ADAM-TTT: ADAPTIVE AGENT MEMORY VIA TEST-TIME TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon language agents must continually adapt as tasks evolve during interaction. Agent memory supports adaptation at deployment by making relevant past experience available during interaction. Existing methods, however, operate mainly through contextual memory—retrieved trajectories or summaries placed in the current prompt—while the policy parameters remain fixed. This confines adaptation to information repeatedly supplied in the prompt, even when useful guidance should persist across subsequent decisions. We investigate whether retrieved experience can also supervise online policy adaptation, allowing memory to influence both the agent's context and its parameters. Although model parameters are conventionally viewed as long-term memory, we use a lightweight adapter as short-term parametric memory: it is updated within an episode, replaced when guidance changes, and never modifies the frozen base model. We introduce **ADAM-TTT** (Adaptive Agent Memory via Test-Time Training), a framework that retrieves successful experiences and distills them into a compact procedural memo relevant to the agent's current task state. The procedural memo guides action generation as contextual memory; together with a compact live-progress summary, it forms the next-token-prediction target used to update a LoRA adapter at test time. This combines immediate contextual guidance with temporary knowledge encoded in the adapter. The policy can retain this paired memory state, update it with new experience, or reset it by clearing the memo and unloading the adapter, allowing adaptation to remain responsive as the episode progresses while the base model stays frozen. On ALFWorld, **ADAM-TTT** achieves a 5.0% relative improvement over the strongest memory baseline, while test-time training contributes only 2.5% of estimated parameter-level compute. These results demonstrate the benefit of coupling contextual memory with short-term parametric adaptation, providing a new mechanism for continual adaptation from experience after deployment.

1 INTRODUCTION

Test-time adaptation asks a deployed model to change its behavior when deployment reveals conditions unavailable during training, without retraining the full model (Xiao & Snoek, 2024). This is acute for language agents: an agent must turn a changing sequence of observations into decisions, often over many steps, while the useful strategy shifts with the task state (Yao et al., 2023; Shinn et al., 2023; Zhang et al., 2026b). The central challenge is to turn accumulated experience into behavior that remains useful as the task state changes. Agent memory offers one practical route by carrying experience across interactions (Zhao et al., 2024; Majumder et al., 2023; Ouyang et al., 2025; Wei et al., 2025; Jiang et al., 2026).

Most agent-memory systems leave the policy frozen and change behavior only through the prompt. They retrieve trajectories, distilled insights, or reusable workflows from an external store and place the selected information in context (Zheng et al., 2024a; Zhao et al., 2024; Wang et al., 2025; Ouyang et al., 2025). Other systems improve what is stored or how memory is generated, but still expose the resulting knowledge through the prompt at inference time (Wei et al., 2025; Cai et al., 2026). This is effective for decisions that can be resolved from a single prompted context. It is less satisfactory when state-relevant guidance must affect several later decisions, because the policy that interprets new observations remains fixed. State-conditioned memory construction makes prompt guidance more relevant as an episode evolves (Zhang et al., 2026a), but does not change this limitation.

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.

Test-time training (TTT) offers a complementary route: it changes the deployed model using data available during interaction. Previous work adapts language models using retrieved text, selected examples, self-generated data, or relevant spans from a long context (Hardt & Sun, 2024; Hübotter et al., 2025; Zweiger et al., 2025; Bansal et al., 2025; Zhu et al., 2026). Agent-focused variants learn from exploration, interaction prefixes, generated summaries, current-session experience, or online action feedback (Chen et al., 2026; Acikgoz et al., 2026; Wang et al., 2026b; Ren et al., 2026; Yu et al., 2026; Lou et al., 2026). What remains unclear is how experience stored across episodes should be represented when it must serve both as immediate guidance and as the signal for an online parameter update.

This question concerns three locations where an agent can retain or access experience (Jiang et al., 2026). *External memory* stores experience outside the model across episodes. *Contextual memory* places selected information in the current prompt. *Parametric memory* encodes information in model parameters. Although parameters usually represent durable knowledge, a lightweight adapter can serve as temporary, replaceable parametric memory within an episode (Hu et al., 2022). It can influence several later actions, then be replaced when the agent needs new guidance, while the base model stays frozen.

We introduce **AdaM-TTT**, Adaptive Agent Memory via Test-Time Training. At an update, the agent retrieves successful trajectories from external memory and distills them into a compact procedural memo conditioned on the current task state. The procedural memo enters the action prompt as contextual memory. In parallel, a live-progress summary captures the agent's current state, completed work, and unresolved blockers from the live prompt. Together, the summary and procedural memo form the next-token-prediction target for a fresh LoRA adapter, creating short-term parametric memory. This separates two roles: retrieved experience supplies reusable procedural guidance in context, while the training target combines that procedure with the agent's current progress.

The policy controls the lifetime of this memory state. At each decision point, it chooses KEEP, which reuses the active memo and adapter; UPDATE, which retrieves new experience, constructs a new memo and progress summary, and trains a fresh adapter; or RESET, which clears the memo and unloads the adapter. UPDATE regenerates the action at the same environment step, whereas RESET returns action generation to the frozen base model. These operations let the agent adapt within an episode without accumulating all updates into one long-lived parameter state.

On ALFWorld, **AdaM-TTT** improves over the strongest memory baseline by 5.0% relative. A controlled ablation shows that supplying the procedural memo through both contextual and parametric pathways outperforms either pathway alone, indicating that the two forms of memory are complementary. Estimated parameter compute further attributes only 2.5% of the method's total compute to test-time training, indicating that the added training cost is small relative to inference.

