/5	0.891	-5.2 ± 6.0	1/5	0.124
Measurement	$+5.4 \pm 4.6$	5/5	0.058	$+4.4 \pm 4.3$	4/5	0.085
Financial	$+4.0 \pm 1.2$	5/5	**0.002**	$+0.4 \pm 1.4$	3/5	0.537
Average	$+3.0 \pm 2.0$	5/5	**0.028**	-0.1 ± 2.4	2/5	0.927
pass@16						
Xarray	$+6.6 \pm 6.2$	3/5	0.078	$+5.5 \pm 7.7$	3/5	0.189
File System	-0.5 ± 2.7	2/5	0.717	-3.6 ± 5.2	1/5	0.199
Measurement	$+0.7 \pm 4.3$	2/5	0.741	$+6.6 \pm 3.5$	5/5	**0.014**
Financial	$+2.5 \pm 2.0$	5/5	**0.049**	$+0.3 \pm 2.2$	2/5	0.749
Average	$+2.3 \pm 3.0$	3/5	0.161	$+2.2 \pm 3.0$	4/5	0.172

A Holm correction over the eight per-world success-rate comparisons leaves only GRPO on Financial below 0.05. With five seeds, an exact sign-flip test cannot yield $p < 0.0625$. SDs describe seed variation, not confidence intervals. Pass@16 measures success within the sampled budget, not unlimited solvability.

C FINANCIAL INTERACTION ANALYSIS

These diagnostics support Section 5.1 and use first-round attempts only. A turn is one model generation, including the final submission; one generation may contain multiple tool operations. Turns measure neither token cost nor wall-clock cost. FINANCIAL allows up to four querying turns before submission on the fifth; a one-turn submission does not query the database.

Table 7: FINANCIAL**: attempt length, success rate, and reward signal.** Best configuration per family, as in Figure 3. Turns: means in the first and last stream quarters. Share: one-turn submissions and five-turn cutoffs without submission. Success rate: pooled over seeds by attempt length. Advantage: reward minus the same round's mean reward, averaged within seed and then across seeds.

	Turns		Share (%)		Success rate (%)		Advantage	
Family	first	last	1 turn	cut off	1 turn	2–4 turns	1 turn	2–4 turns
Baseline	3.9	4.1	8.0	28.6	13.5	52.9	−0.09	+0.09
ICL	3.3	1.6	34.3	5.8	23.1	43.9	−0.07	+0.05
SDPO	2.7	2.0	47.6	8.9	17.1	48.2	−0.07	+0.11
ICL + SDPO	2.7	1.5	57.7	4.0	25.0	47.4	−0.05	+0.08
GRPO	3.7	3.4	8.0	13.6	10.6	49.3	−0.14	+0.06
ICL + GRPO	3.3	2.7	12.2	7.3	19.3	44.3	−0.09	+0.02

ICL reduces the baseline's cutoff frequency but increasingly submits in one turn. With GRPO, attempts remain longer near the end of the stream. One-turn attempts have below-average reward and two- to four-turn attempts above-average reward in every family, consistent with GRPO favoring interaction before submission. These conditional groups can contain different tasks: the comparison does not identify a causal effect of adding queries to an otherwise identical attempt.

Table 8: FINANCIAL**: robustness across ICL policies.** Mean turns in the first and last stream quarters, and the paired success-rate gain from GRPO (percentage points, mean over five seeds). Seeds > 0 counts positive gains.

	ICL turns		ICL+GRPO turns	Success rate gain from GRPO	
ICL policy	first	last	last	pp	seeds > 0
FIFO, All	2.5	1.1	2.3	+5.9	5/5
FIFO, Successful	3.2	1.7	2.9	+0.4	3/5
Prefix, All	2.2	1.3	2.5	+6.1	5/5
Prefix, Successful	3.3	1.6	2.7	+3.6	4/5

Table 8 shows that shorter late-stream ICL attempts and positive mean GRPO gains occur under all four policies, although the gain is not positive in every seed under every policy.

Table 9: **Interaction length across worlds.** Mean turns per first-round attempt, including unsuccessful attempts, mean ± sample SD across seeds. ICL families average the four policies within each seed. Turn limits and action semantics differ across worlds, so comparisons are within a world.

