ACTIONS SHOULD EARN THEIR CREDIT: TEST-TIME ADAPTATION AND ROUTE CONSOLIDATION FOR FROZEN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents can use feedback from repeated attempts to improve their decisions without updating model parameters. However, positive action credits can continue to favor actions that repeatedly achieve the same score, even when higher scores are achievable. Under a fixed interaction budget, trying new actions in the final attempts can also lower performance. We introduce EARN, a method that uses past experience to guide further progress and retain earlier performance through conditional reuse of effective action sequences. EARN reduces action credits for reaching scores repeatedly attained in earlier episodes and assigns larger credits to action segments that reach higher scores. These credits adjust the probabilities of LLM-proposed actions. During the final episodes, EARN directly replays stored actions when the current context matches the recorded context and the actions remain available, bypassing LLM candidate generation. We evaluate performance at the end of the interaction budget using the average score over the final four episodes. Across Jericho, ScienceWorld, and Crafter, EARN achieves the best score on this metric in 31 of 33 task–budget settings among the evaluated methods, including ties. Across the evaluated Jericho budgets, replay bypasses 78.9–86.3% of candidate-generation calls in the final four episodes, averaged across games.

1 INTRODUCTION

Completing interactive tasks, such as text-adventure games, multimodal open-world survival games, and virtual science experiments, typically requires an agent to perform a sequence of actions based on current observations. Because the right action is not always clear, the agent may try different actions, observe their outcomes, and adjust its choices. Specifically, an LLM agent uses the task description, current observations, and interaction records in its prompt to select the next action (Yao et al., 2023). If an attempt is unsuccessful, the agent can start again and try a different sequence. Importantly, the earlier attempt still provides useful information (Shinn et al., 2023). For example, a failed attempt to open a door may reveal that the agent must first obtain a key. Such information is particularly useful in later attempts, when the agent can avoid the same mistake and focus on completing the remaining task. Thus, earlier experience can help the agent make better choices rather than repeat the same failed actions.

Existing methods help agents reuse experience without updating the LLM's parameters. Specifically, Reflexion stores written reflections to guide later attempts (Shinn et al., 2023), while EvoTest revises prompts, memory, and other agent settings (He et al., 2026). JitRL more directly adjusts action probabilities using returns from earlier interactions (Li et al., 2026). However, actions that earned points in the past do not necessarily help the agent achieve higher scores. For example, the agent may find a key and open the first door to earn points in every attempt, yet fail to complete the remaining task because it misses an important action. Although these actions remain useful, continuing to prioritize them simply because they previously earned points may lead the agent to repeat the same actions and obtain the same score. Thus, using past results effectively requires distinguishing repeated progress from progress beyond what the agent has already achieved.

Therefore, two challenges arise. **(1) Moving beyond previously achieved results.** Actions that earned points in earlier attempts may continue to receive positive credit even when they produce no

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.

further progress. For example, an agent may repeatedly open the first door but never complete the remaining task. Thus, experience reuse should distinguish previously achieved progress from new gains. **(2) Retaining earlier performance.** Choosing actions step by step based on past experience does not ensure that an earlier good result will be reproduced. Even after finding an effective action sequence, the agent may choose different actions midway through a later attempt and obtain a lower score. This matters particularly when the number of attempts is limited and only a few remain. Repeating the earlier sequence can reduce this risk, but only when its actions still apply to the current situation. For example, without the key, the agent cannot repeat an action that previously unlocked the door. Therefore, reusing an effective sequence requires checking whether the current conditions still support its execution.

We introduce EARN, a training-free method that uses persistent memory to adjust action probabilities and conditionally reuse a previously effective action sequence (Figure 1). We group consecutive episodes into blocks and select the highest-scoring trajectory from each block. From this trajectory, EARN assigns numerical credits to actions leading up to a score increase, using these credits to guide later choices. Actions receive less credit when the resulting score does not exceed a level reached more than once in earlier episodes. To also use information from the other trajectories, EARN computes step-wise credits after every episode. During action selection, it combines the two credit estimates and reduces the block-wise estimate when few records match the current situation and candidate action, or when their credit values vary substantially. Separately, EARN stores a route containing an ordered action sequence and the situation recorded before each action. During the final episodes, the agent directly executes stored actions while the recorded and current situations match and the actions remain available. Otherwise, it resumes memory-guided action selection.

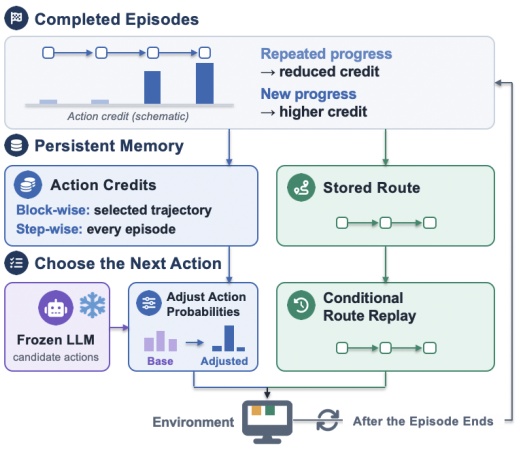


Figure 1: **EARN uses earlier scores to guide action credit assignment.**

We evaluate whether this combination of memory-guided decisions and conditional route reuse improves performance in the final episodes across Jericho (Hausknecht et al., 2020), ScienceWorld (Wang et al., 2022a), and Crafter (Hafner, 2022). Our evaluation covers three aspects: final-four average scores relative to existing baselines, retention of previously achieved scores and inference savings when routes remain executable, and adaptation when task instances change. Together, these experiments assess whether EARN can improve later decisions and reuse accumulated experience under both repeated and changing execution conditions. Our contributions are threefold:

- We introduce EARN, a training-free method that combines progress-relative block-wise credit attribution from selected trajectories with step-wise feedback from every episode to adjust candidate-action probabilities through support-controlled fusion without parameter updates.
- We develop conditional route replay to complement credit-guided action selection. It preserves an applicable route's action order and reduces LLM candidate-generation calls by directly executing stored actions. When replay stops, memory-guided action selection resumes.
- We evaluate EARN across three interactive benchmarks. Among the compared methods, EARN achieves the highest or tied-highest final-four average episode score in 14 of 15 Jericho game–budget settings, 14 of 15 ScienceWorld task–budget settings, and all three Crafter budgets.

2 Related Work

Test-time adaptation of LLM agents. Existing methods differ in what they retain from earlier episodes and how the agent uses that information in later decisions. Some methods store text derived from earlier episodes: Reflexion (Shinn et al., 2023) stores verbal reflections on task feedback, CLIN (Majumder et al., 2024) records causal relations between actions and observed effects, and ReasoningBank (Ouyang et al., 2026) distills reasoning strategies from successful and failed

trajectories. Beyond updating memory, EvoTest (He et al., 2026) revises the agent's prompt, hyperparameters, and tool-use routines. Other methods store numerical estimates for actions. REMEMBERER (Zhang et al., 2023) retrieves past actions and their estimated scores to guide action selection through the prompt. JitRL (Li et al., 2026) is the closest prior method to EARN. It uses an LLM evaluator to assign numerical feedback to actions, computes discounted returns, and averages retrieved returns for each candidate action to adjust action probabilities. EARN instead uses a score repeatedly attained in earlier episodes as a reference for assigning action credits and discounts credits for score increases up to that reference. It combines block-wise credits from one selected trajectory per block with step-wise credits from every episode to adjust action probabilities.

Reusing behavior from earlier episodes. Prior systems represent behavior from earlier interactions as trajectories, textual guidance, workflows, or executable skills. ExpeL (Zhao et al., 2024), Synapse (Zheng et al., 2024), and Self-Generated In-Context Examples (Sarukkai et al., 2025) use earlier trajectories as in-context examples from which the LLM generates new actions. AutoGuide (Fu et al., 2024) derives textual guidelines from trajectories and pairs each guideline with a description of the context in which it applies. Agent Workflow Memory (Wang et al., 2025) derives reusable workflows from trajectories and includes the workflows in prompts for later tasks. Voyager (Wang et al., 2024) stores executable code skills that can be retrieved and composed. EARN stores a route containing recorded actions and their preceding execution contexts, then directly executes those actions in order while execution checks pass.

3 Problem Formulation

Preliminaries. We consider interactive tasks in which an agent takes a sequence of actions to achieve an objective, and the environment provides observations and scores that indicate task progress. Because a single attempt may leave parts of the task unfinished, attempts allow the agent to try alternative actions and use feedback from earlier attempts. We therefore consider a session of K episodes of the same task, with a fixed objective and scoring criterion. Each episode k begins with an environment reset to initial configuration v_k, which may vary across episodes, and ends upon environment termination or after H steps, giving a length $T_k \leq H$. At step t, the agent receives execution context $s_{k,t}$, containing the current observation and supplementary information such as available actions, and executes action $a_{k,t}$. The environment then returns feedback $y_{k,t}$, including the next context, the score change, and a termination indicator. Let $S_{k,t}$ denote the running score after action $a_{k,t}$ and $S_{k,0}$ the initial score. The score change at step t is $S_{k,t} - S_{k,t-1}$.