Our contributions are:

- We formulate agent adaptation as coordinated external, contextual, and short-term parametric memory, using a replaceable lightweight adapter as episode-level memory.
- We introduce a procedural memo for contextual guidance and a live-progress summary for test-time supervision, with policy-controlled retention, update, and reset operations.
- We implement asynchronous training and serving for per-environment adapters, and show improved performance over memory baselines, complementary contextual and parametric pathways, and a small test-time-training compute share.

2 Related Work

Memory-augmented LLM agents. Agent memory ranges from concrete demonstrations to abstract lessons. Synapse uses trajectory exemplars with state abstraction (Zheng et al., 2024a); ExpeL extracts insights and retrieves successful experiences (Zhao et al., 2024); and Agent Workflow Memory abstracts reusable workflows (Wang et al., 2025). Reflexion and CLIN revise textual knowledge from prior trials (Shinn et al., 2023; Majumder et al., 2023), while ReasoningBank distills experience from both successful and failed attempts (Ouyang et al., 2025). More recent systems also adapt what is written to memory or when memory is refreshed during execution: Evo-Memory's ReMem interleaves acting with memory refinement (Wei et al., 2025), and MemoPilot trains a memory

generator through reinforcement learning while keeping the player fixed at test time (Cai et al., 2026). AdaMEM retrieves historical trajectories and generates state-conditioned strategy memory for ongoing execution (Zhang et al., 2026a). Adaptive Minds exposes existing specialized adapters as tools (Shekar & Krishnan, 2025), and SLAaaT lets an agent switch its own adapter within a trajectory (Ge, 2026). These works show that adapters can be controllable resources during agent execution. AdaM-TTT differs by training a new adapter from a state-conditioned procedural memo derived from retrieved experience, while retaining that memo as contextual guidance.

Test-time learning for LLM agents. TTT-NN fine-tunes on text retrieved for the current input (Hardt & Sun, 2024); SIFT selects adaptation examples to reduce uncertainty and redundancy (Hübotter et al., 2025); and SEAL learns to generate self-edits that support subsequent adaptation (Zweiger et al., 2025). For long contexts, qTTT updates query projections on sampled spans while reusing cached keys and values (Bansal et al., 2025). Self-Guided TTT selects question-relevant spans for next-token training and retains the original context for answering (Zhu et al., 2026). These methods show that the choice of training text matters. Chen et al. (2026) adapt LLM agents through environment interaction; TT-SI identifies uncertain inputs, generates related examples, and temporarily adapts the model (Acikgoz et al., 2026); and OLIVIA updates an action-selection layer over frozen representations from step-level feedback (Yu et al., 2026). AgentOdyssey examines learning and memory in long-horizon text-game agents (Zhang et al., 2026b). Two particularly close methods are aTTT, which supports LoRA training on interaction prefixes or progress summaries with repetition-aware token weighting (Wang et al., 2026b), and TMEM, which trains on current-session extractions when the context budget is exceeded (Ren et al., 2026). In TMEM's instantiated update rule, extracted supervision is absorbed into continuing fast weights and textual memory is cleared. Our procedure instead retrieves experience from prior episodes, preserves the state-conditioned procedural memo in context, and replaces the adapter with one trained afresh on that memo. Our distinction is that prior-episode experience is distilled into one memo that remains in context while a newly trained adapter governs later actions. The adapted model continues to act in an evolving environment.

Self-improving LLM agents. Interaction can supply reward-based adaptation signals: TTED learns from trials in decomposed web environments (Li et al., 2026), while ARTS and TTT-Discover optimize scientific search using experiment outcomes (Juneja et al., 2026; Yuksekgonul et al., 2026). TTRL uses majority-derived rewards on unlabeled reasoning problems (Zuo et al., 2025); TTPO combines agreement-conditioned self-distillation with penalties for disagreeing rollouts (Wang et al., 2026a). SDPO uses feedback-conditioned self-distillation, including a test-time setting (Hübotter et al., 2026). EMPO^2 internalizes memory-guided behavior through hybrid policy optimization during agent training (Liu et al., 2026). Our method instead trains directly on procedural-memo tokens. It uses neither online task rewards nor a teacher distribution over action trajectories.

3 METHOD

3.1 SETTING AND ADAPTATION STATE

We consider a partially observed sequential task with goal g. At step t, the agent observes o_t, selects a textual action a_t, and receives the ensuing observation. Let H_t denote its interaction history before that decision and let $x_t = \mathrm{Prompt}(g, o_t, H_t)$ be the live agent prompt. This is the prompt available to the agent; we do not assume access to the underlying environment state.

The agent has a frozen base language model with parameters θ_0 and access to an external historical trajectory bank $\mathcal{B}$. Before step t, its adaptation state is (m_{t-1}, ϕ_{t-1}): a contextual procedural memo m and an optional LoRA adapter ϕ that serves as short-term parametric memory. A live-progress summary u_t is constructed from the current prompt only when a new adapter is trained. We write $p_{\theta_0,\phi}$ for the adapted language model and $\phi = \varnothing$ for base-model inference. For each adapted linear layer, LoRA keeps the pretrained weight matrix W_0 frozen and learns a low-rank update ΔW, yielding the effective weight matrix $W = W_0 + \Delta W$; ϕ denotes the collection of these trainable updates (Hu et al., 2022). The history of adaptation decisions is recorded separately as D_t.