Family	FILE SYSTEM	MEASUREMENT	FINANCIAL	XARRAY
Baseline	27.3 ± 0.3	28.6 ± 0.1	4.0 ± 0.0	36.7 ± 0.2
ICL	12.2 ± 1.7	22.6 ± 1.4	1.9 ± 0.1	32.3 ± 0.4
GRPO	27.5 ± 0.9	26.2 ± 0.6	3.6 ± 0.4	35.5 ± 0.7
ICL + GRPO	11.7 ± 0.9	21.3 ± 0.6	2.6 ± 0.2	30.5 ± 0.8
SDPO	26.6 ± 0.4	27.9 ± 0.2	2.2 ± 0.1	35.5 ± 0.6
ICL + SDPO	10.9 ± 1.6	21.5 ± 1.4	1.6 ± 0.1	28.5 ± 1.5

The cross-world summary supports the broader distinction between shorter execution and better adaptation: ICL reduces turns in FILE SYSTEM and FINANCIAL, whereas weight updates do not consistently shorten attempts. SDPO can shorten execution without improving success rate (Table 2).

D PASS@k AND ICL-POLICY SENSITIVITY

For a task with c successful attempts among $n = 16$, we estimate

$$\widehat{\text{pass@}}k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}, \qquad 1 \le k \le n, \tag{3}$$

with $\binom{n-c}{k} = 0$ when $k > n - c$ (Chen et al., 2021). We compute this within each task and configuration before averaging over tasks and the four ICL policies within each seed. Thus policy averaging does not increase the attempt budget. Pass@1 equals success rate, and pass@16 is the observed fraction of tasks with any success.

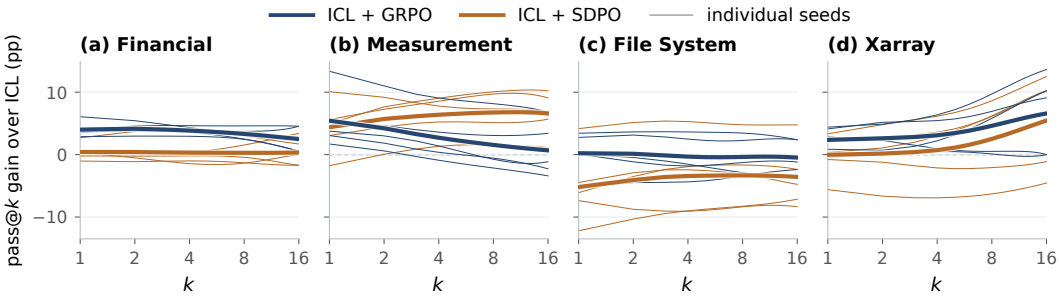

Figure 6: **Pass@k gain from adding weight updates to ICL.** Hybrid minus ICL, paired by seed, averaged over tasks and the same four ICL policies. Thin lines show individual seeds and thick lines their mean. Other-world panels provide context for the XARRAY analysis in Figure 4.

Table 10: XARRAY**: sensitivity to ICL policy.** Hybrid minus the corresponding ICL policy, percentage points, paired mean ± sample SD across five seeds. Mean pass@16 gains exceed mean success-rate gains under each policy.

ICL policy	GRPO Δ success rate	GRPO Δ pass@16	SDPO Δ success rate	SDPO Δ pass@16
FIFO	-0.5 ± 6.8	$+3.6 \pm 10.4$	-1.0 ± 2.8	$+1.8 \pm 8.9$
FIFO (Succ.)	$+1.5 \pm 4.3$	$+6.4 \pm 8.9$	$+1.1 \pm 4.4$	$+8.2 \pm 7.5$
Prefix	$+4.3 \pm 1.1$	$+6.4 \pm 8.3$	-1.9 ± 5.6	$+2.7 \pm 13.9$
Prefix (Succ.)	$+4.1 \pm 3.8$	$+10.0 \pm 10.9$	$+1.8 \pm 3.5$	$+9.1 \pm 7.2$

On XARRAY, the gain at pass@16 exceeds the gain at pass@1 in four of five seeds for each updater; a positive pass@16 gain itself occurs in three of five seeds. The former is the seed-level comparison underlying Section 5.2. These observations concern finite-budget success and do not by themselves identify a change in solution diversity or guarantee gains on every seed.

E CONTEXT RETENTION AND THE VALUE OF WEIGHT UPDATES

Table 11 summarizes explicit history under the 75,776-token allowance, with weights fixed and All admission. FIFO evicts old episodes; Prefix stops admission at overflow.