Memory Update and Action Selection. The LLM parameters θ remain fixed throughout the session. Persistent memory $\mathcal{M}_k$ retains information from episodes $1, \ldots, k-1$, with $\mathcal{M}_1 = \varnothing$. The agent constructs prompt $x_{k,t}$ from the task specification, current context, within-episode history, and retrieved memory. An action-selection rule Π_θ uses both the prompt and persistent memory:

$$a_{k,t} \sim \Pi_\theta(\cdot \mid x_{k,t}, \mathcal{M}_k). \tag{1}$$

The trajectory τ_k records the ordered sequence of contexts, actions, and feedback from episode k:

$$\tau_k = \big(s_{k,t}, a_{k,t}, y_{k,t}\big)_{t=1}^{T_k}. \tag{2}$$

The task's scoring criterion assigns episode score S_k to the completed trajectory. An action credit is instead a numerical value constructed from interaction feedback to guide action selection in later episodes. After each episode, a memory-update rule U produces the memory for the next episode:

$$\mathcal{M}_{k+1} = U(\mathcal{M}_k, \tau_k, S_k). \tag{3}$$

Goal. We seek a memory-update rule U and an action-selection rule Π_θ that maximize the expected average score over the final W episodes, where $1 \leq W \leq K$, subject to the episode budget K and per-episode step limit H:

$$J_{\text{term}} = \mathbb{E}_{\text{agent, environment}}\left[\frac{1}{W}\sum_{k=K-W+1}^{K} S_k\right]. \tag{4}$$

4 METHOD

EARN uses scores repeatedly attained in earlier episodes as a reference for assigning action credits, discounting credits for reaching those scores again. The memory-update rule U adds step-wise credits after every episode and block-wise credits from score-increase events after selecting one trajectory from each block. The action-selection rule Π_θ retrieves credits to adjust candidate-action probabilities and conditionally replays a stored route. Figure 2 illustrates both credit assignments on one trajectory. Algorithm 1(Appendix 9.4) summarizes our methods.

4.1 LEARNING: ACTION CREDIT AND PERSISTENT MEMORY

Block-wise Credit Attribution. To prioritize experience with greater task progress, we compare trajectories within non-overlapping blocks of B consecutive acquisition episodes and select one per block, primarily by the highest running score reached. This provides several outcomes to compare while allowing later experience to contribute without surpassing the best score of the entire session. To give less credit for progress already achieved, we maintain a progress frontier F_b using route scores observed multiple times in recent episodes. The frontier starts at zero, never decreases, and is fixed at the start of each block. A score-increase event e occurs when $S_{k,t_e} > S_{k,t_e-1}$; let $S_e = S_{k,t_e}$. Because earlier actions may contribute to this increase, we credit a segment rather than only the scoring action. For $S_e \le F_b$, we use a local segment ending at t_e. Otherwise, we use the segment after first reaching or exceeding F_b through t_e, covering the actions leading to progress beyond the frontier. Appendix 9.2 gives the selection, frontier-update, and segment details.

For a segment of L_e actions, let $c_{e,i}$ denote the credit assigned to the action at position i. We assign larger credits to actions closer to the score-increase event and discount credits for repeated progress. The interpolation endpoints ℓ_e and h_e depend on the discount coefficient α_{pre} for repeated progress and the credit offset ρ for progress beyond F_b:

$$c_{e,i} = \left(1 - \frac{i}{L_e}\right)\ell_e + \frac{i}{L_e}h_e, \qquad (\ell_e, h_e) = \begin{cases} (0, \alpha_{\text{pre}}), & S_e \le F_b, \\ (\rho, 1), & S_e > F_b, \end{cases} \quad i = 1, \ldots, L_e. \tag{5}$$

Credit increases linearly toward the scoring action. For repeated progress, it receives α_{pre}; for new progress, it receives 1. The offset ρ gives preceding actions positive credit in new-progress segments.

Step-wise Credit Attribution. To provide timely feedback between block-wise updates and retain information from unselected trajectories, we compute step-wise credits after every episode, including replay episodes. For each step t, we construct a numerical signal $\tilde{r}_{k,t}$ from environment feedback and interaction history. A score-increase event contributes 1 if the running score after the event exceeds the progress frontier, and a discounted positive signal otherwise. Here, we use the frontier before updating memory from the current episode. We additionally give small bonuses for previously unobserved outcomes and apply penalties for detected terminal failures. Using temporal discount factor γ, we accumulate these signals backward through the episode to compute step-wise credit $C^{\text{step}}_{k,t}$ for action $a_{k,t}$:

$$C^{\text{step}}_{k,t} = \sum_{u=t}^{T_k} \gamma^{u-t} \tilde{r}_{k,u}. \tag{6}$$

Here, γ controls the decay of event feedback as it is propagated backward to preceding actions. An action can receive credit from subsequent events even if it produces no immediate score increase. The resulting credits are stored after each episode, without additional LLM evaluation calls.

Persistent Memory. Persistent memory $\mathcal{M}_k$ stores a route $\mathcal{R}^\star_k$ and action-credit records $\mathcal{E}_k$. A route is an ordered action sequence extracted from an episode trajectory, together with the execution context recorded before each action. Each credit record contains an execution context, an action, an assigned credit, and the credit source. Step-wise records are added after every episode; block-wise records from score-increase events are added after selecting a trajectory at block completion. Route storage is independent of block-wise selection: after each episode, the agent considers the new trajectory for route retention based on performance and executability. Details are in Appendix 9.

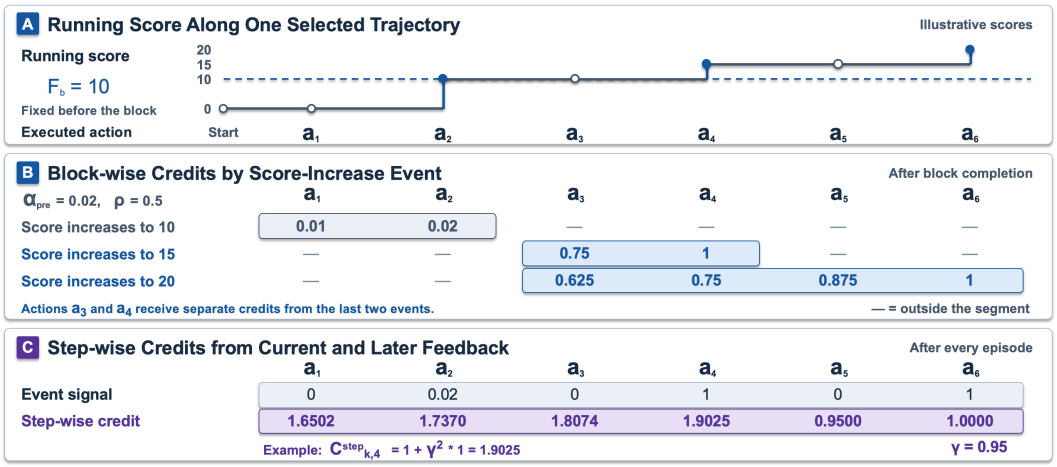


Figure 2: **An illustrative example of EARN's** block-wise and step-wise credit attribution. Environment scores are illustrative. Both attribution rules use a frontier of 10 in this example; the step-wise calculation includes only score-increase signals.

4.2 Adaptation: Adjusting Candidate-Action Probabilities

Given execution context $s = s_{k,t}$, the agent incorporates retrieved memory into prompt $x_{k,t}$. The frozen LLM then proposes candidate actions and reports confidence scores, which we normalize into a base distribution $p^0(a \mid s)$ over the candidate set $\mathcal{A}(s)$. Additional memory candidates, when enabled, are incorporated as described in Appendix 9.5.

Credit Estimates. EARN retrieves block-wise and step-wise credit records from persistent memory $\mathcal{M}_k$. Semantic retrieval can supplement insufficient exact matches; Appendix 9.5 describes this procedure. For block-wise records matching context s and candidate action a exactly, let n_{blk}, μ_{blk}, and σ^2_{blk} denote their count, mean, and sample variance. These statistics depend on (s, a); we omit the arguments for brevity. To moderate estimates based on limited or inconsistent credit records, we shrink the block-wise estimate toward zero using count-based scaling and a variability correction:

$$r_{\text{blk}}(s,a) = \frac{n_{\text{blk}}}{n_{\text{blk}} + \nu}, \qquad Q_{\text{blk}}(s,a) = r_{\text{blk}}(s,a)\,\text{sgn}(\mu_{\text{blk}}) \left[|\mu_{\text{blk}}| - \xi \sqrt{\frac{\sigma^2_{\text{blk}}}{n_{\text{blk}}}} \right]_+ . \tag{7}$$

Here, $[x]_+ = \max(0, x)$, ν controls count-based scaling, and ξ controls the variability correction. Without matching records, $r_{\text{blk}} = Q_{\text{blk}} = 0$. Each retrieved step-wise credit is scaled according to its episode's replay status and whether the running score before or after the action exceeded that episode's initial progress frontier. The estimate $Q_{\text{step}}(s, a)$ is the mean scaled credit over records matching action a. We center each estimate against its channel's baseline: $A_{\text{blk}} = Q_{\text{blk}} - V_{\text{blk}}$ and $A_{\text{step}} = Q_{\text{step}} - V_{\text{step}}$. Here, $V_{\text{blk}}(s)$ is the mean block-wise estimate over candidate actions weighted by r_{blk}, and $V_{\text{step}}(s)$ is the mean scaled credit over all retrieved step-wise records. These centered estimates express each candidate's value relative to its channel's baseline.