The persistence boundary is deliberate. The external bank persists across episodes. A memo is created at an update and remains in the action context until a later update or reset; its paired adapter

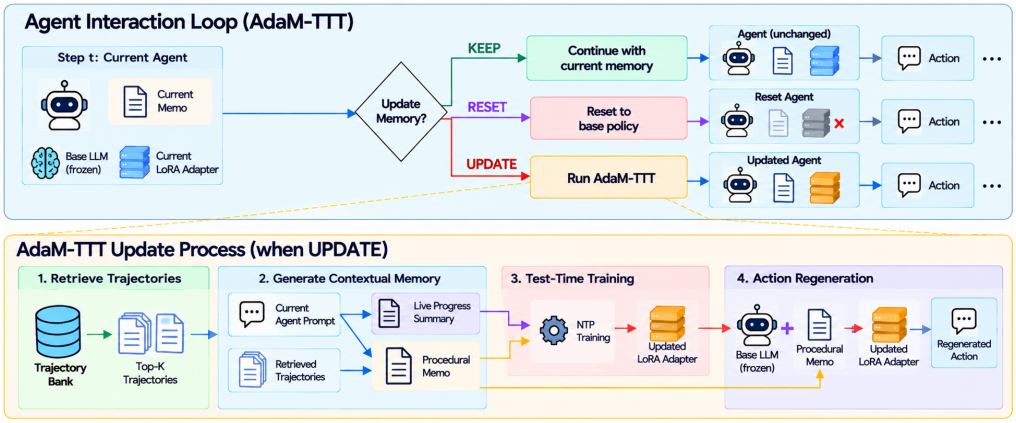


Figure 1: AdaM-TTT maintains external, contextual, and short-term parametric memory within one agent policy. Successful trajectories are distilled with the current context into a procedural memo for action generation, while a live-progress summary and that memo supervise a fresh LoRA adapter. The policy can retain this paired state, update it with new summaries and an adapter, or reset it by clearing the memo and unloading the adapter.

remains loaded over the same interval. UPDATE replaces both, whereas RESET clears the memo and unloads the adapter. Thus, we repurpose parametric memory from a conventionally long-lived store of learned knowledge into a lightweight episode-level state that can be refreshed quickly.

3.2 State-Conditioned Memory Construction

When an update is requested, a retrieval service queries the external trajectory bank using the current agent context and returns up to K relevant trajectories:

$$\mathcal{R}_t = \mathrm{Retrieve}_K(x_t; \mathcal{B}). \tag{1}$$

A summarization call then reads the live prompt and all retrieved trajectories together to produce one consolidated procedural memo,

$$m_t = S_{\mathrm{proc}}(x_t, \mathcal{R}_t). \tag{2}$$

The prompt asks for reusable action ordering, state transitions, and constraints relevant to the present task. The memo therefore describes how past experience may apply now, rather than reproducing each retrieved trajectory independently. In the experiments, the bank contains successful historical trajectories. The training objective itself does not require that restriction; it only requires a memo constructed from retrieved experience and the current context.

Memo construction uses the configured base model without the environment's active adapter. The memo is appended to x_t for action generation; raw trajectories are used only during memo construction.

3.3 Test-Time Training on Procedural and Progress Memory

At an UPDATE, a second summarization call reads only the live prompt and produces a compact progress summary,

$$u_t = S_{\mathrm{prog}}(x_t), \tag{3}$$

which records current task progress, useful outcomes, and unresolved blockers. We serialize the live-progress summary and procedural memo into one training sequence $z_t = \mathrm{Serialize}(u_t, m_t)$. A new adapter is trained with the causal language-modeling objective

$$\mathcal{L}_t(\phi) = -\frac{1}{|z_t|} \sum_{j=1}^{|z_t|} \log p_{\theta_0,\phi}(z_{t,j} \mid z_{t,<j}). \tag{4}$$

The token sequence includes the serialization prefix and heading. The raw prompt and trajectories do not enter this training example directly: the prompt influences training through u_t and m_t, and

retrieved trajectories influence it through m_t. Equation 4 therefore trains on a compact account of both current progress and reusable procedure rather than on full trajectories.

Every update initializes fresh LoRA factors on the same base checkpoint and performs a bounded number of optimization steps:

$$\phi_t = \mathrm{Train}_{\mathrm{LoRA}}(\theta_0, z_t; \phi_{\mathrm{init}}). \tag{5}$$

The previous adapter does not initialize this optimization. Adaptation persists between updates through the currently loaded adapter, but successive updates do not accumulate gradients into a continuing adapter. Replacement lets each update specialize to its newly constructed memo. It does not guarantee a better policy: successful source trajectories do not certify a generated memo, and the training objective does not verify it.

3.4 Policy-controlled retention and replacement

At each step, a decision prompt presents the live context, current memo, adapter status, and adaptation history to the active policy. One generation produces a proposed environment action $\widetilde{a}_t$ and a discrete decision d_t:

$$(\widetilde{a}_t, d_t) \sim p_{\theta_0,\phi_{t-1}}\big(\cdot \mid x_t, m_{t-1}, D_t, \mathrm{status}(\phi_{t-1})\big), \quad d_t \in \{\textsc{keep}, \textsc{update}, \textsc{reset}\}. \tag{6}$$

This is a prompted decision by the acting policy, not a separately trained meta-controller or a measured uncertainty threshold. The operations are:

- **KEEP:** retain (m_{t-1}, ϕ_{t-1}) and execute the proposed action if it is parseable; otherwise request an action from the retained state.
- **UPDATE:** retrieve trajectories, construct m_t and u_t, train a fresh adapter, and replace the previous memo and adapter. Regenerate the environment action after the replacement.
- **RESET:** clear the active memo, unload the active adapter, and regenerate the action from the frozen base model.

After a successful update, action generation uses

$$a_t \sim p_{\theta_0,\phi_t}(\cdot \mid x_t, m_t). \tag{7}$$

Regeneration ensures that a changed memory state can affect the action taken at the same environment step. RESET instead produces $a_t \sim p_{\theta_0}(\cdot \mid x_t)$ with no memo or adapter. We force an initial update at step zero. In the main evaluation, the policy then selects KEEP, UPDATE, or RESET before each action; fixed refreshes are used only in the controlled ablation.

