Just-In-Time Reinforcement Learning: Continual Learning in LLM Agents Without Gradient Updates

Yibo Li [1] **Zijie Lin** [1] **Ailin Deng** [1] **Xuan Zhang** [1] **Yufei He** [1] **Shuo Ji** [1] **Tri Cao** [1] **Bryan Hooi** [1]

Abstract

While Large Language Model (LLM) agents excel at general tasks, they inherently struggle with continual adaptation due to the frozen weights after deployment. Conventional reinforcement learning (RL) offers a solution but incurs prohibitive computational costs and risk of catastrophic forgetting. We introduce **Just-In-Time Reinforcement Learning (JitRL)**, a training-free framework that enables test-time policy optimization without any gradient updates. JitRL maintains a dynamic, non-parametric memory of experiences and retrieves relevant trajectories to estimate action advantages on-the-fly. These estimates are then used to directly modulate the LLM's output logits. We theoretically prove that this additive update rule is the exact closed-form solution to the KL-constrained policy optimization objective. Extensive experiments on WebArena and Jericho demonstrate that JitRL establishes a new state-of-the-art among training-free methods. Crucially, JitRL outperforms the performance of computationally expensive fine-tuning methods (e.g., WebRL) while reducing monetary costs by over 50×, offering a scalable path for continual learning agents.

1. Introduction

While humans can learn "on the fly", current AI agents fundamentally lack this ability because their weights are frozen after training (Gao et al., 2025; Fang et al., 2025). This limitation severely restricts their practicality: when deployed in unfamiliar or dynamic settings, AI agents are analogous to newly hired employees without on-the-job training, which often repeatedly make the same errors due to their failure to learn, whereas humans can adapt to their environment, including by learning from their mistakes. Highlighting the importance of this issue, Hendrycks et al. (2025) proposes an "AGI Score" assessing progress toward artificial general intelligence and finds that current AI systems are most deficient in their "capability to continually learn new information".

One potential solution is conventional reinforcement learning (RL) (Qi et al., 2025; Qian et al., 2025; Li et al., 2025), performed in a continually policy gradient updating manner. However, this approach requires a large amount of training data to perform well. It is also computationally expensive, due to the high cost of RL and the need for frequent updates to keep the model up to date. In addition, it is prone to catastrophic forgetting, degrading the model's previously learned capabilities (Li et al., 2024). Prior work finds that conventional RL yields only limited improvements when evaluated in continual adaptation settings (He et al., 2025).

An alternative approach to this problem relies on in-context learning (ICL) (Dong et al., 2024) to enable the agent to learn at test time (Shinn et al., 2023; Wang et al., 2025). These methods rely on prompt or context engineering to incorporate information from past experiences. However, such approaches tend to struggle as the required context length grows, especially since agentic tasks often involve long sequences of interactions (Liu et al., 2024; Zhang et al., 2025). Crucially, ICL lacks the generality of RL. While prompts are confined to explicit textual descriptions, RL optimizes policies through rewards, enabling the mastery of complex skills that are difficult to articulate in text.

In light of these limitations, a fundamental question arises: *Can we enable agents to learn continually in a flexible way, without expensive parameter updates?* Our approach, Just-In-Time Reinforcement Learning (JitRL), enables general-purpose learning by adopting an RL formulation, while maintaining efficiency by avoiding gradient updates. Instead of gradient updates, JitRL maintains a dynamic memory bank that stores trajectories as `<state, action, reward>` triplets, as illustrated in Figure 1. Then, given the agent's current state, it retrieves relevant trajectories from the memory, and learns from them "just in time." To do so, it uses these trajectories to estimate the advantage of each action in the current state – i.e., how much better each action is relative to the average. These advantage es-

[1]National University of Singapore, Singapore. Correspondence to: Bryan Hooi <bhooi@comp.nus.edu.sg>.

Preprint.

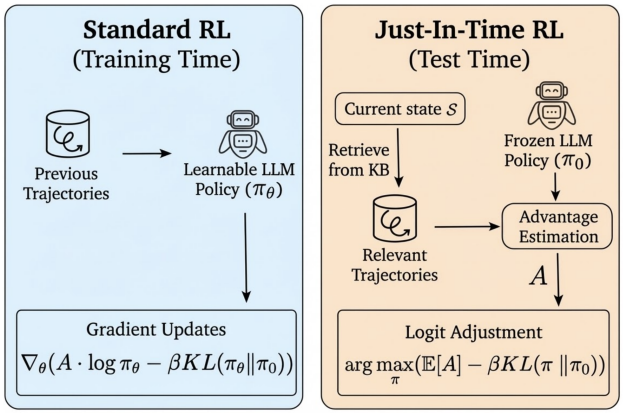


Figure 1. **Standard RL vs. Just-In-Time RL (JitRL).** Standard RL performs policy gradient updates at training time. In contrast, at test time, JitRL retrieves trajectories relevant to the current state s, uses them to estimate advantages A, which it uses to adjust the output logits based on a KL-regularized policy optimization objective.

timates are then used to adjust the model's output logits. Crucially, we theoretically prove that our update rule is the exact closed-form solution to the policy optimization objective under a KL constraint. In this way, JitRL enables continual self-evolution without the prohibitive costs of gradient-based training.

We empirically validate JitRL on two benchmarks: WebArena (Zhou et al., 2024) for realistic web navigation and Jericho (Hausknecht et al., 2020) for long-horizon text-based games. Extensive experiments demonstrate that JitRL sets a new state-of-the-art, outperforming training-free baselines and weight-update methods, while reducing the monetary costs of conventional RL by over 50×. Furthermore, our results highlight JitRL's robustness, showing consistent gains across diverse LLM backbones and effective generalization to unseen tasks.