Combining Credit Estimates. Because block-wise selection retains only one trajectory per block, step-wise credits use feedback from every episode, including those not selected. When matching block-wise records are sparse, we retain a larger coefficient on the step-wise estimate. As block-wise support grows, we reduce this coefficient to place greater emphasis on credits from selected trajectories. We therefore combine the centered estimates as:

$$A_{\mathcal{M}}(s,a) = g_{\text{blk}} A_{\text{blk}}(s,a) + \lambda_{\text{step}} [1 - \eta g_{\text{blk}}]_+ \, g_{\text{step}} q_{\text{step}} A_{\text{step}}(s,a). \tag{8}$$

The gates $g_{\text{blk}}, g_{\text{step}} \in [0, 1]$ regulate each channel's contribution according to retrieved support. The coefficient λ_{step} sets the step-wise scale, η controls its suppression as g_{blk} increases, and q_{step} is the largest record-scaling factor among matching step-wise records. The gates and q_{step} depend on (s, a); their arguments are omitted. Step-wise feedback supplements sparse block-wise evidence. As block-wise support grows, the suppression factor reduces the step-wise contribution to the combined estimate before standardization. Appendix 9.4 specifies the record-scaling factors.

Probability Adjustment. We standardize $A_{\mathcal{M}}(s,a)$ over the candidate set using its mean and population standard deviation, then clip the result to $[-\zeta,\zeta]$, obtaining $\widehat{A}_{\mathcal{M}}(s,a)$. With adjustment strength $\beta > 0$, the adjusted distribution is

$$p_{\mathcal{M}}(a \mid s) = \frac{p^0(a \mid s)\exp\Big(\beta \widehat{A}_{\mathcal{M}}(s,a)\Big)}{\sum_{a' \in \mathcal{A}(s)} p^0(a' \mid s)\exp\Big(\beta \widehat{A}_{\mathcal{M}}(s,a')\Big)}. \tag{9}$$

This adjustment shifts the base probability ratio in favor of candidates with higher standardized credit estimates, while retaining the LLM's base probabilities as a factor. The parameter $\zeta > 0$ bounds the standardized estimates. Appendix 9.5 describes the numerical handling.

4.3 Conditional Route Replay

Block-wise and step-wise credits adjust candidate-action probabilities, but do not ensure that the agent follows the action order in a stored route $\mathcal{R}_k^\star$. The agent therefore maintains a replayable route separately and conditionally follows its action order when execution checks pass.

Let K_c denote the number of final episodes scheduled for route replay, with $z_k = \mathbb{I}\{k > K - K_c\}$ indicating replay eligibility. The evaluation window W independently specifies which episode scores enter J_{term}. Let m index the next action $a_m^\star$ in $\mathcal{R}_k^\star$. The indicator $\chi_{k,t} \in \{0,1\}$ equals one when replay remains active in the episode, a route step remains, the current execution context matches its recorded context, and the action passes availability and terminal-failure checks. The operator Select follows the configured action-selection rule. The agent executes

$$a_{k,t} = \begin{cases} a_m^\star, & z_k \chi_{k,t} = 1, \\ \text{Select}(p_{\mathcal{M}}(\cdot \mid s_{k,t})), & \text{otherwise.} \end{cases} \tag{10}$$

Each replayed action advances the route pointer and bypasses LLM candidate generation, while still counting as an environment interaction. A failed execution check or an exhausted route ends replay for the current episode. The agent then resumes candidate generation and uses the block-wise and step-wise credit estimates to adjust action probabilities.

5 Experiments

We organize our experiments around three questions on memory-guided test-time adaptation and final-stage consolidation in LLM agents: **Q1: Performance in the final episodes.** Does EARN improve performance in the final episodes over the compared baselines across interactive environments? **Q2: Performance Retention and Inference Savings.** When stored routes are broadly applicable, how reliably does EARN reproduce its best acquired performance, and how many LLM candidate-generation calls does conditional consolidation save? **Q3: Experience Transfer.** When stored routes are inapplicable, how does EARN use accumulated experience to adapt its decisions and maintain performance in the final episodes?

5.1 Experimental Setup

Environments and Tasks. Experiments span three benchmarks with complementary demand profiles. **Jericho** (Hausknecht et al., 2020) offers parser-based interactive fiction; we adopt the Focus-5 subset (DeepHome, Detective, Zork I, Zork III, ZTUU) and run lifelong sessions of 30×30, 50×50, and 70×70 (episodes × steps per episode). **ScienceWorld** (Wang et al., 2022a) is evaluated on five tasks, with at most 100 steps per episode and session lengths of 10, 20, and 30 episodes. Task variations alter objects, materials, or environmental configurations across episodes, requiring stored experience to be checked for applicability to the current instance. **Crafter** (Hafner, 2022) is an open-ended survival environment evaluated in a multimodal setting using visual observations and textual context. Performance is measured by the number of distinct achievements unlocked (0–22), with at most 100 steps per episode and session lengths of 10, 20, and 30 episodes.

Compared Approaches. All approaches share an identical frozen W4A16-quantized backbone Gemma-4-31B-Instruct (Team et al., 2026), served locally via vLLM (Kwon et al., 2023). Baselines

Table 1: Jericho results under different episode/step budgets. Last-4 denotes the average score over the final four episodes, and Last denotes the final episode's score (↑). Bold values indicate the best results within each budget.

Episodes / Steps	Method	DeepHome		Detective		Zork I		Zork III		ZTUU	
		Last-4	Last	Last-4	Last	Last-4	Last	Last-4	Last	Last-4	Last
30/30	Memory	13.8	14.0	70.0	80.0	5.0	5.0	1.8	2.0	3.8	5.0
	Reflexion	6.8	7.0	20.0	20.0	2.5	0.0	0.8	1.0	0.0	0.0
	AWM	14.0	14.0	50.0	50.0	10.0	10.0	0.0	0.0	0.0	0.0
	EvoTest	14.8	15.0	82.5	90.0	40.0	40.0	2.3	2.0	2.5	5.0
	JitRL	19.0	18.0	172.5	**190.0**	42.3	**50.0**	2.5	2.0	6.3	10.0
	Ours	**19.3**	**20.0**	**190.0**	**190.0**	**47.5**	**50.0**	**4.0**	**4.0**	**20.8**	**22.0**
50/50	Memory	13.5	14.0	90.0	110.0	41.3	45.0	3.0	3.0	7.5	10.0
	Reflexion	5.0	1.0	17.5	20.0	2.5	0.0	0.5	0.0	0.0	0.0
	AWM	15.0	15.0	72.5	80.0	10.0	10.0	1.0	1.0	0.0	0.0
	EvoTest	15.0	12.0	310.0	310.0	41.0	40.0	2.5	3.0	8.8	10.0
	JitRL	14.8	19.0	310.0	310.0	47.5	50.0	3.0	3.0	0.0	0.0
	Ours	**23.0**	**23.0**	**340.0**	**340.0**	**54.0**	**54.0**	**5.0**	**5.0**	**27.0**	**27.0**
70/70	Memory	16.3	13.0	252.5	300.0	5.0	5.0	2.8	3.0	**20.0**	20.0
	Reflexion	4.8	1.0	17.5	20.0	5.0	0.0	0.5	1.0	0.0	0.0
	AWM	27.0	27.0	50.0	50.0	0.0	0.0	3.0	3.0	5.0	5.0
	EvoTest	51.5	**64.0**	315.0	310.0	42.0	40.0	2.8	3.0	10.0	10.0
	JitRL	20.0	20.0	300.0	300.0	41.0	55.0	2.8	4.0	0.0	0.0
	Ours	**64.0**	**64.0**	**320.0**	**320.0**	**59.0**	**59.0**	**5.0**	**5.0**	19.5	**27.0**

retain their native decoding, memory, retrieval settings. EARN uses an action-generation temperature of 0.8 across all three benchmarks. EARN uses all-MiniLM-L6-v2 (Wang et al., 2020) embeddings for semantic retrieval in Jericho and ScienceWorld, exact-context retrieval for action evidence in Crafter. We compare against five baselines that reuse experience while keeping the LLM weights fixed: (1) **Reflexion** (Shinn et al., 2023), which appends verbal self-critiques of past failures to the prompt; (2) **AWM** (Wang et al., 2025), which distills successful episodes into persisted reusable workflows; (3) **FIFO Memory**, which prepends full episode transcripts under a first-in-first-out context budget; (4) **EvoTest** (He et al., 2026), which evolves prompts, memory entries, and decoding hyperparameters across episodes; and (5) **JitRL** (Li et al., 2026), which converts returns of past trajectories into advantages that rescale candidate-action probabilities.

Protocol and Metrics. All methods are evaluated under the same episode and step budgets. We report **Last-4**, $S_{\text{Last-4}} = \frac{1}{4}\sum_{k=K-3}^{K} S_k$, and **Last**, $S_{\text{Last}} = S_K$. For Jericho, we additionally report **Retention**, $R_{\text{ret}} = \frac{1}{4}\sum_{k=K-3}^{K} \mathbf{1}[S_k = S^\star \wedge S^\star > 0]$, where $S^\star = \max_{1\le j\le K} S_j$ is the run's peak score. Retention measures the fraction of final-window episodes matching a positive peak and is zero for runs without one. EARN enables conditional route replay during the final-four consolidation window. More implementation details are provided in Appendices 10.

5.2 Main Results

Performance in the final episodes (Q1). EARN achieves the highest or tied Last-4 score in 14 of 15 Jericho game–budget settings (Table 1) and the best score in 14 of 15 ScienceWorld settings (Table 2), including ties. Crafter results evaluate its performance in multimodal interaction (Table 3). Across these settings, the results support the value of selectively reusing accumulated experience for terminal performance, both when stored routes remain applicable and when decisions require adaptation to the current context. Larger interaction budgets do not uniformly yield higher scores, highlighting that how experience is used matters alongside how much is collected. Interaction creates opportunities for progress, but does not by itself ensure that experience will be retained and effectively reused. The comparison highlights the importance of experience selection and context-aware reuse, alongside exploration, for achieving strong performance near the budget endpoint.