3.5 Implementation details: TTT infrastructure

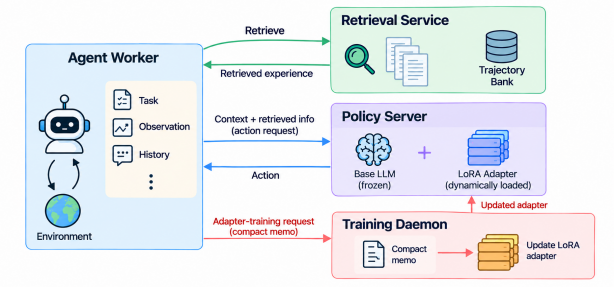


Figure 2: TTT infrastructure for an UPDATE. The worker coordinates retrieval, inference, training, and adapter loading; each environment keeps separate memo and adapter state.

The system has four services: an agent worker, retrieval service, policy server, and training daemon (Figure 2). Each worker owns the environment history, current procedural memo, adaptation decisions, and active adapter identifier. The retrieval service returns historical trajectories; the policy server runs the frozen backbone with the selected adapter using vLLM (Kwon et al., 2023); and the daemon

trains LoRA adapters through LlamaFactory (Zheng et al., 2024b). Memo construction and live-progress summarization use the configured base-model endpoint. The action, memo-construction, live-progress, decision, and training-serialization templates are reproduced in Appendix A. On UPDATE, the worker retrieves trajectories, constructs a procedural memo and live-progress summary, and submits their serialization as one compact causal-language-modeling example to the daemon. The daemon starts a fresh adapter from the frozen base checkpoint and returns an adapter artifact. The worker then replaces the previously loaded adapter and regenerates the current action with the new memo and adapter. On RESET, it clears the memo and unloads the environment's adapter before requesting an action from the frozen base model. GPU slots can process training requests from different workers concurrently, but an adapting environment waits for its own training and loading to finish; per-environment adapter identifiers keep these routing states separate.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Agent benchmark. Following the evaluation protocol of AdaMEM (Zhang et al., 2026a), we evaluate on its 140-task ALFWorld evaluation split (Shridhar et al., 2021). ALFWorld is a text-based embodied benchmark in which an agent navigates household environments and manipulates objects to satisfy high-level goals. The evaluation spans six task categories: Pick, Pick Two, Look, Heat, Cool, and Clean. An episode is successful only if the requested goal is completed within 50 environment steps. We report success rate as the mean and sample standard deviation over three independent runs.

Backbone and agent framework. All methods use Qwen3-4B-Instruct-2507 as the acting model and follow the ReAct interaction format (Yao et al., 2023), interleaving reasoning with environment actions. We use a single-model setup: the frozen base model generates procedural memos and the same backbone generates environment actions, optionally with the active adapter. Thus improvements cannot be attributed to distillation from a stronger teacher model. Generation uses temperature 0.7, and each repetition evaluates all 140 tasks.

Memory bank and retrieval. We use the same long-term trajectory memory as AdaMEM (Zhang et al., 2026a). The bank contains 1,596 successful trajectories collected from the ALFWorld training set. All memory-based methods access this identical bank, controlling the available experience across methods. Retrieval uses Qwen3-Embedding-4B representations and returns the top $K = 8$ trajectories relevant to the current state. Because each memory-based method retrieves from this bank, the comparisons hold the available source experience fixed; methods differ in how they represent and use it.

Compared methods. We compare with a memory-free base agent and three memory-based baselines that use the same trajectory bank. **Synapse** retrieves a successful trajectory and supplies it as an in-context exemplar, testing whether a raw past solution is sufficient (Zheng et al., 2024a). **ReasoningBank** retrieves distilled reasoning memories from prior experience, representing a compact but task-agnostic textual memory (Ouyang et al., 2025). **AdaMEM** retrieves trajectories and dynamically synthesizes a state-relevant strategy memo, providing the closest contextual-memory comparison (Zhang et al., 2026a). These baselines span no memory, raw trajectory retrieval, distilled memories, and state-conditioned procedural memories. **ADAM-TTT** retains a procedural memo in context and additionally creates temporary parametric memory through a LoRA update. It performs a mandatory initialization at step zero; afterward, the acting policy selects KEEP, UPDATE, or RESET before each environment action.

Test-time training. Each update trains a fresh rank-32 LoRA adapter for one epoch with learning rate 5×10^{-5} using the causal language-modeling objective in Section 3.3. Each update initializes fresh LoRA factors on the frozen base checkpoint; the previous adapter is not used for initialization. Prefix caching is disabled because the active adapter can change during an episode. The training daemon and policy server execute separately, so workers that are not awaiting an adapter can continue serving their own environments.

Method	Memory representation			Success rate (%) ↑
	Param.	Context.	External	
No Memory	Frozen	None	None	45.2 ±1.8
Synapse	Frozen	Trajectory	Trajectory	51.7 ±2.2
ReasoningBank	Frozen	Memo	Memo	55.0 ±0.7
AdaMEM	Frozen	Memo	Trajectory	62.1 ±1.9
ADAM-TTT	Adaptive	Memo	Trajectory	**65.2** ±2.2

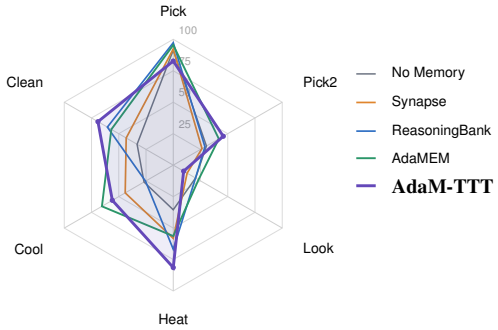


Figure 3: ALFWorld success with retrieval from successful trajectories. **Left:** aggregate success rate (mean ± sample standard deviation over three repetitions). **Right:** mean success by task category.

