Exploratory Memory-Augmented LLM Agent via Hybrid On- and Off-Policy Optimization

Anonymous authors
Paper under double-blind review

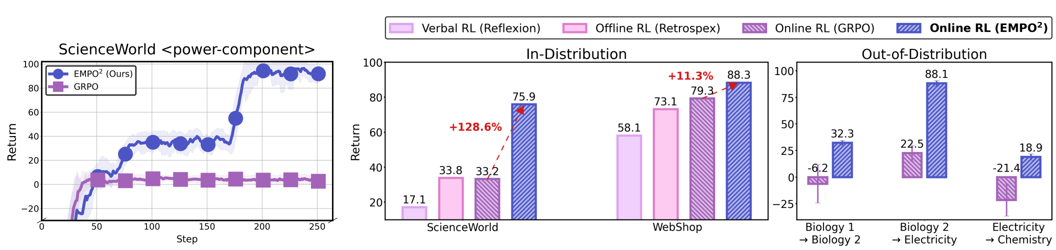
Abstract

Exploration remains the key bottleneck for large language model agents trained with reinforcement learning. While prior methods exploit pretrained knowledge, they fail in environments requiring the discovery of novel states. We propose Exploratory Memory-Augmented On- and Off-Policy Optimization (EMPO^2), a hybrid RL framework that leverages memory for exploration and combines on- and off-policy updates to make LLMs perform well with memory while also ensuring robustness without it. On ScienceWorld and WebShop, EMPO^2 achieves 128.6% and 11.3% improvements over GRPO, respectively. Moreover, in out-of-distribution tests, EMPO^2 demonstrates superior adaptability to new tasks, requiring only a few trials with memory and no parameter updates. These results highlight EMPO^2 as a promising framework for building more exploratory and generalizable LLM-based agents.

1 Introduction

Large Language Models (LLMs) have recently emerged as powerful agents capable of reasoning, planning, and interacting with external environments (Achiam et al., 2023; Park et al., 2023; Yao et al., 2023; Kim et al., 2025). When combined with reinforcement learning (RL), such agents can adapt their behavior based on experience and feedback, enabling them to go beyond static prompting or supervised fine-tuning (Guo et al., 2025; Tan et al., 2024). This paradigm has driven recent progress in areas such as interactive decision-making, tool use, and embodied AI (Feng et al., 2025b; Lu et al., 2025b; Feng et al., 2025a; Dong et al., 2025; Luo et al., 2025).

However, a key limitation of current LLM-based agents lies in their reliance on exploiting prior knowledge rather than engaging in systematic exploration. While RL frameworks emphasize balancing exploration and exploitation, many LLM-agent systems primarily leverage pretrained knowl-



(a) (b)

Figure 1: (a) **Comparison of the learning curves of GRPO and EMPO^2 (ours)** on the ScienceWorld `power-component` task. While GRPO converges to suboptimal performance, EMPO^2 continues to improve and accomplish the task. (b) **Comparison of EMPO^2 and other baselines in in-distribution (ID) and out-of-distribution (OOD) settings** on and WebShop. In ID experiments, it adapts well to familiar environments, achieving **128.6% on ScienceWorld** and **11.3% on Webshop** improvements over GRPO. In OOD experiments, it also shows strong performance with few trials and no weight updates, indicating effective use of memory to explore unfamiliar environments. Full results are in Tables 1, 2, and Figure 8.

edge and conduct only limited search within familiar distributions. As a result, these agents often struggle in environments where progress depends on discovering novel states or actively acquiring new information, rather than reusing what is already known.

To address this challenge, recent research has incorporated external memory modules into LLMs as a form of long-term memory. This enables models to leverage past experiences to correct failed attempts, thereby improving decision-making in subsequent trials without requiring parameter updates (Shinn et al., 2023; Zhang et al., 2023). However, as noted in Zhang et al. (2023), the performance of such methods tends to saturate quickly, since collecting experiences with static parameters cannot fully capture the diversity needed for continuous improvement.

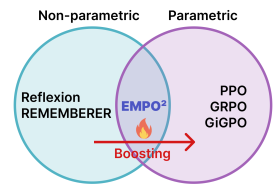


Figure 2: Non-parametric updates can encourage exploration, bootstrapping parametric updates.