Performance Retention and Inference Savings (Q2). When stored routes remain applicable, EARN combines reliable performance reproduction with fewer candidate-generation calls. At 50×50 on Jericho, Last-4 retention reaches 1.00 on every game, compared with five-game averages of 0.55 for EvoTest and 0.05 for JitRL (Figure 3). EARN therefore matches its run-best scores throughout the final four episodes, rather than recovering only in the last episode. On Jericho, replay avoids 78.90–86.27% of candidate-generation calls in the final four episodes. These

Table 2: ScienceWorld results across task variations, i.e., instances of the same task with different objects, materials, or environmental configurations. Last-4 denotes the mean solving score over the final four episodes, and Last denotes the final episode's score (↑). Each episode allows 100 steps.

Episodes	Method	Conductivity		Unknown Conductivity		Plant		Power		Melting Known	
		Last-4	Last	Last-4	Last	Last-4	Last	Last-4	Last	Last-4	Last
10	Memory	19.0	19.0	13.3	10.0	11.3	13.0	**58.0**	**58.0**	**100.0**	**100.0**
	AWM	19.0	19.0	14.5	19.0	11.0	13.0	20.0	20.0	93.3	**100.0**
	Reflexion	10.0	10.0	10.0	10.0	3.3	1.0	10.0	10.0	86.5	73.0
	EvoTest	27.8	29.0	24.3	29.0	11.8	13.0	**58.0**	**58.0**	93.3	**100.0**
	JitRL	47.5	60.0	60.0	60.0	**34.8**	**37.0**	22.8	0.0	84.8	**100.0**
	Ours	**87.3**	**83.0**	**82.0**	**83.0**	**34.8**	**37.0**	**58.0**	**58.0**	**100.0**	**100.0**
20	Memory	17.8	33.0	16.5	14.0	10.8	11.0	**58.0**	**58.0**	**100.0**	**100.0**
	AWM	19.0	19.0	12.3	29.0	10.3	11.0	25.0	25.0	71.3	**100.0**
	Reflexion	8.8	10.0	8.8	10.0	3.5	5.0	10.0	10.0	93.3	73.0
	EvoTest	31.0	29.0	22.8	29.0	13.0	13.0	**58.0**	**58.0**	**100.0**	**100.0**
	JitRL	60.0	60.0	60.0	60.0	**33.8**	**33.0**	24.0	5.0	76.5	**100.0**
	Ours	**74.0**	**69.0**	**77.5**	**79.0**	**33.8**	**33.0**	55.5	53.0	**100.0**	**100.0**
30	Memory	18.0	10.0	18.0	24.0	10.0	8.0	56.8	**58.0**	**100.0**	**100.0**
	AWM	10.0	10.0	19.3	24.0	10.0	8.0	22.5	20.0	**100.0**	**100.0**
	Reflexion	10.0	10.0	8.8	10.0	1.8	2.0	8.8	10.0	75.0	**100.0**
	EvoTest	29.0	29.0	19.3	19.0	11.0	8.0	**58.0**	**58.0**	**100.0**	**100.0**
	JitRL	60.0	60.0	60.0	60.0	**33.5**	**32.0**	36.0	53.0	76.5	**100.0**
	Ours	**75.0**	**83.0**	**60.5**	**74.0**	**33.5**	**32.0**	**58.0**	**58.0**	**100.0**	**100.0**

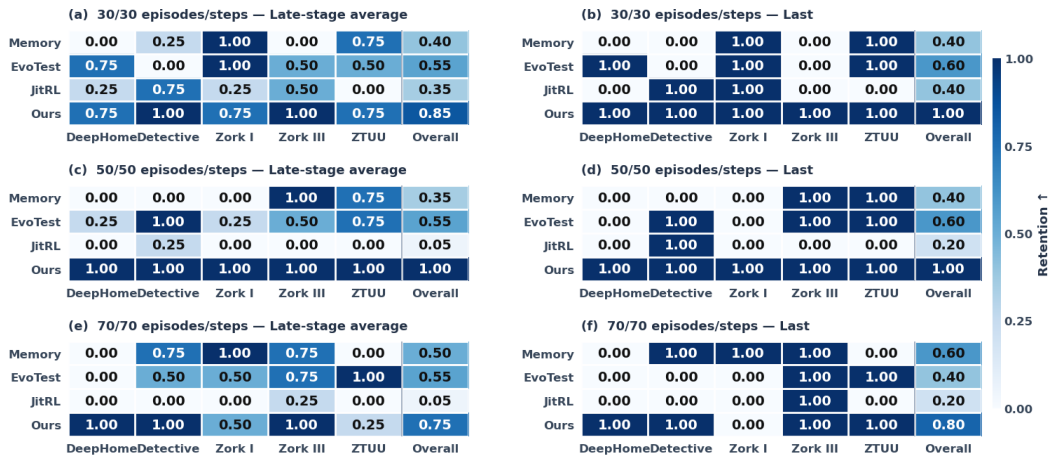


Figure 3: Late-stage memory retention on Jericho (↑). Darker indicates higher retention.

Table 3: Crafter achievement counts (0–22), with 100 steps per episode. Last-4 denotes the final-four average; Last denotes the final episode's score (↑).

Method	10 Episodes		20 Episodes		30 Episodes	
	Last-4	Last	Last-4	Last	Last-4	Last
Memory	7.0	7.0	8.3	**9.0**	4.0	5.0
AWM	4.8	2.0	2.8	6.0	5.3	**8.0**
Reflexion	5.0	4.0	5.0	6.0	5.5	4.0
EvoTest	6.5	7.0	6.0	7.0	6.0	6.0
JitRL	5.8	6.0	6.8	7.0	6.0	7.0
Ours	**7.8**	**9.0**	**9.0**	**9.0**	**8.5**	**8.0**

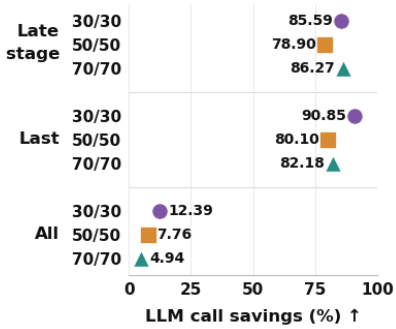


Figure 4: Jericho candidate-generation call savings (%, ↑)

final-window replay steps correspond to whole-run call savings of 4.94–12.39%, averaged equally across games (Figure 4). Retention measures reproduction of each run's own peak, not global optimality. The saved calls could support additional exploration under a fixed inference budget, although the performance benefit of such reallocation remains untested.

Table 4: Ablation results over 30 episodes and 30 steps each. Last-4 denotes the final-four-episode mean, and Last denotes the final-episode score (↑). Bold indicates the highest score in each column.

Method	ZTUU		Zork III	
	Last-4	Last	Last-4	Last
Memory	3.8	5.0	1.8	2.0
Reflexion	0.0	0.0	0.8	1.0
AWM	0.0	0.0	0.0	0.0
EvoTest	2.5	5.0	2.3	2.0
JitRL	6.3	10.0	2.5	2.0
EARN (Full)	**20.8**	**22.0**	4.0	4.0
w/o Block-wise Credit	12.0	12.0	3.0	3.0
w/o Step-wise Credit	16.5	17.0	**5.0**	**5.0**
w/o Replay	12.0	12.0	3.0	3.0

Table 5: Sensitivity to step-wise credit suppression on ZTUU and Zork III over 30 episodes with 30 steps each. The base step-wise credit weight is fixed at 0.35, while its effective contribution is attenuated by the suppression coefficient according to the strength of block-wise evidence. EARN (Full) uses the original suppression coefficient of 0.85. Min denotes the minimum scores over the final four episodes (↑).

Method	Supp.	ZTUU			Zork III		
		Last-4	Last	Min	Last-4	Last	Min
Full	0.85	**20.8**	**22.0**	**17.0**	4.0	4.0	**4.0**
EARN	0.0	16.0	16.0	16.0	3.0	3.0	3.0
	0.5	12.0	12.0	12.0	3.0	3.0	3.0
	1.0	16.8	18.0	13.0	**4.5**	**5.0**	**4.0**

Experience Transfer (Q3). In ScienceWorld, task variations change the environment across episodes, so stored routes may become inapplicable. EARN can still use retrieved experience and action-level evidence to adjust candidate preferences, while generating actions conditioned on the current observation. This separates experience reuse from fixed-route execution: changes in object locations or task requirements need not invalidate all previously acquired evidence. EARN achieves the best or tied Last-4 score in 14 of 15 task–budget settings (Table 2). EARN reuses experience without requiring an entire route to remain applicable: retrieved memory informs candidate generation, while supported action-level evidence adjusts preferences under the current observation. This enables selective reuse of prior experience even when environmental changes prevent direct replay.

5.3 Ablation Study and Method Analysis

Ablation Studies. We remove block-wise credit, step-wise credit, and conditional replay individually on ZTUU and Zork III to assess their contributions to final-episode performance and examine whether the gains come solely from replay. All settings use 30 episodes and 30 steps each. As shown in Table 4, removing either block-wise credit or conditional replay reduces Last-4 from 20.8 to 12.0 on ZTUU and from 4.0 to 3.0 on Zork III. The decline without block-wise credit, even with replay retained, supports contributions from both credit-based action adjustment and route reuse. The effect of step-wise credit varies across tasks: removing it hurts ZTUU, supporting the use of feedback from all trajectories to complement block-wise experience, but improves both Last-4 and Last on Zork III. Additional step-wise feedback therefore does not necessarily improve performance, motivating further analysis of its contribution strength.