5 RESULTS

5.1 MAIN COMPARISON

Figure 3 compares methods that share the same source bank but differ in where retrieved experience is represented. **ADAM-TTT** achieves the highest aggregate success rate, 65.2%, exceeding AdaMEM by 3.1 percentage points (5.0% relative). The left panel therefore supports the central claim that a procedural memo is more useful when it can guide the current action and update a short-term adapter. The radar chart shows that the gain is not confined to one task family: **ADAM-TTT** is strongest on Heat, Clean, and Pick Two, while AdaMEM leads on Cool and the memory-free agent leads on Pick.

The prompt interfaces behind this comparison are listed verbatim in Appendix A. A step-level trace of a successful episode, including its procedural memo, update points, adapter replacement, and subsequent actions, is given in Appendix B.

5.2 ABLATION STUDY

Table 1 separates the two routes through which a procedural memo can affect behavior by holding update opportunities constant. The deployed agent chooses when to keep or replace its memory, so update timing and count vary across episodes. In the ablation, every variant retrieves trajectories and constructs a new memo at step zero and then every five environment steps. Variants with parametric memory train a fresh adapter at those same points. The comparison therefore changes only how a newly constructed memo can influence action selection.

Variant	Memo in prompt	Memo as TTT target	Success (%) ↑
ADAM-TTT (fixed)	✓	✓	$\mathbf{65.2 \pm 0.8}$
w/o parametric memory	✓	×	58.8 ± 1.5
w/o contextual memory	×	✓	46.9 ± 3.2
w/o both	×	×	45.2 ± 1.8

Table 1: Ablation of contextual and parametric use of the procedural memo. All rows force refreshes at step zero and every five environment steps.

Necessity of contextual memory. To test whether the adapter can replace direct access to the procedural memo, we remove the memo from the action prompt while retaining it as the TTT target. Performance drops from 65.2% to 46.9%, close to the No Memory result of 45.2%. Thus, encoding the memo through a lightweight parameter update is not sufficient by itself. The result indicates that parameter adaptation alone does not substitute for the memo in the current action prompt.

Necessity of parametric memory. We next keep the procedural memo in the action prompt but disable TTT, leaving the policy parameters frozen. This variant reaches 58.8%, versus 65.2% for the full method. The 6.4-point gap shows that placing a memo in the prompt does not capture the full benefit: training an adapter from the same memo yields an additional gain in overall success.

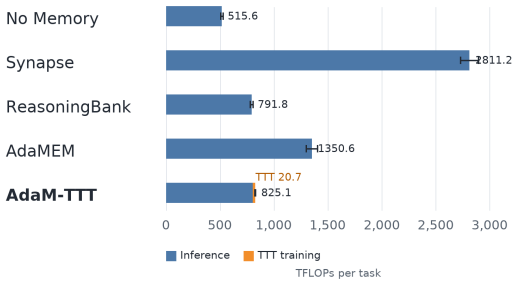
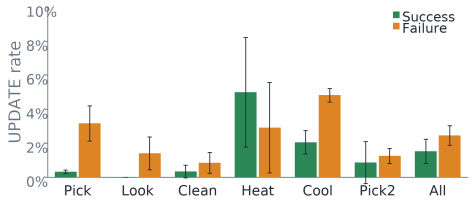
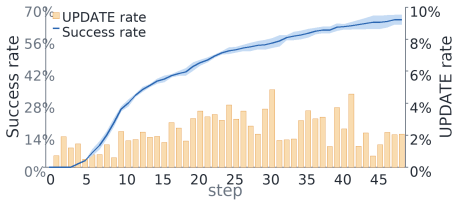


Figure 4: Estimated parameter compute per task. Bars separate inference and TTT training; error bars show sample standard deviations over three repetitions.



(a) UPDATE frequency by task category and terminal outcome. Bars show mean ± sample standard deviation across three repetitions; "All" aggregates categories.



(b) Per-step UPDATE rate (bars, right axis) and cumulative success rate (line, left axis). The band shows mean ± sample standard deviation; the denominator is all 140 environments.

Figure 5: Observed discretionary adaptation behavior across three ALFWorld repetitions. Mandatory step-zero initialization is excluded from UPDATE statistics.

Complementarity of the two pathways. Removing both pathways recovers the No Memory agent at 45.2%. Parametric memory alone yields 46.9%, contextual memory alone yields 58.8%, and combining them reaches 65.2%. Together, the variants show that the full method needs both uses of the memo: direct contextual guidance and the accompanying parameter update. The takeaway from Table 1 is that the two memory pathways are complementary: removing either one substantially reduces success, while removing both returns to the no-memory baseline.

5.3 Test-time compute

Online adaptation is useful only if its extra cost is commensurate with its benefit. We therefore estimate model compute from saved trajectories rather than compare wall-clock time across different serving deployments. We approximate token counts from logged text using four characters per token and use the standard parameter-FLOP accounting of $2P$ per inference token and $6P$ per causal-language-modeling training token for a $P = 4 \times 10^9$ parameter model (Kaplan et al., 2020). We average per-task estimates within each repetition and report the mean and sample standard deviation over three repetitions. These are parameter-FLOP estimates, not measurements of latency or energy use, and they omit attention's prompt-length-dependent cost.

As shown in Figure 4, **AdaM-TTT** adds 20.71 TFLOPs of memo training per task, only 2.51% of its estimated total. The estimate is therefore dominated by inference rather than TTT training. The takeaway is that the added adaptation cost is small relative to the inference cost.