2. Related Work

Reinforcement Learning. Reinforcement learning (RL) algorithms like PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024) have effectively aligned LLM agents for complex tasks. This paradigm spans diverse domains, ranging from optimizing search queries in information retrieval (Jin et al., 2025; Song et al., 2025) to refining tool execution (Li et al., 2025; Qian et al., 2025). However, these gradient-based methods suffer from prohibitive computational costs and yield static models, fundamentally limiting their adaptability to distribution shifts.

Training-Free Inference Enhancement. Recent works have integrated external memory modules to enhance agents directly at test time. Systems like MemGPT and Generative Agents (Packer et al., 2023; Park et al., 2023) utilize hierarchical memory to store historical interactions, while frameworks like Voyager and Reflexion (Wang et al., 2023; Shinn et al., 2023) retrieve textual descriptions of past skills or failures to improve future performance. Similarly, A-mem (Xu et al., 2025) constructs interconnected knowledge networks through dynamic indexing. Rather than merely retrieving text for in-context learning, our method treats memory as a non-parametric policy distribution. We perform soft updates directly on the LLM's logits, effectively achieving policy improvement without the overhead of parameter updates.

3. Preliminaries

Reinforcement Learning (RL) aims to optimize a policy π_θ, which maps a given state to a probability distribution over actions, to maximize the expected cumulative reward. The objective function is defined as $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$, where $\tau = (s_0, a_0, r_0, s_1, a_1, r_1 \ldots, s_n, a_n, r_n)$ represents a trajectory of n interactions, and $R(\tau)$ denotes the cumulative discounted return of that trajectory.

A fundamental approach to optimize this objective is the Policy Gradient method (Williams, 1992; Sutton et al., 1999). Intuitively, this method shifts the probability mass towards actions that yield higher returns by updating parameters in the direction of the gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s,a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) \cdot A(s,a)], \quad (1)$$

where $A(s,a)$ is the **advantage function**. The advantage function serves as a critic to evaluate the selected action relative to a baseline, formulated as:

$$A(s,a) = Q(s,a) - V(s). \quad (2)$$

Here, $Q(s,a)$ is the **action-value function**, representing the expected return of taking action a in state s, while $V(s)$ is the **state-value function**, representing the average expected return of being in state s. Thus, $A(s,a)$ quantifies *how much better* a specific action is compared to the average performance of the policy in that state.

Reinforcement Learning typically requires the training of additional value networks to estimate the advantage. This process is computationally expensive and results in a static model that lacks the flexibility to adapt at test time.

4. Method

To address the limitations of the gradient-based RL, we present **Just-In-Time Reinforcement Learning (JitRL)** framework. Instead of updating model parameters θ, JitRL functions as a test-time policy optimization method that modulates a frozen prior π_θ towards an optimal posterior π^*. As illustrated in Figure 2, the framework consists of three key components: (1) constructing a granular experience

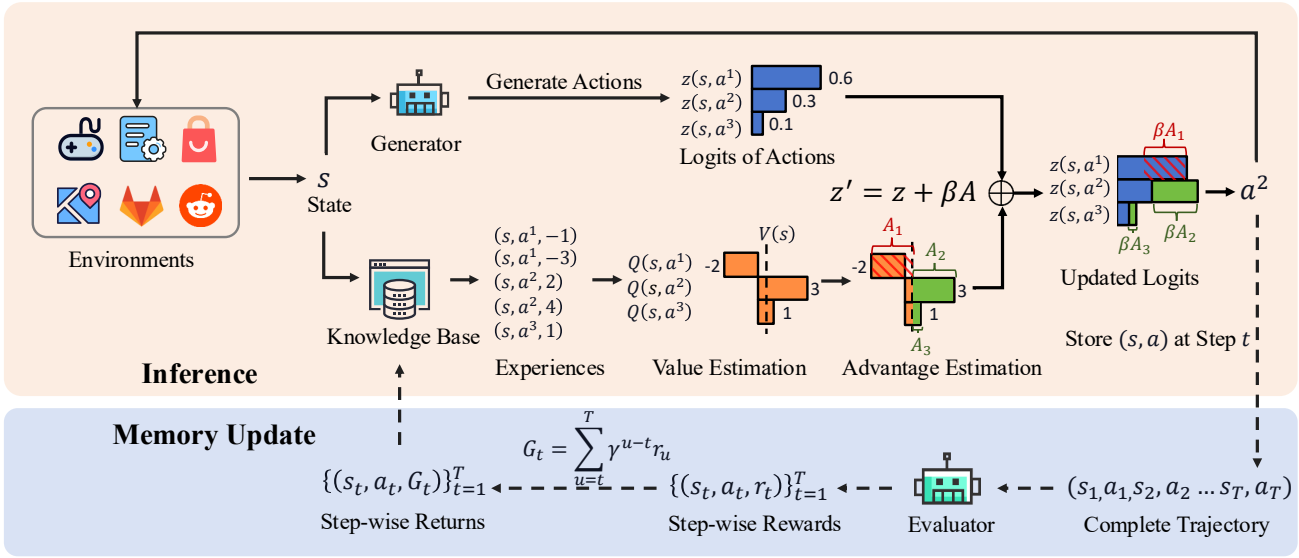


Figure 2. **Overview of the Just-In-Time Reinforcement Learning (JitRL) framework.** The system operates in a continuous loop comprising two streams: (1) In the *Inference Stream* (top), the agent retrieves relevant past experiences $\mathcal{N}(s)$ from the non-parametric memory $\mathcal{M}$. The base LLM's logits z are then adjusted in closed-form ($z' = z + \beta A$) using the estimated advantage A derived from historical returns, enabling test-time policy improvement without gradient updates. (2) In the *Memory Update* (bottom), completed trajectories are analyzed by an evaluator to compute discounted returns G_t. These new experiences are stored back into $\mathcal{M}$, allowing the agent to evolve its policy across episodes.

memory from completed trajectories, (2) estimating state and action values via retrieval during inference, and (3) dynamically updating the LLM's policy logits based on estimated advantages. The overall algorithm can be found in Appendix A.