In this work, we present a unified framework that enables LLM agents to learn more effectively through broader exploration by jointly updating their **parametric policy parameters** with RL and their **non-parametric memory module** through interaction. Crucially, the non-parametric updates not only complement but also enhance the efficiency of parametric learning, thereby enabling more effective exploration and adaptation. This dual-update paradigm serves as a bridge between parameter-level optimization and memory-augmented reasoning. While memory is utilized during learning, moving toward more generalizable intelligence requires reducing dependence on external memory and instead embedding its benefits directly into the model's parameters. To this end, we propose **E**xploratory **M**emory-Augmented **On-** and **Off-P**olicy **O**ptimization (EMPO^2), a new hybrid RL algorithm that incorporates two modes in the rollout phase—depending on whether memory is used—and two modes in the update phase—on-policy and off-policy learning—thereby enabling agents to leverage memory when available while remaining robust in its absence.

In our experiments, we evaluate EMPO^2 on two widely used multi-step embodied reasoning environments that require exploration to solve complex tasks: ScienceWorld (Wang et al., 2022) and WebShop (Yao et al., 2022). We compare its performance against a range of non-parametric and parametric (offline and online) RL approaches. As summarized in Figure 1, EMPO^2 substantially outperforms prior algorithms, achieving a 128.6% improvement on ScienceWorld and an 11.3% improvement on WebShop over the strong online RL baseline GRPO. The training curve in Figure 1 (a) further shows that, unlike GRPO, which converges prematurely to a suboptimal solution, EMPO² leverages continuous exploration and successfully solves the task. Moreover, for the OOD experiments (Figure 1, rightmost), the model also achieves good scores with only a few trials and no weight updates, indicating that the updated model has acquired the ability to use memory to explore unseen or unfamiliar environments. These results highlight EMPO^2 as a promising direction for building more adaptive and generalizable embodied agents.

2 Preliminaries

Online RL consists of alternating between a rollout phase, in which trajectories are generated using the current policy π parameterized by θ, and an update phase, in which the policy is optimized based on those rollouts.

Policy Rollout. We consider a setting where, given a sampled task $u \sim p(\mathcal{U})$, an LLM agent solves the task through multi-step interactions with the environment. Starting from task u, the LLM π_θ generates the first natural-language action $a_1 \sim \pi_\theta(\cdot \mid u) \in \mathcal{A}$. Executing this action, the environment returns a reward r_1 and the next state s_1. At a general timestep t, conditioned on the current state s_t and the task u, the policy produces the next action $a_{t+1} \sim \pi_\theta(\cdot \mid s_t, u)$. This interaction loop continues until the task is completed or a maximum number of steps is reached. A rollout trajectory is thus defined as the sequence of states, actions, and rewards, $\tau = \big(u, a_1, r_1, s_1, a_2, r_2, \ldots, s_T\big)$.

Group Relative Policy Optimization. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) updates the policy by comparing multiple rollouts of the same task u, removing the need for the value function in PPO (Schulman et al., 2017). Given a task u, the policy π_θ generates N rollout trajectories $\{\tau^{(1)}, \ldots, \tau^{(N)}\}$. Each trajectory receives a return $\{R^{(1)}, \ldots, R^{(N)}\}$, defined as the sum of rewards along the trajectory: $R^{(i)} = \sum_{t=1}^{T} r_t^{(i)}$.. For each action $a_t^{(i)}$ taken in trajectory $\tau^{(i)}$, we define its relative advantage as: $A(a_t^{(i)}) = \frac{R^{(i)} - \frac{1}{N}\sum_{j=1}^{N} R^{(j)}}{\sigma(R)}$, where actions from trajectories with higher-than-average reward obtain positive advantage, while those from lower-performing ones obtain negative advantage. The GRPO loss is then:

$$\mathbb{E}_{\substack{u \sim p(\mathcal{U}) \\ \{\tau^{(i)}\}_{i=1}^{N} \sim \pi_{\theta_{\text{old}}}}} \left[\frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \min\Big(\rho_\theta(a_t^{(i)}) A(a_t^{(i)}), \text{clip}\big(\rho_\theta(a_t^{(i)}), 1-\epsilon, 1+\epsilon\big) A(a_t^{(i)}) \Big) \right] - \beta\, D_{\text{KL}}\big(\pi_\theta(\cdot|u) \,\|\, \pi_{\text{ref}}(\cdot|u)\big), \tag{1}$$

where $\rho_\theta(a_t^{(i)}) = \frac{\pi_\theta(a_t^{(i)}|s_t^{(i)},u)}{\pi_{\theta_{\text{old}}}(a_t^{(i)}|s_t^{(i)},u)}$, with $\beta \geq 0$ controlling the regularization strength toward a reference policy π_{ref}.