5.4 Adaptation behavior

Why should a policy replace its temporary memory rather than retain it? Figure 5 summarizes the logged decisions, which we examine to characterize when UPDATE is used without treating these observations as causal evidence. The analysis covers all episodes in three complete 140-task

repetitions. Each episode has a mandatory initialization at step zero, so the statistics below describe only later discretionary decisions.

Across the three repetitions, the policy made 11,317 post-initialization decision calls. It replaced the temporary state with a newly retrieved memo and adapter 238 times (2.1%) and reset it 130 times (1.1%); the remaining calls retained the current state. The first post-initialization replacement had median step 17, and successive replacement events were separated by a median of two steps. Thus, the policy generally reuses its current state and adapts selectively rather than retraining at every environment step.

Replacement events were concentrated in unsuccessful episodes: unsuccessful episodes averaged 293 replacements across 146 episodes (2.01 per episode), whereas successful episodes averaged 75 across 274 episodes (0.27 per episode). This association is descriptive rather than causal: difficult episodes can both trigger repeated adaptation and be more likely to fail. It nevertheless shows that the decision policy responds more often when the current state is not solving the task. The takeaway from Figure 5 is that AdaM-TTT usually reuses its temporary state and invokes replacement selectively when progress is difficult.

6 Discussion and limitations

AdaM-TTT suggests a practical way to make deployed agents improve from experience without changing the base model. External memory can accumulate successful behavior across episodes, contextual memory can make that experience immediately actionable, and a temporary adapter can carry its effect across subsequent decisions. This separation could support agents that personalize to a user, workspace, or task family after deployment while keeping the general capabilities of the base model intact. Because adapters are replaced rather than merged indefinitely, different environments or users could maintain isolated adaptation states, reducing interference between concurrent tasks. The same design also offers a systems-level benefit: retrieval, inference, and adapter training can be served independently, so adaptation need not require a monolithic retraining pipeline.

These implications remain conditional on several limitations. Retrieval and summarization quality determine whether past success becomes useful guidance; a successful trajectory can still produce a memo that does not fit the current state, and a progress summary can omit information needed for training. Our evaluation uses one embodied-text benchmark and one backbone, so transfer to other environments, models, and memory banks remains open. The compute analysis estimates parameter FLOPs rather than end-to-end latency, memory footprint, or energy use, and the adaptation analysis is observational rather than causal. Future work should evaluate broader settings and learn cost-aware keep, update, and reset decisions that explicitly trade adaptation benefit against serving cost.

7 Conclusion

We introduced **AdaM-TTT**, which connects three locations where an agent can retain experience. Successful trajectories remain in external memory; a state-conditioned procedural memo makes the relevant procedure available in context; and a lightweight LoRA adapter temporarily encodes that memo in the policy. Training a fresh adapter from the same memo lets contextual guidance affect both the current action and later decisions, Because each update trains a fresh adapter from the frozen base model, updates do not accumulate in a continuing adapter. On ALFWorld, this design improves on the strongest evaluated memory baseline, and controlled ablations show that neither contextual nor parametric use of the memo alone achieves the full gain. A key open question is how to decide when an update is worth its added cost, particularly in episodes that trigger repeated updates.

References

Emre Can Acikgoz, Cheng Qian, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. TT-SI: self-improving LLM agents with test-time training. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 9483–9508. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.FINDINGS-ACL.462. URL https://doi.org/10.18653/v1/2026.findings-acl.462.

Rachit Bansal, Aston Zhang, Rishabh Tiwari, Lovish Madaan, Sai Surya Duvvuri, Devvrit Khatri, David Brandfonbrener, David Alvarez-Melis, Prajjwal Bhargava, Mihir Sanjay Kale, and Samy Jelassi. Let's (not) just put things in Context: Test-Time Training for Long-Context LLMs, 2025. URL https://arxiv.org/abs/2512.13898.

Yishuo Cai, Xingyu Guo, Xuancheng Huang, Jinhua Du, Can Huang, Wenxuan Huang, Wenhan Ma, Yuyang Hu, Aohan Zeng, Jie Tang, and Xu Sun. From Player to Master: Enhancing Test-Time Learning of LLM Agents via Reinforcement Learning over Memory, 2026. URL https://arxiv.org/abs/2606.08656.

Arthur Chen, Zuxin Liu, Jianguo Zhang, Akshara Prabhakar, Zhiwei Liu, Shelby Heinecke, Silvio Savarese, Victor Zhong, and Caiming Xiong. Test-Time Adaptation for LLM Agents via Environment Interaction. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=OH4PE0TDo0.

Kenneth Ge. Agents unlock new capabilities through Switching LoRA Adapters as a Tool (SLAaaT), 2026. URL https://arxiv.org/abs/2608.17034.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=CNL2bku4ra.

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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jonas Hübotter, Sascha Bongni, Ido Hakimi, and Andreas Krause. Efficiently learning at test-time: Active fine-tuning of llms. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=NS1G1Uhny3.

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, 2026. URL https://arxiv.org/abs/2601.20802.

Dongming Jiang, Yi Li, Songtao Wei, Jinxin Yang, Ayushi Kishore, Alysa Zhao, Dingyi Kang, Xu Hu, Feng Chen, Qiannan Li, and Bingzhe Li. Anatomy of Agentic Memory: Taxonomy and Empirical Analysis of Evaluation and System Limitations, 2026. URL https://arxiv.org/abs/2602.19320.