Table 11: **Context capacity, weights fixed and All admission.** Five-seed means. Evictions: preparations evicting episodes. Freeze: first task position where admission stops. Retained: distinct prior tasks before the final task's first round; Available: total prior tasks.

World	FIFO evictions (%)	Prefix freeze	Retained (FIFO / Prefix)	Available
FILE SYSTEM	31.0	7.8	12.8 / 6.8	20
MEASUREMENT	11.4	17.4	18.6 / 16.4	21
FINANCIAL	46.4	16.4	18.8 / 15.4	43
XARRAY	47.7	4.2	6.4 / 3.2	21

Reconstructed task histories are checked against telemetry (`results/memory/`). Under Successful admission, which limits contributions to one episode per task, MEASUREMENT with fixed weights never evicts or freezes.

Paired training gains. With All admission, average GRPO gains are positive in 5/5 seeds for Prefix and 3/5 for FIFO; Prefix's gain is larger in all five. Their mean gain difference is $+5.2 \pm 2.4$ points, versus -0.1 ± 4.4 under Successful admission (Table 12). The interaction depends on admission policy. A larger gain need not imply higher final performance: Prefix starts from a lower untrained average under All admission.

Table 12: **GRPO gains by retention and admission policy.** Seed-paired mean $\pm$ SD, percentage points. Average weights worlds equally. p: two-sided paired t-test on the gain difference ($df = 4$); bold: uncorrected $p < 0.05$. Neither absolute Prefix $-$ FIFO comparison has $p < 0.05$ in any row.

	GRPO gain		Prefix − FIFO		Gain difference	
World	FIFO	Prefix	no GRPO	GRPO	Prefix − FIFO	p
All *admission*						
XARRAY	-0.5 ± 6.8	$+4.3 \pm 1.1$	-3.7 ± 7.3	$+1.1 \pm 4.8$	$+4.8 \pm 6.7$	0.182
FILE SYSTEM	-0.8 ± 7.3	$+6.8 \pm 5.4$	-4.3 ± 10.7	$+3.3 \pm 9.2$	$+7.6 \pm 7.1$	0.075
MEASUREMENT	-0.2 ± 6.9	$+8.0 \pm 6.3$	-2.3 ± 4.9	$+5.9 \pm 8.1$	$+8.2 \pm 9.1$	0.115
FINANCIAL	$+5.9 \pm 3.6$	$+6.1 \pm 4.1$	-0.7 ± 2.4	-0.4 ± 5.0	$+0.2 \pm 4.2$	0.909
Average	$+1.1 \pm 3.4$	$+6.3 \pm 2.2$	-2.8 ± 2.6	$+2.4 \pm 2.9$	$+5.2 \pm 2.4$	**0.009**
Successful *admission*						
XARRAY	$+1.5 \pm 4.3$	$+4.1 \pm 3.8$	-1.7 ± 1.5	$+1.0 \pm 5.1$	$+2.7 \pm 5.4$	0.332
FILE SYSTEM	-1.5 ± 7.1	-3.6 ± 9.7	$+0.2 \pm 5.2$	-1.8 ± 8.8	-2.1 ± 11.3	0.701
MEASUREMENT	$+9.1 \pm 11.1$	$+4.9 \pm 2.0$	$+7.2 \pm 10.0$	$+2.9 \pm 3.9$	-4.3 ± 9.6	0.379
FINANCIAL	$+0.4 \pm 2.5$	$+3.6 \pm 6.1$	$+0.2 \pm 3.7$	$+3.4 \pm 3.8$	$+3.2 \pm 7.1$	0.375
Average	$+2.4 \pm 3.6$	$+2.2 \pm 2.8$	$+1.5 \pm 2.9$	$+1.3 \pm 1.6$	-0.1 ± 4.4	0.953

Performance after admission stops. On XARRAY with All admission, all Prefix runs with weights fixed or GRPO freeze by position 5. Over positions 6–22, GRPO raises the Prefix success rate from 39.2% to 43.5% ($+4.3 \pm 2.4$ points); FIFO gains -0.4 ± 8.0 points. The window is chosen from telemetry, not performance. SDPO does not reproduce the Prefix success-rate gain (`results/retention_focus/`).

Interpretation. Runs develop different policies and retained contexts. These comparisons do not establish preservation of excluded episodes or more stable optimization. Post-freeze gains can reflect earlier updates; isolating subsequent learning requires a control that freezes weights at the same cutoff. Paired SDs measure seed variation, not temporal stability.