Sensitivity Analysis. We vary the suppression coefficient $\eta \in \{0, 0.5, 0.85, 1.0\}$ in Equation 8, keeping $\lambda_{\text{step}} = 0.35$ and the remaining settings fixed. As shown in Table 5, the default $\eta = 0.85$ achieves the highest Last-4 score on ZTUU (20.8). On Zork III, $\eta = 1.0$ achieves the highest Last-4 score (4.5), compared with 4.0 for the default; both attain a final-four minimum score of 4.0. On both tasks, $\eta = 1.0$ outperforms $\eta = 0$ in Last-4, although performance does not vary monotonically with suppression strength. Thus, retaining a stronger step-wise contribution does not necessarily improve performance. The default performs well on both tested tasks, ranking first on ZTUU and remaining close to the best tested setting on Zork III.

6 Conclusion

We introduced EARN, a training-free method that combines progress-relative block-wise credits with step-wise feedback to adjust candidate-action probabilities. Block-wise selection prioritizes high-performing trajectories, while step-wise updates retain feedback from every episode. Conditional route replay reuses applicable action sequences during the final episodes, bypassing LLM candidate generation and returning to memory-guided selection when replay ends. Across Jericho, ScienceWorld, and Crafter, EARN achieves the best or tied-best Last-4 score in 31 of 33 task–budget settings, with substantial candidate-generation call savings on Jericho. Future work will examine experience reuse under changing conditions and stochastic transitions.

AI USE STATEMENT

We used generative AI tools to polish manuscript writing and assist with literature searches and the discovery of related work. We did not use these tools for research ideation, synthetic data generation, or mathematical proofs. All coding and experiments were carried out independently by the authors. We reviewed all AI-assisted text and checked the cited sources. We take responsibility for the final content of this work, including its text, claims, and AI-assisted content.

REFERENCES

Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval, 2019. URL https://arxiv.org/abs/1908.04742.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/16105fb9cc614fc29e1bda00dab60d41-Paper.pdf.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline, 2020. URL https://arxiv.org/abs/2004.07211.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning, 2019. URL https://arxiv.org/abs/1902.10486.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. In *Advances in Neural Information Processing Systems*, volume 37, pp. 119919–119948. Curran Associates, Inc., 2024. doi: 10.52202/079017-3811.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=1W0z96MFEoH.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7903–7910, 2020.

Yufei He, Juncheng Liu, Yue Liu, Yibo Li, Tri Cao, Zhiyuan Hu, Xinxing Xu, and Bryan Hooi. Evotest: Evolutionary test-time learning for self-improving agentic systems. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=JFnnajbkvP.

Hankyul Kang, Gregor Seifer, Donghyun Lee, and Jongbin Ryu. Do your best and get enough rest for continual learning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10077–10086. IEEE, 2025.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Yibo Li, Zijie Lin, Ailin Deng, Xuan Zhang, Yufei He, Shuo Ji, Tri Cao, and Bryan Hooi. Just-in-time reinforcement learning: Continual learning in LLM agents without gradient updates. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://openreview.net/forum?id=pLvye0zHUC.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning, 2022. URL https://arxiv.org/abs/1706.08840.

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. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=xS6zx1aBI9.

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. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf.

Vishnu Sarukkai, Zhiqiang Xie, and Kayvon Fatahalian. Self-generated in-context examples improve LLM agents for sequential decision-making tasks. In *Advances in Neural Information Processing Systems*, volume 38, pp. 64392–64425. Curran Associates, Inc., 2025. doi: 10.52202/085713-2158.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kassraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivière, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Biao Zhang, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Jakub Adamek, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Bražinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Daniele Calandriello, Glenn Cameron, Charlotte Caucheteux, Rahma Chaabouni, Garima Chadha, Jetha Chan, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg,

Milen Ferev, Sebastian Flennerhag, Takumi Fujimoto, João Gabriel Oliveira, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Tamara von Glehn, Alec Go, Alhaad Gokhale, Alex Grills, Yiming Gu, Mayank Gupta, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang, Kevin Hui, Adrian Hutter, Te I, Fotis Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Ivan Korotkov, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Vivek Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Alon Levkovitch, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Ivan Lobov, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuszak, Daniel McDuff, Gaurav Menghani, Maciej Mikuła, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James O'Donnell, Brendan O'Donoghue, Kate Olszewska, Bernett Orlando, Wanqiong Pan, Rina Panigrahy, Unnati Parekh, Nicolas Perez-Nieves, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikołaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, George Scrivener, Benoit Seguin, Andrew Sellergren, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Bobak Shahriari, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Jean Tarbouriech, Chetan Tekur, Shantanu Thakoor, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Çağlar Ünlü, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiyi Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Cindy Wu, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Morteza Zadimoghaddam, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=u3vEuRr08MT.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJlxm30cKm.

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. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

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*, pp. 11279–11298, 2022a.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. URL https://arxiv.org/abs/2002.10957.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Zhenyi Wang, Li Shen, Le Fang, Qiuling Suo, Tiehang Duan, and Mingchen Gao. Improving task-free continual learning by distributionally robust memory evolution, 2022b. URL https://arxiv.org/abs/2207.07256.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

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

Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. Large language models are semi-parametric reinforcement learning agents. In *Advances in Neural Information Processing Systems*, volume 36, pp. 78227–78239. Curran Associates, Inc., 2023. doi: 10.52202/075280-3419.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. Lifelongagentbench: Evaluating llm agents as lifelong learners, 2025. URL https://arxiv.org/abs/2505.11942.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *International Conference on Learning Representations*, volume 2024, pp. 19036–19066, 2024.

7 MAIN NOTATION

Table 6: Main notation used in EARN.

Notation	Description
Problem Formulation	
$K,\ T_k,\ W$	Episode budget, steps in episode k, and number of final evaluation episodes.
$s_{k,t},\ a_{k,t},\ y_{k,t}$	Execution context, action, and environment feedback at step t of episode k.
$\theta,\ x_{k,t}$	Frozen LLM parameters and the prompt at step t of episode k.
$S_{k,t},\ S_k$	Total score after step t and episode score assigned by the scoring criterion.
J_{term}	Expected average episode score over the final W episodes.
Credit and Memory	
τ_k	Episode trajectory: execution contexts, actions, and feedback.
$B,\ F_b$	Block size and progress frontier fixed at the start of block b.
$e,\ S_e$	Scoring event and total score immediately after that event.
$c_{e,i}$	Block-wise credit assigned to action i in the segment for scoring event e.
$\tilde{r}_{k,t},\ C^{\mathrm{step}}_{k,t}$	Constructed numerical signal and step-wise credit.
$\mathcal{M}_k,\ \mathcal{R}^{\star}_k,\ \mathcal{E}_k$	Persistent memory, stored route, and action-credit records.
Probability Adjustment and Replay	
$Q_{\mathrm{blk}},\ Q_{\mathrm{step}}$	Block-wise and step-wise credit estimates.
$V_{\mathrm{blk}},\ V_{\mathrm{step}}$	Means subtracted to obtain the centered credit estimates.
$A_{\mathrm{blk}},\ A_{\mathrm{step}},\ A_{\mathcal{M}}$	Centered block-wise and step-wise credit estimates, and their combined credit estimate.
$n_{\mathrm{blk}},\ r_{\mathrm{blk}}$	Number of matching block-wise credit records and the resulting scaling factor.
$g_{\mathrm{blk}},\ g_{\mathrm{step}},\ q_{\mathrm{step}}$	Gates based on matching record counts and the largest scaling factor among matching step-wise records.
$p^0,\ p_{\mathcal{M}}$	Base and adjusted distributions over candidate actions.
$z_k,\ \chi_{k,t}$	Replay-episode indicator and replay-check indicator.

8 ADDITIONAL RELATED WORK

Attribution and Credit Assignment in Sequential Decision Making Credit assignment in sequential decision making aims to determine which historical actions led to delayed outcomes. RUDDER reallocates delayed rewards by decomposing returns, attributing them back to critical decision points; similarly, JitRL estimates the advantage of current candidate actions using returns from past trajectories (Arjona-Medina et al., 2019; Li et al., 2026).

Memory Retention Prior work studies example-level forgetting (Toneva et al., 2019) and memorization dynamics in language models (Tirumala et al., 2022). Continual learning methods mitigate forgetting through parameter regularization (Kirkpatrick et al., 2017), episodic-memory constraints (Lopez-Paz & Ranzato, 2022), and replay with small memory buffers (Chaudhry et al., 2019). Experience Replay for Continual Learning (Rolnick et al., 2019) uses replay to reduce forgetting, while subsequent work explores interference-based replay selection (Aljundi et al., 2019) and preservation of past model outputs (Buzzega et al., 2020). Do Your Best and Get Enough Rest for Continual Learning (Kang et al., 2025) examines how replay intervals affect retention. Distributionally robust memory evolution dynamically adjusts the memory data distribution to reduce buffer overfitting and alleviate forgetting (Wang et al., 2022b). In our setting with a frozen model, we define late-stage memory retention as the fraction of final episodes attaining the run's peak score.