4.1. Memory Construction

In the absence of a trained value network, we estimate expected returns by querying a dynamic memory $\mathcal{M} = \{(s_i, a_i, G_i)\}_{i=1}^{N}$, where G_i denotes the discounted reward. $\mathcal{M}$ implicitly represents the empirical distribution of the environment's dynamics. Upon the completion of an episode, we first assign a specific reward r_t to each step using an evaluator. These rewards are then aggregated into the discounted return G_t to quantify the long-term value of each action:

$$G_t = \sum_{u=t}^{T} \gamma^{u-t} r_u, \tag{3}$$

where $\gamma \in [0, 1]$ balances the weight of immediate versus future rewards.

Raw states (e.g., full HTML DOM trees or verbose game text) are often too noisy for effective retrieval. We abstract them into compact, structured states s_t that preserve task-relevant semantics while discarding irrelevant details. Our key design principle is mapping functionally equivalent states to similar representations.

Ultimately, each transition is stored in the dynamic memory $\mathcal{M}$ as a compact triplet (s_t, a_t, G_t), implicitly representing the empirical distribution of the environment's dynamics.

4.2. Test-Time Value Estimation

Instead of relying on computationally expensive value networks, we perform on-the-fly value estimation by retrieving relevant transitions from memory. During inference, the current observation is abstracted into a structured state s. We then retrieve top-k similar neighbors to form a local neighborhood $\mathcal{N}(s)$.

We formulate the **state value** estimation $\widehat{V}(s)$ for state s as an expectation of the returns across this neighborhood:

$$\widehat{V}(s) := \frac{1}{|\mathcal{N}(s)|} \sum_{i \in \mathcal{N}(s)} G_i \xrightarrow{p} \mathbb{E}_{\tau \sim \mathcal{N}(s)}[G]. \tag{4}$$

Similarly, to evaluate a specific candidate action a, we condition the **action value** $\widehat{Q}(s, a)$ on the state-action pair (s, a) over the relevant subset $\mathcal{N}(s, a) = \{(s_i, a_i, G_i) \in \mathcal{N}(s) : a_i = a\}$: We distinguish between two cases:

1. **Known Actions:** If historical evidence exists (i.e., $|\mathcal{N}(s, a)| > 0$), we estimate the action value by averaging the returns within the subset:

$$\widehat{Q}(s, a) := \frac{1}{|\mathcal{N}(s, a)|} \sum_{j \in \mathcal{N}(s,a)} G_j \xrightarrow{p} \mathbb{E}_{\tau \sim \mathcal{N}(s,a)}[G]. \tag{5}$$

2. **Unseen Actions:** For actions lacking historical data (i.e., $|\mathcal{N}(s,a)| = 0$), we encourage exploration by adopting an *optimism under uncertainty* principle inspired by UCB (Auer et al., 2002) With probability λ, we assign an optimistic value:

$$\widehat{Q}(s,a) \coloneqq V(s) + \frac{\alpha}{|\mathcal{N}(s)|}. \tag{6}$$

The bonus $\alpha/|\mathcal{N}(s)|$ reflects epistemic uncertainty: sparse memory coverage (small $|\mathcal{N}(s)|$) implies unreliable estimates, warranting exploration. As experiences accumulate, the bonus diminishes, naturally shifting toward exploitation. With probability $1-\lambda$, we assign $Q(s,a) = 0$ to prevent over-exploration.

The **test-time advantage** is then derived by centering the action value against the state baseline:

$$\widehat{A}(s,a) = \widehat{Q}(s,a) - \widehat{V}(s). \tag{7}$$

This allows us to identify actions that outperform the local average purely through inference-time retrieval, serving as a proxy for the advantage function $A^{\pi}(s,a)$.

To ensure numerical stability across different scales of rewards, we normalize the advantage:

$$\tilde{A}(s,a) = \frac{A(s,a)}{\max_{a' \in \mathcal{C}} |A(s,a')| + \epsilon}. \tag{8}$$

4.3. Policy Update

Given these value estimates, we formally derive the policy update as a constrained optimization problem. We seek an optimal policy π^* that maximizes the expected advantage while minimizing the KL-divergence from the frozen reference policy π_θ to preserve linguistic coherence. The closed-form solution to this optimization objective is:

$$\pi^*(a|s) \propto \pi_\theta(a|s) \exp\left(\beta A(s,a)\right) \tag{9}$$

where β is the temperature parameter that controls the strength of the constraint.

To implement this optimal policy without gradient updates, we map Eq. (9) directly to the logit space. Let $z(s,a)$ be the logits of the base LLM such that $\pi_\theta(a|s) = \text{Softmax}(z(s,a))$. Taking the logarithm of Eq. (9) yields an additive update rule:

$$z'(s,a) = z(s,a) + \beta \cdot A(s,a) \tag{10}$$

By applying the Softmax function to the updated logits $z'(s,a)$, we recover the optimal policy distribution π^*.