Gurusha Juneja, Arnav Kumar Jain, Deepak Nathani, William Yang Wang, and Xin Eric Wang. Learning the ARTS of Search for Automated Discovery, 2026. URL https://arxiv.org/abs/2606.21891.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In Jason Flinn, Margo I. Seltzer, Peter Druschel, Antoine Kaufmann, and Jonathan Mace (eds.), *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pp. 611–626. ACM, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Junxuan Li, Zijun Liu, Ziyi Huang, Peng Li, Yuzhou Liu, Ming Yan, and Yang Liu. Learning Simple Test-Time Environments for LLM Web Agents, 2026. URL https://arxiv.org/abs/2608.29305.

Zeyuan Liu, Jeonghye Kim, Xufang Luo, Dongsheng Li, and Yuqing Yang. Exploratory Memory-Augmented LLM Agent via Hybrid On- and Off-Policy Optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=UOzxviKVFO.

Zhanzhi Lou, Hui Chen, Yibo Li, Qian Wang, and Bryan Hooi. Learning to Learn-at-Test-Time: Language Agents with Learnable Adaptation Policies, 2026. URL https://arxiv.org/abs/2604.00830.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. CLIN: A Continually Learning Language Agent for Rapid Task Adaptation and Generalization, 2023. URL https://arxiv.org/abs/2310.10134.

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, 2025. URL https://arxiv.org/abs/2509.25140.

Tao Ren, Weiyao Luo, Hui Yang, Rongzhi Zhu, Xiang Huang, Yuchuan Wu, Bingxue Chou, Jieping Ye, Jiafeng Liang, Yongbin Li, and Yijie Peng. Scaling Self-Evolving Agents via Parametric Memory, 2026. URL https://arxiv.org/abs/2606.04536.

Pavan C Shekar and Aswanth Krishnan. Adaptive Minds: Empowering Agents with LoRA-as-Tools, 2025. URL https://arxiv.org/abs/2510.15416.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Aozhe Wang, Zhengxi Lu, Jianze Wang, Shangke Lv, Ying Liu, Weiming Lu, Jun Xiao, Yueting Zhuang, Hua Yang, Qianglong Chen, and Yongliang Shen. TTPO: Test-Time Policy Optimization, 2026a. URL https://arxiv.org/abs/2608.27448.

Yanbo Wang, Jinhua Hao, Yuze Shi, Kun Yuan, and Ming Sun. No Time Like the Present: Agentic Test-Time Training for LLM Agents, 2026b. URL https://arxiv.org/abs/2607.03441.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

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, 2025. URL https://arxiv.org/abs/2511.20857.

Zehao Xiao and Cees G. M. Snoek. Beyond Model Adaptation at Test Time: A Survey, 2024. URL https://doi.org/10.48550/arXiv.2411.03687.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Sheldon Yu, Junda Wu, Xintong Li, Nikki Lijing Kuang, Sizhe Zhou, Tong Yu, Jiawei Han, Jingbo Shang, and Julian McAuley. OLIVIA: Online Learning via Inference-time Action Adaptation for Decision Making in LLM ReAct Agents, 2026. URL https://arxiv.org/abs/2605.11169.

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, 2026. URL https://arxiv.org/abs/2601.16175.

Yunxiang Zhang, Yiheng Li, Ali Payani, and Lu Wang. AdaMEM: Test-Time Adaptive Memory for Language Agents, 2026a. URL https://arxiv.org/abs/2606.05684.

Zheyuan Zhang, Zehao Wen, Alvin Zhang, Andrew Wang, Jianwen Xie, Daniel Khashabi, and Tianmin Shu. AgentOdyssey: Open-Ended Long-Horizon Text Game Generation for Test-Time Continual Learning Agents, 2026b. URL https://arxiv.org/abs/2606.24893.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19632–19642. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a. URL https://openreview.net/forum?id=Pc8AU1aF5e.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. Llamafactory: Unified efficient fine-tuning of 100+ language models. In Yixin Cao, Yang Feng, and Deyi Xiong (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 400–410. Association for Computational Linguistics, 2024b. doi: 10.18653/V1/2024.ACL-DEMOS.38. URL https://doi.org/10.18653/v1/2024.acl-demos.38.

Xinyu Zhu, Zhe Xu, Xiaohan Wei, Yunchen Pu, Fei Tian, Chonglin Sun, Kaushik Rangadurai, Hua Zhi, Frank Shyu, Sandeep Pandey, Luke Simon, Yu Meng, and Xi Liu. Self-Guided Test-Time Training for Long-Context LLMs, 2026. URL https://arxiv.org/abs/2607.09415.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. TTRL: test-time reinforcement learning. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/be690ea16f005c174f6c4102a5970e67-Abstract-Conference.html.

Adam Zweiger, Jyothish Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/6b41e04c41726e2a60e456d0a2b961ab-Abstract-Conference.html.

AI USE DISCLOSURE

We used generative AI tools to assist with drafting and editing portions of the manuscript, organizing and summarizing relevant literature, and iterating on paper figures. We did not use generative AI tools to generate experimental data or results. All AI-assisted text, code, and visual artifacts were reviewed by the authors for accuracy, originality, and consistency with the reported experiments. The authors take full responsibility for the final content of this work, including all claims and artifacts produced with assistance from generative AI.

A PROMPT TEMPLATES

This section records the prompt interfaces used by the AdaM-TTT variant evaluated in the main comparison. Text in square brackets is inserted at runtime. The environment prompt itself is produced by the ALFWorld wrapper and contains the task, current observation, interaction history, admissible actions, and action-format instructions. Raw retrieved trajectories are never appended to the action-generation prompt.