3 The Exploration Problem of LLM Agents

LLMs encode rich prior knowledge, but such priors often fail to reflect the actual rules or dynamics of a given environment. Blind reliance on these priors can lead to erroneous behaviors, making it necessary for agents to adapt through direct interaction and trial-and-error. A key requirement for such adaptation is **exploration**, which involves seeking information beyond pre-training, sometimes by taking atypical or counterintuitive actions. However, current LLM-based agents struggle with this (Qiao et al., 2024; Zhou et al., 2024), as it demands stepping outside the distribution of behaviors where the model feels most confident.

Consequently, many prior studies have sought to align agents with new environments through warm-start supervised fine-tuning (SFT) using numerous golden trajectories (Song et al., 2024; Qiao et al., 2024; Xiang et al., 2024), leveraging large-scale models such as GPT-4 (Tang et al., 2024; Lin et al., 2023), or employing human engineering or well-established simulation information (Choudhury & Sodhi, 2025). While these methods achieve strong results in constrained settings, their effectiveness is limited to cases where such external support is available, and they generalize poorly to unseen scenarios without it.

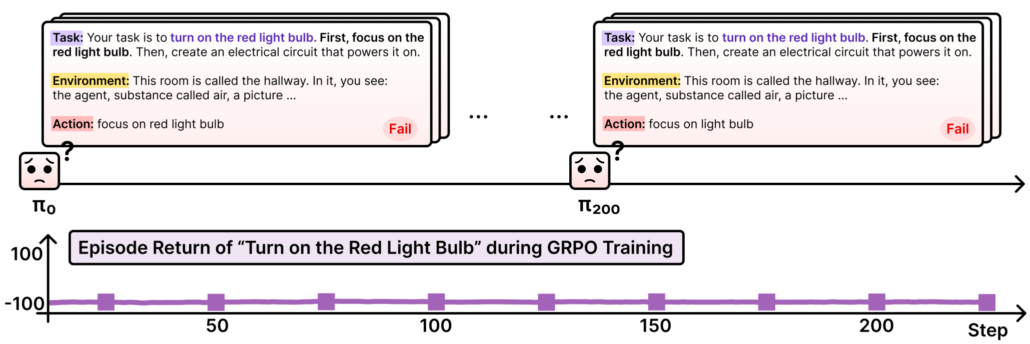


Figure 3: **When training LLM with GRPO in ScienceWorld, the agent struggles because of insufficient exploration.** For instance, in the task "turn on the red light bulb," the agent must first find the red light bulb before activating it. However, the agent fails to locate it and, as a result, cannot complete the task. Rather than analyzing the cause of failure and exploring alternative actions, the agent proceeds unchanged, so its score stagnates even as additional training steps are taken.

Therefore, we focus on how to efficiently train agents in online RL through trial and error, without any prior embedding of the environment's rules. The key challenge is that, without intrinsic exploration capability, online RL struggles to optimize effectively. As illustrated in Figure 3, in ScienceWorld (Wang et al., 2022) environment the agent is given the mission "turn on the red light bulb." The instructions specify that the agent should first focus on the light bulb and then build a circuit to activate it, based on the current room observation. However, since no red light bulb is present in the observation, the agent must search the environment to locate it. Instead, the agent follows the instruction literally, attempts to focus on the red light bulb, and fails because it does not exist in the room. Ideally, when an agent fails to reach its goal, it should analyze the reasons for failure and broaden its action space to discover successful strategies. Yet in representative online RL algorithms GRPO (Shao et al., 2024), prior trajectory rollouts provide no continuity beyond a scalar reward signal, thereby restricting exploration and ultimately limiting learning.

4 METHOD

In this section, we present Exploratory Memory-augmented On- and Off-Policy Optimization (EMPO^2), a novel algorithm aimed at tackling the exploration challenges in online RL. EMPO^2 operates in two modes for both rollout phase and update phase. During rollout, actions can be generated either through **(1) prompting without memory**, where no retrieved information is used, or **(2) memory-augmented prompting**, conditioned on tips retrieved from memory. In the update phase, rollouts with memory-augmented prompting are used in two ways: **(a) on-policy**, where tips are retained and the update is performed with the original prompt, and **(b) off-policy**, where tips are removed during update. Notably, tips are generated not by a separate model but by the policy π_θ itself, which is continually updated during training. The full algorithm is provided in Appendix A.