4.4. Theoretical Analysis

We provide theoretical justification for JitRL through three progressive steps. First, we prove our logit update is the exact closed-form solution for KL-constrained optimization (Theorem 4.1). Next, we demonstrate that our value estimates converge to the true values (Theorem 4.2), ensuring the overall policy update consistently converges to the optimal policy (Theorem 4.3)

Optimality of Logit Update. We formulate the inference-time adjustment as a constrained optimization problem. Our goal is to find a policy π' that effectively utilizes the retrieved advantage information while preserving the linguistic capabilities of the pre-trained model.

Theorem 4.1 (Optimality of Policy Update)**.** *Let π_θ be the reference policy with logits $z(s,a)$. Consider the problem of finding the optimal policy π^* that maximizes the expected advantage subject to a KL-divergence penalty:*

$$\pi^* = \arg\max_{\pi'} \left(\mathbb{E}_{a \sim \pi'}[A(s,a)] - \frac{1}{\beta} D_{KL}(\pi' || \pi_\theta) \right), \tag{11}$$

where β is a temperature hyperparameter. The solution to this optimization problem is given by the additive logit update:

$$z'(s,a) = z(s,a) + \beta \cdot A(s,a) \tag{12}$$

where $z'(s,a)$ denotes the logits of the optimal policy π^*.

Consistency of Value and Advantage Estimates. Next, we demonstrate that the estimators $\widehat{V}_t$, $\widehat{Q}_t$, and $\widehat{A}_t$ converge in probability to their true counterparts V^{π_t}, Q^{π_t}, and A^{π_t}, respectively, even under the non-stationary policy setting $(\pi_t)_{t \geq 1}$.

Theorem 4.2 (Consistency of $\widehat{V}_t$, $\widehat{Q}_t$, and $\widehat{A}_t$)**.** *Under the assumptions given in Appendix C, for any fixed query state s and action a at time t, as $t \to \infty$,*

$$\widehat{V}_t(s) \xrightarrow{p} V^{\pi_t}(s),$$
$$\widehat{Q}_t(s,a) \xrightarrow{p} Q^{\pi_t}(s,a),$$
$$\widehat{A}_t(s,a) \xrightarrow{p} A^{\pi_t}(s,a).$$

Consistency of Policy Update. Finally, combining the previous two theorems, we conclude that our policy update converges to the KL-regularized optimal policy induced by the *true* advantages A^{π_t}:

Theorem 4.3 (Consistency of Policy Update)**.** *Fix a state s, and define*

$$\pi_t^*(a \mid s) \propto \pi_\theta(a \mid s) \exp\big(\beta A^{\pi_t}(s,a)\big),$$
$$\widehat{\pi}_t(a \mid s) \propto \pi_\theta(a \mid s) \exp\big(\beta \widehat{A}_t(s,a)\big).$$

Under the assumptions of Theorem 4.2, as $t \to \infty$,

$$\widehat{\pi}_t(\cdot \mid s) \xrightarrow{p} \pi_t^*(\cdot \mid s),$$

for any finite candidate action set (or more generally, under uniform convergence of $\widehat{A}_t(s,\cdot)$ on the candidate set).

See Appendix B, C, and D for the detailed statements and proofs.

5. Experiments

In this section, we design our experiments to answer the following four research questions:

- **RQ1:** How does JitRL compare against state-of-the-art training-free baselines and weight-update methods?
- **RQ2:** Does JitRL possess generalization capabilities across unseen tasks and different model backbones?
- **RQ3:** How does JitRL qualitatively correct the agent's decision-making?
- **RQ4:** How do key components impact JitRL's performance?

5.1. Experimental Setup

Benchmarks. We evaluate our method on two distinct environments to demonstrate versatility:

- **WebArena** (Zhou et al., 2024): A realistic web environment comprising multiple functional websites. It challenges agents to navigate complex DOM trees and execute sequential actions to fulfill user instructions.
- **Jericho** (Hausknecht et al., 2020): A benchmark suite for interactive fiction games where agents interact purely via textual commands. We evaluate performance on three representative games: Library, Zork 1, and Zork 3.

Baselines. We compare JitRL against a comprehensive set of methods, which fall into two paradigms: training-free and weight-update.

- **Training-Free Methods.** We evaluate five inference-time strategies: (1) **Static**: A non-learning agent with fixed configuration. (2) **Memory**: An in-context learning baseline that fills the context window with full transcripts of past episodes, utilizing a First-In-First-Out (FIFO) strategy to handle token limits. (3) **Reflexion** (Shinn et al., 2023): A prompt-based method where the agent generates structured textual self-reflections after each episode to guide subsequent attempts. (4) **AWM** (Wang et al., 2025): An episodic memory approach that extracts and persistently stores reusable workflows derived from successful trajectories. (5) **EvoTest** (He et al., 2025): An evolutionary optimization framework that iteratively rewrites prompts, updates memory with state-action pairs, and dynamically tunes hyperparameters.
- **Weight-Update Methods**. We compare against leading training approaches. For WebArena, we directly utilize the pre-trained checkpoints provided by **SFT** and **WebRL** (Qi et al., 2025), both built on `Llama-3.1-70B-Instruct`. For Jericho, we employ the task-specific `Qwen3-32B` checkpoints, which are trained via **GRPO** (Shao et al., 2024) for each individual game.

Implementation Details. We specify the key implementation choices regarding the logit update mechanism, action candidate construction, and state similarity computation.