Continual Learning for Agentic Systems Continual learning requires an agent to retain previously acquired effective behaviors while assimilating new experiences. Voyager achieves long-term capability accumulation in open worlds through automatic curriculum design and an executable skills library, whereas studies like LifelongAgentBench highlight that simply saving and replaying all historical experiences are prone to interference from irrelevant information and context length

limitations (Wang et al., 2024; Zheng et al., 2025). Our study addresses cross-episode adaptation under a fixed budget: the environment resets between episodes, but the memory persists.

9 Additional Method Details

9.1 Benchmarks and Tasks

We evaluate on three interactive benchmarks: Jericho, ScienceWorld, and Crafter. The selected environments cover text-based adventure games, scientific tasks, and visual survival gameplay.

Jericho. Jericho (Hausknecht et al., 2020) provides text-based interactive fiction environments. Agents read observations and issue commands to navigate locations, inspect objects, and interact with the game world. We use five games:

- **DeepHome.** The player explores an abandoned mountain kingdom and prepares it for its inhabitants' return by restoring essential systems, opening gates, and addressing threats.
- **Detective.** The player investigates the mayor's murder by visiting locations, examining information, collecting objects, and navigating potentially dangerous encounters.
- **Zork I.** The player explores the Great Underground Empire, solves puzzles, and collects treasures to return to the trophy case in the white house.
- **Zork III.** The player completes trials in the Great Underground Empire to demonstrate worthiness to succeed the Dungeon Master. Progress involves navigation, object use, and interactions with characters.
- **ZTUU.** In Zork: The Undiscovered Underground, the player explores a newly discovered underground region and seeks to emerge successfully by navigating rooms and solving object-based puzzles.

ScienceWorld. ScienceWorld (Wang et al., 2022a) is a text-based simulation environment for elementary science tasks. Agents manipulate objects, use instruments, and perform experiments to satisfy task-specific goals. We evaluate five tasks, each with multiple variations:

- **Conductivity.** Test whether a specified material conducts electricity by constructing a circuit and observing its behavior.
- **Unknown Conductivity.** Determine the electrical conductivity of an unknown substance through circuit-based testing.
- **Plant.** Grow a specified plant by preparing suitable growing conditions and providing the resources needed for its development.
- **Power.** Power an electrical component using an energy source that satisfies the requested renewable or nonrenewable category.
- **Melting Known.** Measure the melting point of a specified known substance using heating equipment and temperature measurements.

Crafter. Crafter (Hafner, 2022) is a visual, open-world survival game with procedurally generated environments. The agent gathers resources, crafts tools, builds useful structures, and manages survival needs. Progress is measured through 22 achievements, ranging from collecting basic resources to producing advanced tools. These achievements have dependencies, requiring the agent to carry out sequences of preparatory actions. We use a multimodal interface combining visual observations with textual information.

Memory storage. Alongside the stored route and action-credit records, persistent memory retains aggregate credit statistics and trajectories awaiting block-wise selection.

9.2 Block Size and Progress Frontier

Block size. We use blocks of $B = 5$ acquisition episodes and select the best-performing trajectory within each block for milestone credit attribution. This provides several trajectories for comparison while keeping the delay before block-wise memory updates short. Smaller blocks offer fewer alternatives, whereas larger blocks delay the use of selected experience. Step-wise credit records continue to update after every episode.

This choice is inspired by self-consistency: Wang et al. (2023) suggest starting with five or ten reasoning paths to balance performance and computational cost. Unlike their answer aggregation procedure, EARN selects among sequential interaction trajectories using environment feedback. We therefore adopt $B = 5$ as a practical setting.

Progress Frontier. Let P_k denote the route score from episode k used to update the progress frontier. We locate the first action that reaches the episode's highest total score. If this action terminates the environment, P_k is the score immediately before it; otherwise, P_k is the score immediately after it. This rule also applies to successful termination.

After episode k, we consider the most recent $W_F = 5$ episodes, including episode k, indexed by $\mathcal{W}_k = \{\max(1, k - W_F + 1), \ldots, k\}$. We round each P_j to six decimal places before comparing values. The candidate frontier is the highest positive route score recorded in at least two episodes within this window:

$$\widehat{F}_k = \max\left(\{0\} \cup \left\{ r > 0 : \sum_{j \in \mathcal{W}_k} \mathbf{1}[P_j = r] \geq 2 \right\}\right). \tag{11}$$

Here, $\mathbf{1}[\cdot]$ is the indicator function. Each episode contributes one route score, rather than all intermediate scores along its trajectory.

Let F_k^{sys} denote the progress frontier after episode k. We initialize it to zero and update it without allowing decreases:

$$F_0^{\mathrm{sys}} = 0, \qquad F_k^{\mathrm{sys}} = \max\left(F_{k-1}^{\mathrm{sys}}, \widehat{F}_k\right). \tag{12}$$

If no positive route score appears at least twice in the window, $\widehat{F}_k = 0$ and the frontier remains unchanged.

Let k_b denote the first episode of block b. We fix the block's frontier before that episode:

$$F_b = F_{k_b - 1}^{\mathrm{sys}}. \tag{13}$$

All block-wise credit attribution in block b uses this fixed value, even if the system frontier changes during the block. When $F_b = 0$, the new-progress segment starts at the beginning of the trajectory, subject to the segment-length limit.

Example of Frontier Updates. Suppose a block has just ended, the frontier before the latest update is 10, and the five most recent route scores are $(10, 12, 10, 15, 12)$. The candidate frontier is $\widehat{F}_k = 12$: both 10 and 12 appear twice, whereas 15 appears only once. Thus, $F_k^{\mathrm{sys}} = \max(10, 12) = 12$. If the next episode starts a new block, that block uses $F_b = 12$. Score-increase events ending at or below 12 receive reduced credit, while those ending above 12 receive higher credit. The block's reference remains fixed at 12 throughout the block.

Trajectory selection and credited segments. After comparing the highest total scores, trajectory selection breaks ties by the score of the replayable prefix, fewer steps to the highest score, absence of terminal failure at that point, fewer local cycles, and earlier episode, in that order. Let t_e be the step of the eth score increase in the selected trajectory, so $S_e = S_{k,t_e}$. If $S_e \leq F_b$, the credited segment begins after the preceding score increase. If $S_e > F_b$, the segment begins after the trajectory first reaches or exceeds F_b; when the score increase at t_e first crosses F_b, the segment instead begins after the preceding score increase. Each segment ends at t_e and retains only the most recent actions if its length exceeds the configured limit. We use $\rho = 1/2$ in Equation 5. We limit repeated admission of identical action segments.

Notation for Block-wise Credit Attribution. The symbols in the credit attribution formula are defined as follows:

- e: A scoring event in the selected trajectory, corresponding to a positive increase in the environment score. A trajectory may contain multiple scoring events.
- L_e: The number of actions in the attribution segment associated with event e.
- i: The action's position within this segment, indexed from 1 to L_e. The final position $i = L_e$ corresponds to the scoring action.
- ℓ_e: The initial reference value for interpolation, corresponding to relative position $i/L_e = 0$. It equals 0 for repeated progress and ρ for new progress.
- h_e: The terminal interpolation value, equal to the scoring action's credit. It equals α_{pre} for repeated progress and 1 for new progress.
- $c_{e,i}$: The credit assigned to the i-th action in the segment for event e.

Since i starts at 1, the first action receives $\ell_e + (h_e - \ell_e)/L_e$, rather than ℓ_e. The final action receives h_e.

9.3 Block-wise Credit Estimates

For execution context s and candidate action a, let $\{c_j\}_{j=1}^{n_{\text{blk}}}$ denote the block-wise credits in records matching both s and a exactly. For $n_{\text{blk}} > 0$, their mean μ_{blk} and sample variance σ^2_{blk} are:

$$\mu_{\text{blk}} = \frac{1}{n_{\text{blk}}} \sum_{j=1}^{n_{\text{blk}}} c_j, \qquad \sigma^2_{\text{blk}} = \begin{cases} \dfrac{1}{n_{\text{blk}} - 1} \displaystyle\sum_{j=1}^{n_{\text{blk}}} (c_j - \mu_{\text{blk}})^2, & n_{\text{blk}} > 1, \\ 0, & n_{\text{blk}} = 1. \end{cases} \tag{14}$$

The number of matching records n_{blk} is not an independent sample size: credits may come from the same trajectory.

For $n_{\text{blk}} > 0$, the scaling factor is $r_{\text{blk}}(s, a) = n_{\text{blk}}/(n_{\text{blk}} + \nu)$; we set $r_{\text{blk}}(s, a) = 0$ when no records match. When $\nu = 0$ and at least one record matches, $r_{\text{blk}}(s, a) = 1$. The centering mean $V_{\text{blk}}(s)$ averages the block-wise credit estimates $Q_{\text{blk}}(s, a)$ over candidate actions $\mathcal{A}(s)$:

$$V_{\text{blk}}(s) = \begin{cases} \dfrac{\sum_{a \in \mathcal{A}(s)} r_{\text{blk}}(s, a) Q_{\text{blk}}(s, a)}{\sum_{a \in \mathcal{A}(s)} r_{\text{blk}}(s, a)}, & \displaystyle\sum_{a \in \mathcal{A}(s)} r_{\text{blk}}(s, a) > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

For $n_{\text{blk}} = 0$, we set $Q_{\text{blk}}(s, a) = 0$ without evaluating Equation 7. For $n_{\text{blk}} = 1$, we set the sample variance to zero, so only the scaling factor r_{blk} reduces the estimate. A single credit does not establish that later executions will receive the same credit. The variance correction in Equation 7 reduces the magnitude of the block-wise credit estimate toward zero without reversing its sign. We fix $\nu = 3$, so three matching records receive a count-based scaling factor of 0.5. A single record can still contribute, but receives less weight, which increases as more records accumulate. We fix $\xi = 0.5$ to account for variation among credit records. For the same mean, variance, and record count, this is more conservative than no variability correction ($\xi = 0$), while retaining at least as much credit magnitude as a unit correction ($\xi = 1$).