4.1 ADVANCING EXPLORATION WITH SELF-GENERATED MEMORY

A key component of EMPO^2 is its use of memory to maintain continuity across rollouts. Information obtained from an agent's interactions can be encoded into parameters through policy optimization, but it can also be recorded in an external memory that the agent continuously consults. Since our policy is initialized from a pretrained LLM with inherent summarization and reflection abilities, these abilities can be leveraged as auxiliary signals in addition to scalar rewards, thereby guiding exploration more effectively. To realize this, EMPO^2 integrates both ***parametric*** (parameter updates within the LLM) and ***non-parametric*** (external memory) updates, strengthening the linkage between rollouts and promoting exploration, with all data and guidance generated autonomously by the agent.

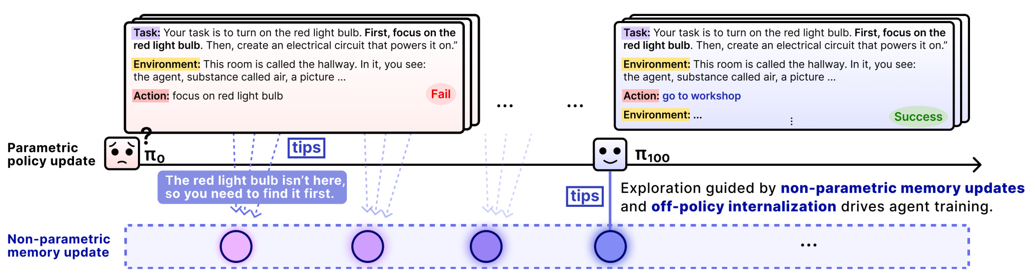


Figure 4: In EMPO^2, the current policy parameters π_θ are used to review past rollouts, with the resulting insights added to memory. This updated memory conditions subsequent rollouts and promotes exploration.

In the non-parametric updates, similar to Reflexion (Shinn et al., 2023), the agent reviews past rollouts, generates self-guidance tips, and stores them in memory. These tips help the agent avoid repeated mistakes and explore new strategies. Unlike Reflexion, focuses on iterative verbal guidance to achieve higher rewards in the next trial, our approach aims for these tips to lead to enhanced exploration that is ultimately consolidated through parametric updates.

Self-Generated Memory and Tips. We define a memory buffer $\mathcal{M} = \{\text{tip}_1, \text{tip}_2, \ldots\}$, which stores reflective tips generated by the policy π_θ during trajectory reflection. Formally, when an episode i of task u terminates at timestep t, the policy takes the final state s_t together with a tip-generation prompt as input and produces a tip, where $\text{tip}_i \sim \pi_\theta(s_t, u, \text{tip-generation prompt})$. A set of illustrative examples is provided below, while the tip-generation prompt is presented in Appendix B, and additional examples are included in Appendix E.1.

Examples of Generated Tips – ScienceWorld <power-component> task

- You moved between the kitchen and bathroom but did not find a green wire or a green light bulb to connect.
- You focused on the red light bulb but did not complete the task of turning on the red light bulb. You are in the hallway and need to find a way.
- The trajectory involves connecting the battery to the green wire terminals to power the green light bulb, but the connections to air and other objects are irrelevant.
- The circuit for the green light bulb was partially connected but still missing the battery connection; the task is not fully completed.

4.2 PARAMETERIZE NON-PARAMETRIC UPDATES VIA HYBRID POLICY OPTIMIZATION

Agents can use memory to improve exploration and learning efficiency, but the acquired knowledge needs be internalized into model parameters to enhance inherent capabilities. To this end, we propose two modes for the rollout and update phases, whose combinations yield three hybrid learning modes (Figure 5).

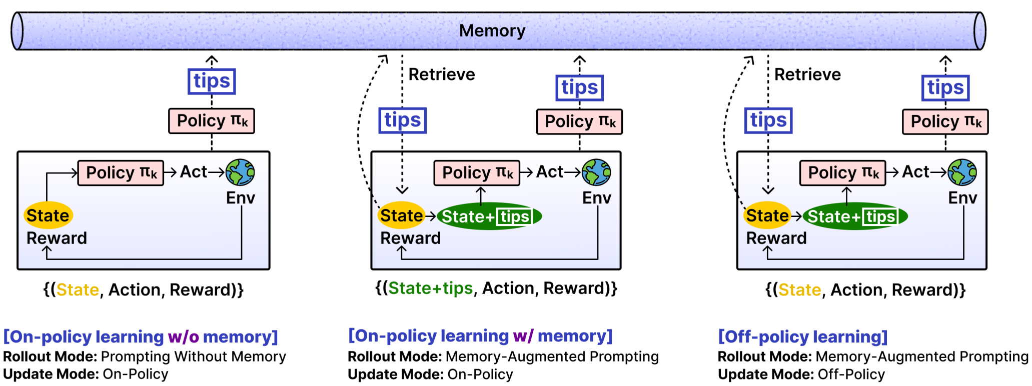


Figure 5: **EMPO^2 mode combinations.** By combining the two rollout modes and update modes, three EMPO mode configurations are possible: on-policy learning without memory, on-policy learning with memory and off-policy learning.