Logit Extraction Variants. We implement two variants to derive logits for the logit-based policy update. This design specifically addresses the constraint that many black-box models do not expose access to raw log-probabilities:

- **Token-level Logit**: We prompt the generator to select an action from k candidates by outputting a corresponding index token (e.g., "1", "2"). We then extract the log-probabilities of these tokens as logits
- **Verbalized Logit**: For models that do not expose log-probabilities, we prompt the LLM to explicitly output a confidence score (0–100) for each candidate action as the logit.

Augmented Candidate Set. To prevent the model from overlooking historically effective actions, we construct an augmented candidate set $\mathcal{C}$. This set merges the LLM's top-k predicted actions ($\mathcal{C}_{\text{LLM}}$) with the unique actions found in the retrieved neighborhood $\mathcal{N}(s)$:

$$\mathcal{C} \leftarrow \mathcal{C}_{LLM} \cup \{a_i : (s_i, a_i, G_i) \in \mathcal{N}(s)\}. \tag{13}$$

For any retrieved action $a \notin \mathcal{C}_{\text{LLM}}$, we initialize its base logit to a neutral value $z(s, a) = 0$.

Retrieval Similarity Metric. We use text-based state representations. To quantify the similarity between the current state s and a memory state s_i, we tokenize both representations and compute the **Jaccard Similarity**:

$$J(s, s_i) = \frac{|T(s) \cap T(s_i)|}{|T(s) \cup T(s_i)|}, \tag{14}$$

where $T(\cdot)$ denotes the set of tokens derived from a state.

More implementation details, such as state representation, memory retrieval, and action handling, can be found in Appendix E. Prompt templates can be found in Appendix F, hyperparameters can be found in Appendix G.

5.2. RQ1: Main Performance Comparison

To address RQ1, we employ a multi-trial sequential testing protocol. Instead of evaluating each task only once, the agent executes L consecutive episodes for each task i. This setup is designed to simulate a realistic deployment scenario where an agent must improve its performance on-the-fly through repeated interactions.

5.2.1. Results on WebArena

Metrics. We run each task for $L = 5$ times and evaluate performance primarily using the Success Rate (SR).

Table 1. Main results on WebArena. We report average (**Avg**) and final (**Final**) success rate (%) for each domain. The gap between Final and Avg reflects learning efficiency.

Method	Admin		GitLab		Map		Reddit		Shopping		Average	
	Avg	Final	Avg	Final	Avg	Final	Avg	Final	Avg	Final	Avg	Final
Static	39.46	39.67	38.92	38.24	30.62	31.25	39.60	42.86	24.06	25.00	35.63	36.30
Memory	47.91	47.80	39.31	40.20	30.47	32.81	53.02	55.04	30.16	33.16	41.36	43.00
Reflexion	48.46	50.55	38.43	40.69	29.38	27.34	54.88	56.59	30.83	31.25	41.08	42.12
AWM	49.46	51.09	37.94	40.20	34.38	32.81	55.66	58.14	23.12	22.40	39.37	40.32
EvoTest	44.51	47.80	37.94	39.22	33.59	41.41	51.78	54.26	26.67	29.17	39.24	42.49
JitRL	**52.31**	**56.59**	**40.78**	**45.00**	**37.66**	**42.19**	**57.64**	**61.98**	**41.67**	**45.83**	**46.98**	**51.35**

Table 2. Comparison of **Final** success rate (%) with weight-update methods on **WebArena-Lite**, the held-out test set. The remainder of WebArena was used to train WebRL.

Method	Admin	GitLab	Map	Reddit	Shopping	Avg
SFT	20.0	20.0	26.7	52.6	13.3	23.0
WebRL	54.3	50.0	40.0	78.9	44.4	49.1
JitRL	**65.71**	**56.67**	**53.57**	**78.95**	**53.33**	**60.0**

Table 3. Results on Jericho games. We report: **Avg**: average score across 50 episodes; **Final**: final episode score.

Method	Library		Zork1		Zork3	
	Avg	Final	Avg	Final	Avg	Final
Static	10.0	10	8.5	10	0.16	0
Memory	13.2	14	22.9	25	0.98	1
Reflexion	15.4	18	26.1	35	1.36	1
AWM	12.7	10	38.8	44	1.92	2
EvoTest	21.5	26	46.8	54	2.56	4
GRPO	13.6	11	16.2	10	1.10	2
JitRL	**25.9**	**30**	**53.0**	**69**	**3.12**	**5**

Let $y_{i,j} \in \{0, 1\}$ denote the binary completion status (success=1, failure=0) of task i at episode j. We report two aggregated metrics: (1) **Average Success Rate (Avg)**, calculated as $\frac{1}{|\mathcal{T}|\cdot 5} \sum_{i=1}^{|\mathcal{T}|} \sum_{j=1}^{5} y_{i,j}$, which averages performance across all attempts to reflect the overall learning efficiency; and (2) **Final Success Rate (Final)**, computed as $\frac{1}{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{T}|} y_{i,5}$, which measures the success rate of the final episode to indicate the agent's converged capability. Notably, a larger differential between Final and Avg indicates a steeper learning curve, demonstrating that the agent successfully leverages historical memory to improve its policy over time.