9.4 Step-wise Credit Estimates

Let $\mathcal{N}(s)$ denote the step-wise credit records retrieved for execution context s, including records from similar observations when semantic retrieval is enabled. Let $\mathcal{N}(s, a)$ denote the subset whose actions match candidate action a. For record j, let C_j^{step} be its assigned step-wise credit. We multiply each credit by a scaling factor q_j:

$$q_j = \begin{cases} q_{\text{replay}}, & \text{episode designated for route replay,} \\ 1, & \text{other episode, with progress beyond its initial frontier,} \\ q_{\text{pre}}, & \text{otherwise.} \end{cases} \tag{16}$$

Progress beyond the episode's initial frontier means that the total score immediately before or after the action exceeds the progress frontier before that episode's memory update. The configured factors q_{replay} and q_{pre} scale credits from replay episodes and other credits at or below the frontier, respectively. Credits from replay episodes receive q_{replay} regardless of total score. For nonempty $\mathcal{N}(s)$, the centering mean $V_{\text{step}}(s)$ is

$$V_{\text{step}}(s) = \frac{\sum_{j \in \mathcal{N}(s)} q_j C_j^{\text{step}}}{|\mathcal{N}(s)|}. \tag{17}$$

The step-wise credit estimate $Q_{\text{step}}(s, a)$ averages $q_j C_j^{\text{step}}$ over $\mathcal{N}(s, a)$, whereas $V_{\text{step}}(s)$ averages over all retrieved step-wise records $\mathcal{N}(s)$. Both denominators count records; neither denominator sums the scaling factors.

If no step-wise records match candidate action a, the step-wise gate g_{step} is zero and that action receives no step-wise contribution to the combined credit estimate $A_{\mathcal{M}}(s, a)$.

Support Gates. Support gates allow limited evidence to influence decisions while reducing its contribution when few records match. For ordinary exact-context block-wise retrieval, we use

$$g_{\text{blk}}(s, a) = \begin{cases} 0, & n_{\text{blk}} = 0, \\ \dfrac{n_{\text{blk}}}{n_{\text{blk}} + \kappa}, & 0 < n_{\text{blk}} < m_{\text{blk}}, \\ 1, & n_{\text{blk}} \geq m_{\text{blk}}, \end{cases} \tag{18}$$

with $m_{\text{blk}} = 2$ and $\kappa = 0.8$. Thus, a single matching record receives a gate of approximately 0.56, while two or more records receive a gate of 1. This allows an initial observation to contribute without giving it the full channel weight.

For step-wise retrieval, let $n_{\text{step}} = |\mathcal{N}(s, a)|$ be the number of matching records. We use

$$g_{\text{step}}(s, a) = \min\left(1, \frac{n_{\text{step}}}{m_{\text{step}}}\right) \tag{19}$$

This gate increases linearly with the record count and reaches 1 at three records. We set $m_{\text{step}} = 3$, giving gate values of $1/3$, $2/3$, and 1 for one, two, and three matching records, respectively. This gradually increases the step-wise contribution as records accumulate. Both gates operate before standardization.

Algorithm, episode and block updates. Algorithm 1 summarizes the update order.

9.5 Action Selection and Route Replay

Context and Action Matching. Execution contexts combine normalized observations with supplementary state information and an environment-provided state identifier when available. Actions are matched after normalizing case, whitespace, and surrounding punctuation.

Retrieval from Similar Observations. In Jericho and ScienceWorld, semantic retrieval supplements insufficient exact matches with records from similar observations. For block-wise retrieval, estimates from similar contexts are combined using observation-similarity weights and a transfer coefficient. Step-wise retrieval instead pools additional records before estimating action values. For Crafter, we disable this observation-text retrieval mechanism in our multimodal configuration and use exact-context retrieval for action evidence. Similarity-based retrieval does not replace the context checks required for route replay.

Memory-Based Probability Adjustment. We normalize the LLM-reported confidence scores. When memory candidates are added, we assign them initial weights and renormalize to obtain $p^0(a \mid s)$. Fused advantages are standardized using the candidate mean and population standard deviation, then clipped to $[-\zeta, \zeta]$. If the standard deviation is below 10^{-8}, the divisor is replaced by the larger of one and the maximum absolute centered advantage. Base probabilities are floored at 10^{-12} before taking logarithms. We fix $\zeta = 3$ and clip standardized estimates to $[-3, 3]$ before probability adjustment, bounding the credit-dependent exponent to $[-3\beta, 3\beta]$. For a final candidate set of at most five actions, population-standardized estimates lie within $[-2, 2]$, so this bound leaves them unchanged.

Algorithm 1 EARN

Require: Frozen LLM f_θ; budgets K, H; block size B; consolidation length K_c

Initialize $\mathcal{M}, \mathcal{T} \leftarrow \emptyset, F \leftarrow 0$
for $k = 1, \ldots, K$ **do**
  Reset environment; initialize s and $\tau_k \leftarrow \emptyset$
  If $k \leq K - K_c$ and $\mathcal{T} = \emptyset$, set $F_b \leftarrow F$
  $\mathcal{R}^\star \leftarrow \mathrm{RetrieveRoute}(\mathcal{M})$; $m \leftarrow 1$
  $r \leftarrow (k > K - K_c) \wedge (\mathcal{R}^\star \neq \emptyset)$
  Conditional replay or action-probability adjustment
  for up to H steps, until termination **do**
    $r \leftarrow r \wedge \mathrm{ReplayValid}(\mathcal{R}^\star, m, s)$
    if r **then**
      $a \leftarrow a^\star_m$; $m \leftarrow m + 1$
    else
      $(\mathcal{A}, p^0) \leftarrow \mathrm{ProposeAndNormalize}(f_\theta, s, \mathcal{M})$
      Estimate $A_{\mathrm{blk}}, A_{\mathrm{step}}, g_{\mathrm{blk}}, g_{\mathrm{step}}, q_{\mathrm{step}}$ from retrieved records
      $A_{\mathcal{M}} \leftarrow g_{\mathrm{blk}} A_{\mathrm{blk}} + \lambda_{\mathrm{step}}[1 - \eta g_{\mathrm{blk}}]_+ g_{\mathrm{step}} q_{\mathrm{step}} A_{\mathrm{step}}$
      $\widehat{A}_{\mathcal{M}} \leftarrow \mathrm{StandardizeAndClip}(A_{\mathcal{M}})$
      $p_{\mathcal{M}}(a \mid s) \propto p^0(a \mid s) \exp(\beta \widehat{A}_{\mathcal{M}}(s, a))$
      $a \leftarrow \mathrm{Select}(p_{\mathcal{M}})$
    end if
    Execute a; observe feedback and next context s'
    Append transition and execution metadata to τ_k; $s \leftarrow s'$
  end for
  Every episode: update step-wise credit and routes
  $\mathcal{C}^{\mathrm{step}}_k \leftarrow \mathrm{StepCredit}(\tau_k, F)$
  Update step-wise records in $\mathcal{M}$ using $\mathcal{C}^{\mathrm{step}}_k$
  Update stored routes in $\mathcal{M}$ using τ_k
  At block completion: select and attribute credit
  if $k \leq K - K_c$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\tau_k\}$
    if $|\mathcal{T}| = B$ or $k = K - K_c$ **then**
      $\tau^{\mathrm{sel}} \leftarrow \mathrm{SelectBest}(\mathcal{T})$
      Update block-wise records in $\mathcal{M}$ with $\mathrm{BlockCredit}(\tau^{\mathrm{sel}}, F_b)$
      $\mathcal{T} \leftarrow \emptyset$
    end if
  end if
  Update F using repeated recent positive route scores
end for
return $\mathcal{M}$

Route Storage and Replay. Route memory is updated after each episode, independently of block-wise credit attribution. Candidate routes are compared by score, with shorter routes preferred when scores tie. Before each replayed action, the agent checks context agreement, action availability when a valid-action list is supplied, and stored failure-based restrictions. A failed check or exhausted route ends replay for the current episode, and the agent resumes memory-guided candidate generation and probability adjustment.

10 Additional Experimental Settings

Evaluation window and replay schedule. We evaluate the final $W = 4$ episodes and use blocks of $B = 5$ episodes for block-wise credit attribution. The evaluation window and block size are separate protocol choices. Conditional route replay is enabled in the final four episodes. Step-wise credit attribution uses every episode, including episodes designated for replay.

Hyperparameters and Hardware.

- **Discount factor ($\gamma = 0.95$).** This setting allows scoring-event feedback to reach earlier preparatory actions while decaying with temporal distance. An action occurring ten steps before a scoring event receives that event's signal with a discount weight of $0.95^{10} \approx 0.60$.