Procedural-memo construction prompt. The summarizer receives the exact live environment prompt followed by all $K = 8$ retrieved trajectories in one call:

```
You write a compact procedural memo for an autonomous agent.  The
supplied base-agent prompt is the current task state; use it to judge
relevance.  Read all retrieved trajectories and synthesize them into
ONE consolidated memo.  Keep reusable state transitions, action
ordering, constraints, and high-level lessons.  Discard irrelevant
exploration, failed attempts, contradictions, and details that do not
apply.  Do not produce a separate memo or analysis for each trajectory.
Do not mention this instruction or the retrieval process.  Do not
provide analysis outside the memo.  Return exactly one memo enclosed in
<memo> and </memo> tags.
Current base-agent prompt:  [CURRENT LIVE ENVIRONMENT PROMPT]
[RETRIEVED TRAJECTORIES 1-8]
```

Action-generation prompt. After a memo is available, the acting policy receives the live prompt with one compact contextual suffix:

```
[CURRENT LIVE ENVIRONMENT PROMPT]
Relevant procedural memo:  [CURRENT MEMO]
```

The policy follows the environment's normal `<think>` and `<action>` format. Raw trajectories and summarizer prompts are not included in this call.

Live-progress summarization prompt. AdaM-TTT uses a second summarizer call that receives only the exact live environment prompt. This summary is used in the TTT target together with the retrieved-trajectory memo; it is not appended separately to the action prompt.

```
You write a compact progress memo for an autonomous agent.
Use only the supplied base-agent prompt.  It already contains the task,
current observation, prior actions, and prior observations.
Summarize the task progress, useful outcomes, and unresolved blockers
that matter for the next decision.  Do not reconstruct or append
history.
Keep concrete state facts and useful lessons; discard verbose or
irrelevant wording.  Do not propose a response to the user and do not
mention this instruction.
Return exactly one memo enclosed in <memo> and </memo> tags.

Base-agent prompt:
[CURRENT LIVE ENVIRONMENT PROMPT]
```

Decision prompt. The decision variant uses one policy call that proposes the environment action and chooses whether to retain or refresh memory. The current memo is included above the following template, and adaptation-decision history is inserted at the placeholder:

```
### MEMORY AND ADAPTER UPDATE OPTION
The current memo (if any) is included above.  After proposing an
action, decide whether to keep or refresh the agent's memory and
adapter state.
### MEMORY/ADAPTER DECISION HISTORY
[ADAPTATION-DECISION HISTORY]
**Implication of Each Decision:** - If you choose 'keep':  retain the
current memo.  Do not retrieve new trajectories.  Retain the current
adapter, if one is loaded.  - If you choose 'update':  retrieve
relevant trajectories and generate a new compact memo to replace the
current memo.  [TTT INSTRUCTION] **Instructions:** 1.  Generate a
proposed action based on the current live state.  2.  Choose 'keep' or
'update' for the memo and adapter state.  3.  Explain the decision
inside your <think> block.
**Output format:** <think>...reasoning about the action and
memory/adapter decision...</think> <action>...your proposed
action...</action> <update_decision>keep or update</update_decision>
```

The main results use the decision prompt described in Section 3.4; this template is included to make the evaluated interface auditable.

Training example. AdaM-TTT sends one causal-language-modeling sequence containing the live-progress summary and the consolidated retrieved memo. It never receives the raw live prompt or retrieved trajectories:

```
Live progress summary:  [LIVE-PROGRESS MEMO]
Retrieved-trajectory summary:  [PROCEDURAL MEMO]
```

The daemon receives only this serialized pair of compact summaries as its causal-language-modeling sequence; it does not receive the raw live prompt or trajectories. This is the target in the NTP objective in Section 3.3.

B TRAJECTORY-LEVEL CASE STUDY

We show a trace from an evaluation log with retrieval from successful trajectories (repetition 3, task 1). The episode succeeds in 23 environment steps; its JSON record is padded to the 50-step horizon.

Task and action trace. The goal is "put a clean soapbar in toilet." The following are the recorded environment actions; entries marked ^{U} are the steps at which the adapter was trained and replaced:

Step	0	1	2	3	4	5	6	7	8	9	10	11
Action	sink	no-op	no-op	cab.1	no-op	no-op	open 1	cab.2	open 2	cab.3	no-op	no-op
Step	12	13	14	15	16	17	18	19	20	21	22	
Action	open 3	sink 2	no-op	cab.4	open 4	counter^{U}	take soap	sink 2	clean	toilet	move	

Here "cab.1" abbreviates the logged action `go to cabinet 1`, and "counter" abbreviates `go to countertop 1`. Parser-produced no-op strings are retained as logged; each caused "Nothing happens" and are not presented as successful actions. After the second update, the agent took `soapbar 3`, cleaned it at `sinkbasin 2`, and moved it to `toilet 1`, completing the task.

Adaptation timeline.

Step	Trigger	Retrieved K	TTT target	Adapter	Next action
0	forced initialization	8	procedural memo	`...env1`	go to sinkbasin 1
1–16	keep	0	–	same	cabinet/sink exploration
17	policy update	8	procedural memo	`...env1` (replaced)	go to countertop 1
18–22	keep	0	–	same	take, clean, move

The first memo proposed the successful high-level sequence "locate, retrieve, clean, and place." At step 17, after cabinets 1–4 and both sinks had been checked, the procedural memo recorded that no soapbar had yet been found and redirected the search to the countertop. The next three relevant actions were then `take soapbar 3`, `clean soapbar 3 with sinkbasin 2`, and `move soapbar 3 to toilet 1`.

What the case does and does not show. This trace makes the information flow explicit: retrieval supplies eight successful trajectories, the summarizer produces compact state-conditioned text, the same text is used as the TTT target and as contextual memory, and the adapter is replaced only at the two recorded update points. It is a concrete success example, not a causal attribution of the final placement to a single memo sentence; the model's action decisions and environment observations remain interleaved throughout the episode.