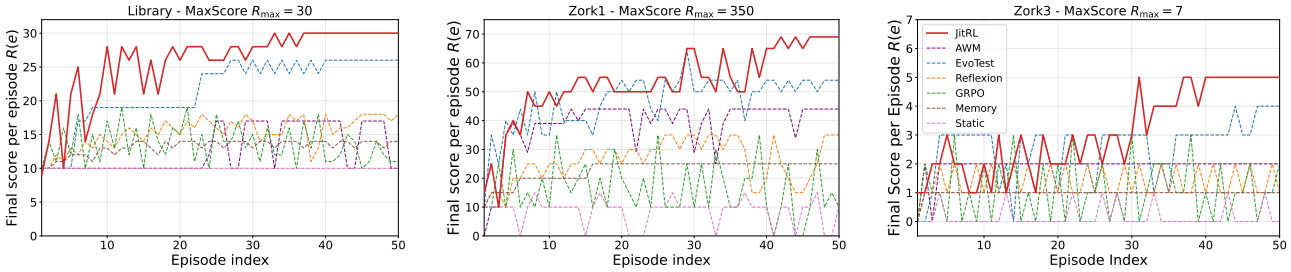
Table 1 presents the performance comparison of JitRL and training-free baselines on WebArena. JitRL outperforms all training-free baselines in both cumulative (Avg) and converged (Final) success rates. Moreover, the significant gap between the Avg and Final highlights JitRL's robust learning capability. Performance gains are most pronounced in structured domains like Shopping (+73.2% over Static) due to high trajectory reusability. In contrast, Reflexion occasionally suffers from "reflection noise" in sites like Map, where misleading feedback degrades performance.

Table 2 compares JitRL with weight-update methods (SFT and WebRL) on the held-out WebArena-Lite subset, as the remaining WebArena tasks were utilized for training WebRL. JitRL attains SOTA performance purely through inference-time optimization. This demonstrates that JitRL serves as a highly efficient alternative to computationally intensive training methods.

5.2.2. Results on Jericho

Metrics. For Jericho, we evaluate the **Game Score**. Let $y_{i,j}$ represent the score obtained in game i during episode j. Similar to WebArena, we report the **Average Score (Avg)** over 50 episodes to capture the learning stability, and the **Final Score (Final)** to assess the performance at the final episode.

Table 3 presents the quantitative comparison on Jericho. JitRL achieves the highest scores across all three games, significantly outperforming baselines. The learning curves for each episode are visualized in Figure 3, which reveal several key insights: (1) **JitRL exhibits rapid initial learning**, achieving competitive performance within the first 10-15 episodes; (2) **the performance gap between JitRL and baselines widens as episodes progress**, indicating effective experience accumulation; and (3) **JitRL shows reduced variance in later stages**, implying stable policy convergence. In contrast, GRPO exhibits high variance throughout, suggesting that gradient-based updates struggle with the sparse reward signals. Memory-based approaches (Memory, AWM) plateau early, because they over-rely on established memory patterns, which inhibit further exploration and cause the agents to converge to suboptimal policies.



(a) Library (b) Zork1 (c) Zork3

Figure 3. Learning curves on Jericho games. JitRL shows consistent improvement across episodes.

Table 4. Comparison of average (Avg) and **Final** success rate between JitRL and baseline methods across different backbone LLMs.

Method	Admin		Reddit	
	Avg	Final	Avg	Final
Gemini-2.5-flash				
Static	39.46	39.67	39.60	42.86
Memory	47.91	47.80	53.02	55.04
Reflexion	48.46	50.55	54.88	56.59
AWM	49.46	51.09	55.66	58.14
EvoTest	44.51	47.80	51.78	54.26
JitRL (Ours)	**52.31**	**56.59**	**57.64**	**61.98**
GPT-5-mini				
Static	37.07	37.50	47.29	46.51
Memory	43.04	45.65	50.54	51.94
Reflexion	40.43	41.30	49.15	49.61
AWM	40.00	40.22	50.23	51.94
EvoTest	42.93	43.48	51.94	53.49
JitRL (Ours)	**48.04**	**51.63**	**54.26**	**57.36**
DeepSeek-V3.2				
Static	32.07	32.61	42.02	43.41
Memory	37.07	45.65	51.47	56.28
Reflexion	38.04	40.22	**55.04**	56.59
AWM	42.07	44.02	52.71	55.04
EvoTest	44.35	45.65	50.85	53.49
JitRL (Ours)	**50.65**	**54.35**	54.42	**61.24**

5.3. RQ2: Generalization Capabilities

To answer RQ2, we assess the universality of our framework. This section evaluates JitRL's performance across different LLM backbones and investigates its zero-shot generalization on unseen tasks where no direct historical memory exists.

Generalization Across Backbones. We further evaluate JitRL on GPT-5-mini and DeepSeek-V3.2 (Sands et al., 2025) to assess its generalizability against other baselines. As shown in Table 4, JitRL achieves state-of-the-art performance in most cases. This indicates that our logit-update mechanism is model-agnostic and robustly enhances the capabilities of efficient models without parameter tuning.

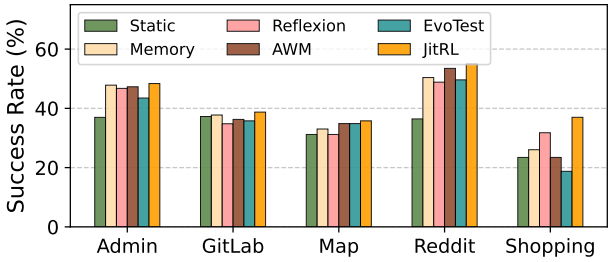


Figure 4. Generalization to Unseen Tasks. We report the average success rate (%) where the agent is restricted to retrieving memories exclusively from disjoint tasks.