- **Probability-adjustment strength ($\beta = 0.75$).** This coefficient controls how strongly memory changes the base action probabilities. When two candidates differ by one unit in their standardized and clipped credit estimates, the adjustment multiplies their base probability ratio by $e^{0.75} \approx 2.12$ in favor of the higher-valued candidate. Memory can therefore meaningfully influence selection while the LLM's base distribution continues to contribute to the adjusted probabilities.
- **Block-wise credit parameters ($\alpha_{\mathrm{pre}} = 0.02$, $\rho = 0.5$).** For repeated-progress segments, credits follow $0.02i/L_e$, reaching 0.02 at the scoring action. For new-progress segments, credits follow $0.5 + 0.5i/L_e$, reaching 1 at the scoring action. This assigns higher credit to new progress while retaining small positive credits for repeated progress.
- **Base step-wise weight ($\lambda_{\mathrm{step}} = 0.35$).** The base weight of step-wise credit is fixed at 0.35. Its contribution to the combined estimate is further regulated by block-wise support, the step-wise support gate, and record scaling.
- **Step-wise suppression coefficient ($\eta = 0.85$).** The block-wise support gate $g_{\mathrm{blk}}(s, a) \in [0, 1]$ modulates the step-wise coefficient:

  $$\lambda_{\mathrm{step}}[1 - \eta g_{\mathrm{blk}}(s, a)]_+ = 0.35[1 - 0.85 g_{\mathrm{blk}}(s, a)]_+.$$

  This coefficient decreases from 0.35 when $g_{\mathrm{blk}} = 0$ to 0.0525 when $g_{\mathrm{blk}} = 1$, before applying the step-wise gate g_{step} and record scale q_{step}. Thus, stronger block-wise support attenuates the step-wise contribution before standardization without eliminating it through suppression alone.

For EARN, the generation temperature is 0.8, and the LLM is requested to propose five candidate actions per decision. We serve the W4A16-quantized Gemma-4-31B-it checkpoint using vLLM on a single NVIDIA H100 GPU. Model parameters remain frozen throughout evaluation.

11 Inference Savings from Conditional Replay

Measurement. We measure how conditional replay reduces LLM candidate-generation calls on the Jericho runs reported in Table 1. A replayed action is still executed in the environment, but bypasses LLM candidate generation. Thus, each replayed step avoids one logical generation call without reducing the number of environment interactions. We count replayed steps from transition logs and verify that non-replay steps match the corresponding decision-history records. This measurement excludes API retries, token consumption, and latency.

Metrics. For game g, let $T_{g,k}$ denote the number of executed steps in episode k, and let $N_{g,k}^{\mathrm{replay}}$ denote the number of directly replayed steps. Table 7 reports three percentages:

$$\begin{aligned} \mathrm{Save}_g^{\text{Last-4}} &= \frac{100}{4} \sum_{k=K-3}^{K} \frac{N_{g,k}^{\mathrm{replay}}}{T_{g,k}}, \\ \mathrm{Save}_g^{\mathrm{Last}} &= 100 \frac{N_{g,K}^{\mathrm{replay}}}{T_{g,K}}, \\ \mathrm{Save}_g^{\mathrm{Final/Run}} &= 100 \frac{\sum_{k=K-3}^{K} N_{g,k}^{\mathrm{replay}}}{\sum_{k=1}^{K} T_{g,k}}. \end{aligned} \tag{20}$$

Last-4 averages the per-episode saving rates over the final four episodes, and Last reports the final episode's rate. Final/Run divides replayed steps in the final four episodes by all executed steps in the run. Each reported average gives equal weight to the five games.

Results. Conditional replay avoids 78.90–86.27% of candidate-generation calls in the final four episodes, averaged equally across games. These final-window savings correspond to 12.39%, 7.76%, and 4.94% of all executed steps under the 30/30, 50/50, and 70/70 budgets, respectively. When replay becomes inapplicable, the agent resumes LLM candidate generation and probability adjustment. These counts measure the generation calls bypassed during the recorded runs. They do not measure how episode scores would change if the agent generated an action at every step.

Table 7: Candidate-generation call savings (%) for EARN on five Jericho games. Last-4 averages the final four per-episode rates; Last reports the final episode. Final/Run counts only replayed steps in the final four episodes, divided by all executed steps in the run. Average gives equal weight to each game. Budget labels give episodes/maximum steps per episode.

Game	**30/30**			**50/50**			**70/70**		
	Last-4	**Last**	**Final/Run**	**Last-4**	**Last**	**Final/Run**	**Last-4**	**Last**	**Final/Run**
DeepHome	93.08	100.00	15.84	84.88	84.00	9.05	90.00	90.00	5.22
Detective	100.00	100.00	14.13	97.87	97.87	9.24	97.22	97.22	5.46
Zork I	76.67	90.00	10.98	88.00	88.00	8.22	78.28	90.91	5.35
Zork III	84.85	90.91	10.67	50.55	48.65	4.21	92.69	82.76	3.96
ZTUU	73.33	73.33	10.32	73.23	82.00	8.07	73.17	50.00	4.72
Average	85.59	90.85	12.39	78.90	80.10	7.76	86.27	82.18	4.94

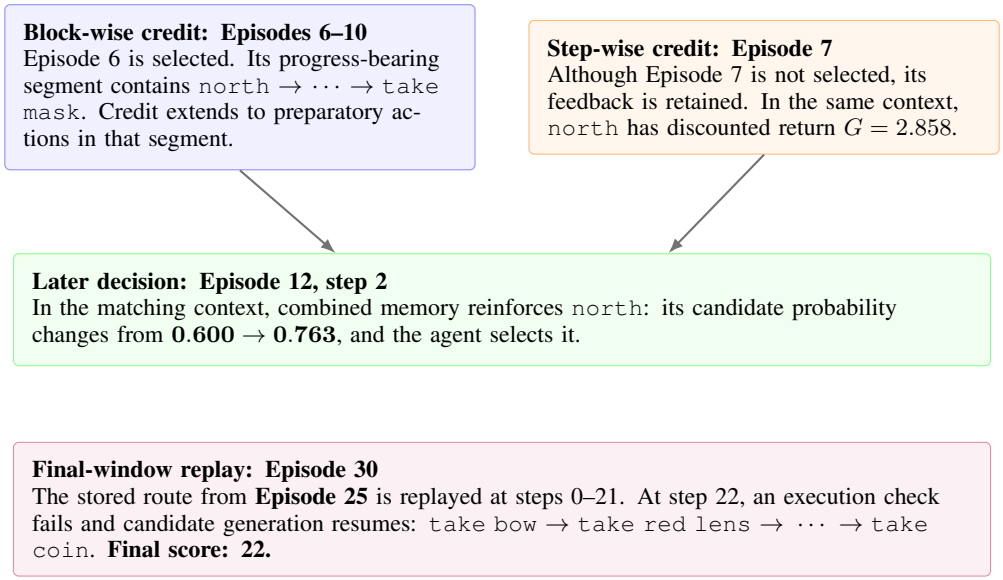


Figure 5: ZTUU case study in the 30-episode, 30-step setting. The upper panels illustrate the two credit sources and a later combined-memory decision. The lower panel separately shows conditional replay and fallback in the final episode. The probability change reflects accumulated memory, not an isolated contribution from either displayed episode.

12 Case Study

Case analysis. Figure 5 shows that the key to conditional replay is that it reuses a validated route prefix without locking the agent into the entire stored trajectory. Block-wise credit emphasizes progress in high-performing trajectories, while step-wise credit retains feedback from other episodes; when replay no longer applies, the agent can return to memory-guided action generation. ZTUU's final episode illustrates this flexibility. The first 22 replayed steps bring the score to 12. After the replay diverges at step 22, the agent resumes generation, obtains further progress through `take red lens` and `take coin`, and recovers its previous best score of 22.

Thus, replay preserves a reliable prefix while leaving the remaining steps open to adaptation, rather than forcing the agent to repeat an entire old route. The avoided LLM calls could potentially be reallocated to additional exploration under a fixed call budget, provided further environment interactions are available; this reallocation is not evaluated here.

EARN Action Generation Prompt

System:
You are an agent interacting with an environment. Use the current observation and past experience to propose actions that make progress toward the task objective.

Task: {task_description}

Current observation:
{observation}

Recent interaction history:
{recent_history}

Cross-episode memory:
The agent's memory module retrieved the following records for the current or similar execution contexts. They describe previously attempted actions and their observed outcomes. Consider their relevance to the current situation when proposing actions.
{retrieved action--outcome records}

Available actions, when provided:
{available_actions}

Output instruction:
Propose five distinct candidate actions with brief reasoning. Assign each candidate action an integer confidence score from 0 to 100 indicating your relative preference for selecting that action. The five scores must sum to 100. Follow the provided action constraints.

Figure 6: EARN general prompt structure. Observations are textual in Jericho and ScienceWorld and include visual input in Crafter.

13 Additional Controlled Comparisons

13.1 Baselines with Their Native Implementations

We run Reflexion, AWM, EvoTest, and JitRL following the protocols in their publicly available implementations. Each retains its method-specific decoding, memory, and retrieval settings. For FIFO Memory, we prepend full transcripts of previous episodes to the prompt and remove the oldest transcripts when the memory context budget is exceeded. Within each benchmark, all methods use the same frozen backbone, task instances, interaction budgets, and scoring protocol. This preserves their distinct decision mechanisms while controlling the model and evaluation conditions.

13.2 EARN under the Matched Evaluation Protocol

EARN uses the same backbone and evaluation protocol as the baselines, without parameter updates. For EARN, we set the action-generation temperature to 0.8 across all three benchmarks, five candidate actions, and all-MiniLM-L6-v2 embeddings for semantic retrieval.

13.3 Prompt Template

EARN programmatically retrieves historical action–outcome records from persistent memory and formats them into the prompt, without additional LLM summarization calls. The following simplified template illustrates the structure used for candidate-action generation.