Table 5. Distribution (%) of Cross-task Memory Usage across Sites

Admin	GitLab	Map	Reddit	Shopping	Avg
40.54	38.47	37.64	56.31	62.19	47.03

Cross Task Generalization. We assess cross-task generalization on WebArena by simulating a cold-start setting, where the agent utilizes memories solely from disjoint tasks. As shown in Figure 4, JitRL consistently outperforms baselines, indicating effective transfer of abstract procedural knowledge rather than specific solutions.

Additionally, we observed the retrieval patterns when both cross-task memory and cross-episode memory are enabled. As shown in Table 5, cross-task memory accounts for nearly 50% of the retrieved context, indicating a substantial reliance on this mechanism alongside cross-episode memory. Furthermore, websites that trigger a higher frequency of cross-task memory retrieval also demonstrate superior cross-task generalization performance compared to Static.

5.4. Qualitative Analysis: Correcting Semantic Priors

To provide an intuitive understanding of JitRL's mechanism, we examine representative cases where the retrieved memory successfully overrides the base model's erroneous intuition. As shown in Table 6, regarding **Site Functionality**, the agent learns to ignore intuitive but incorrect links

Table 6. **Qualitative Analysis of Policy Improvement.** We select three representative cases showing how JitRL modifies decision-making. **Base** and **JitRL** denote the logits before and after the memory-based update. Compared to **Base**, **JitRL** decreases the logits for **incorrect** options and increases them for **correct** options. In this way, JitRL corrects semantic priors and optimizes efficiency.

Task	Candidate Action	Base	JitRL	Mechanism Explanation
Find customer reviews	**click(CATALOG)**	**0.90**	0.40	While "Catalog" appears semantically intuitive, memory corrects this prior: reviews are located under "Marketing".
	click(MARKETING)	0.70	**1.40**	
Find latest posts in specific subreddit	**fill(Search, "..")**	**0.95**	0.45	Global search often yields noisy results. Memory steers the agent toward "Forums" for a deterministic and accurate path.
	click(Forums)	0.80	**1.80**	
Access product inventory	**click(Products)**	**0.95**	-0.28	Clicking loads a generic page. Memory identifies "hover" as an efficient shortcut to instantly reveal subcategories.
	hover(Products)	0.40	**0.90**	

Table 7. Ablation: Logit update vs. prompting on WebArena.

Method	Admin	Reddit
JitRL (Prompt Update)	49.46	53.02
JitRL (Logit Update)	**52.31**	**57.64**

(e.g., Catalog") and instead navigates to the correct section (Marketing"). Similarly, for **Navigation Precision**, memory steers the agent away from noisy global searches toward deterministic navigation links. Additionally, in terms of **UI Mechanics**, JitRL optimizes interaction efficiency, identifying shortcuts such as prioritizing hover actions to directly reveal fine-grained subcategories rather than clicking into broad category pages. Additional case studies on Jericho can be found in Appendix H.

5.5. RQ4: Ablation Studies

Finally, we address RQ4 by analyzing the sensitivity of JitRL to key components.

Impact of Logit Update. To investigate whether the performance gain stems from the logit update mechanism or just the retrieved information, we compare JitRL's **Logit Update** against **Prompt Update**, which uses the exact same memory retrieval but appends them to the prompt instead of modifying logits. Table 7 shows that direct Logit Update outperforms Prompt Update on Admin and Reddit websites. We attribute this to the limitations of prompting: as context length grows, LLMs often struggle to attend to retrieved cues or fail to strictly follow instructions. In contrast, our logit update directly modulates the output distribution, ensuring the agent effectively exploits historical advantages.

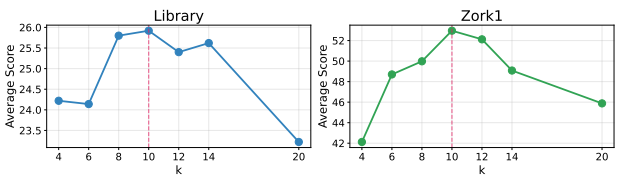


Figure 5. Impact of Retrieval Neighbor Count (k).

Impact of Retrieval Neighbor Count. We evaluate the impact of retrieval neighbor count (k) on Library and Zork1. As shown in Figure 5, the agent achieves robust performance with k ranging from 8 to 14. Outside this range, performance degrades due to the trade-off between context sufficiency and noise: a small k limits the historical evidence needed for stable value estimation, leading to high variance, while a large k introduces excessive noise that hinders efficient exploration and delays convergence.

More ablation studies such as the impact of state representation can be found in Appendix I.

5.6. Training Efficiency Analysis

We compare the training time and estimated monetary costs of JitRL against all baselines. As shown in Table 8, weight-update method, WebRL, incur massive overheads; for these methods, costs are estimated based on training expenses on NVIDIA H100 GPUs. In contrast, JitRL aligns with other training-free methods (e.g., Static, Memory, Reflexion, AWM, EvoTest) whose costs are calculated based on API usage prices. Despite its lower cost, JitRL achieves superior performance as demonstrated in our main experiments.

Table 8. Comparison of training time cost.

Metric	Static	Memory	Reflexion	AWM	EvoTest	WebRL	JitRL
Cost	$200	$230	$220	$250	$220	~$15000	$290

6. Conclusion

In this work, we propose Just-In-Time Reinforcement Learning (JitRL), a training-free framework that enables frozen LLMs to continuously adapt at test time by directly optimizing logits. We theoretically prove that our update rule is the optimal closed-form solution for RL policy optimization. Experimental results show that JitRL not only outperforms existing training-free methods but also outperforms expensive weight-update baselines, while reducing monetary costs by over 50×, offering a scalable and efficient path for agentic continuous learning.

References

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002. doi: 10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Chang, B., Sun, X., Li, L., and Sui, Z. A survey on in-context learning. In Al-Onaizan, Y., Bansal, M., and Chen, Y. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 1107–1128. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.64. URL https://doi.org/10.18653/v1/2024.emnlp-main.64.

Fang, J., Peng, Y., Zhang, X., Wang, Y., Yi, X., Zhang, G., Xu, Y., Wu, B., Liu, S., Li, Z., Ren, Z., Aletras, N., Wang, X., Zhou, H., and Meng, Z. A comprehensive survey of self-evolving AI agents: A new paradigm bridging foundation models and lifelong agentic systems. *CoRR*, abs/2508.07407, 2025. doi: 10.48550/ARXIV.2508.07407. URL https://doi.org/10.48550/arXiv.2508.07407.

Gao, H., Geng, J., Hua, W., Hu, M., Juan, X., Liu, H., Liu, S., Qiu, J., Qi, X., Wu, Y., Wang, H., Xiao, H., Zhou, Y., Zhang, S., Zhang, J., Xiang, J., Fang, Y., Zhao, Q., Liu, D., Ren, Q., Qian, C., Wang, Z., Hu, M., Wang, H., Wu, Q., Ji, H., and Wang, M. A survey of self-evolving agents: On path to artificial super intelligence. *CoRR*, abs/2507.21046, 2025. doi: 10.48550/ARXIV.2507.21046. URL https://doi.org/10.48550/arXiv.2507.21046.

Hausknecht, M. J., Ammanabrolu, P., Côté, M., and Yuan, X. Interactive fiction games: A colossal adventure. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7903–7910. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6297. URL https://doi.org/10.1609/aaai.v34i05.6297.

He, Y., Liu, J., Liu, Y., Li, Y., Cao, T., Hu, Z., Xu, X., and Hooi, B. Evotest: Evolutionary test-time learning for self-improving agentic systems. *CoRR*, abs/2510.13220, 2025. doi: 10.48550/ARXIV.2510.13220. URL https://doi.org/10.48550/arXiv.2510.13220.

Hendrycks, D., Song, D., Szegedy, C., Lee, H., Gal, Y., Brynjolfsson, E., Li, S., Zou, A., Levine, L., Han, B., et al. A definition of agi. *arXiv preprint arXiv:2510.18212*, 2025.

Jin, B., Zeng, H., Yue, Z., Wang, D., Zamani, H., and Han, J. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516, 2025. doi: 10.48550/ARXIV.2503.09516. URL https://doi.org/10.48550/arXiv.2503.09516.

Li, H., Ding, L., Fang, M., and Tao, D. Revisiting catastrophic forgetting in large language model tuning. In Al-Onaizan, Y., Bansal, M., and Chen, Y. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pp. 4297–4308. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-EMNLP.249. URL https://doi.org/10.18653/v1/2024.findings-emnlp.249.

Li, X., Zou, H., and Liu, P. Torl: Scaling tool-integrated RL. *CoRR*, abs/2503.23383, 2025. doi: 10.48550/ARXIV.2503.23383. URL https://doi.org/10.48550/arXiv.2503.23383.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Trans. Assoc. Comput. Linguistics*, 12:157–173, 2024. doi: 10.1162/TACL_A_00638. URL https://doi.org/10.1162/tacl_a_00638.

Packer, C., Fang, V., Patil, S., Lin, K., Wooders, S., and Gonzalez, J. Memgpt: Towards llms as operating systems. 2023.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Qi, Z., Liu, X., Iong, I. L., Lai, H., Sun, X., Sun, J., Yang, X., Yang, Y., Yao, S., Xu, W., Tang, J., and Dong, Y. Webrl: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=oVKEAFjEqv.

Qian, C., Acikgoz, E. C., He, Q., Wang, H., Chen, X., Hakkani-Tür, D., Tur, G., and Ji, H. Toolrl: Reward is all tool learning needs. *CoRR*, abs/2504.13958, 2025. doi: 10.48550/ARXIV.2504.13958. URL https://doi.org/10.48550/arXiv.2504.13958.

Sands, B., Wang, Y., Xu, C., Zhou, Y., Wei, L., and Chandra, R. An evaluation of llms for generating movie reviews: Gpt-4o, gemini-2.0 and deepseek-v3. *CoRR*, abs/2506.00312, 2025. doi: 10.48550/ARXIV.2506.00312. URL https://doi.org/10.48550/arXiv.2506.00312.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Song, H., Jiang, J., Min, Y., Chen, J., Chen, Z., Zhao, W. X., Fang, L., and Wen, J. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *CoRR*, abs/2503.05592, 2025. doi: 10.48550/ARXIV.2503.05592. URL https://doi.org/10.48550/arXiv.2503.05592.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In Solla, S., Leen, T., and Müller, K. (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Wang, Z. Z., Mao, J., Fried, D., and Neubig, G. Agent workflow memory. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=NTAhi2JEEE.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Xu, W., Liang, Z., Mei, K., Gao, H., Tan, J., and Zhang, Y. A-MEM: agentic memory for LLM agents. *CoRR*, abs/2502.12110, 2025. doi: 10.48550/ARXIV.2502.12110. URL https://doi.org/10.48550/arXiv.2502.12110.

Zhang, X., Jiang, Z., Meng, R., Leng, Y., Xiao, Z., Wang, Z. Z., Shang, Y., and Kong, D. Universal retrieval for multimodal trajectory modeling. In *ICML 2025 Workshop on Computer Use Agents*, 2025.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. Webarena: A realistic web environment for building autonomous agents. 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=oKn9c6ytLx.