Rollout Modes. During rollouts, the agent samples between the two modes, selecting mode (1) with probability p and mode (2) with $1-p$. In our experiment, we set $p = 0.75$. The ablation study of p can be found in Appendix E.1.

(1) **Prompting Without Memory.** For each task u, at each timestep t, the policy π_θ generates actions conditioned only on the current state s_t and the task u: $a_{t+1} \sim \pi_\theta(\cdot \mid s_t, u)$.

(2) **Memory-Augmented Prompting.** For each task u, at each timestep t, a retrieval operator $\mathrm{Retr}(o_t; \mathcal{M}) \subseteq \mathcal{M}$ selects tips most relevant to the current state s_t, e.g., via similarity search in the embedding space. We denote the retrieved set as $\boxed{\text{tips}}_t$. In memory-augmented prompting, the policy conditions its action on both s_t and $\boxed{\text{tips}}_t$: $a_{t+1} \sim \pi_\theta\big(\cdot \,\big|\, s_t, u, \boxed{\text{tips}}_t\big)$. We limit the number of retrieved tips at 10.

Update Modes. Trajectories generated under rollout mode (1) are directly used for updates, whereas those generated under rollout mode (2) — memory-augmented prompting — follow one of two update modes chosen at random during the update phase. In our experiments, mode (a) is selected with probability $q = 1/3$, and mode (b) with probability $1-q$. The ablation study of q can be found in Appendix E.1.

(a) **On-Policy Updates.** On-policy update uses the same prompt as in the rollout, and $\rho_\theta(a_t^{(i)})$ in eq.1 becomes $\rho_\theta(a_t^{(i)}) = \frac{\pi_\theta(a_t^{(i)}|s_t^{(i)},u,\boxed{\text{tips}}_t)}{\pi_{\theta_{\text{old}}}(a_t^{(i)}|s_t^{(i)},u,\boxed{\text{tips}}_t)}$.

(b) **Off-Policy Updates.** In this mode, the stored log-probabilities $\ell_t^{\text{tips}} = \log \pi_\theta(a_t \mid s_t, u, \boxed{\text{tips}}_t)$ are replaced with the log-probabilities assigned by the same policy π_θ when conditioned only on (s_t, u), namely $\ell_t^{\text{no-tips}} = \log \pi_\theta(a_t \mid s_t, u)$. In this formulation, the advantage update is performed based on how natural the action appears under the distribution without tips.
This construction can be interpreted as a form of **reward-guided knowledge distillation**. Trajectories sampled under the tips-conditioned policy act as teacher demonstrations, while the student policy $\pi_\theta(\cdot \mid s, u)$ is updated to reproduce those trajectories in proportion to their advantage. High-reward trajectories ($\hat{A}_t > 0$) are reinforced, while low-reward trajectories ($\hat{A}_t < 0$) are suppressed, resulting in selective distillation that emphasizes beneficial behaviors. In this way, tips serve as an intermediate scaffolding mechanism that improves exploration and trajectory quality, while the reward signal ensures that only advantageous behaviors are ultimately retained. Consequently, the final policy learns to internalize the benefits of tip conditioning without requiring tips at inference time. Appendix B provides an illustrative breakdown and a summary table for the calculation of the importance sampling ratio.

Stabilizing Off-Policy Training. Off-policy training is prone to instability and may collapse (see Figure 6). In such cases, gradient normalization, entropy loss, KL loss, and policy gradient loss can all diverge to NaN. Prior work, Yang et al. (2025) shows that low-probability tokens destabilize training by amplifying gradient magnitudes through unbounded likelihood ratios. Motivated by this, we introduce a masking mechanism that suppresses the advantage term for tokens whose probability under π_θ falls below a threshold δ. Finally, the loss in Eq. 1 is modified as

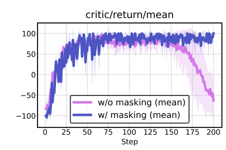


Figure 6: Masking tokens stabilizes training.

$$\mathbb{E}_{\substack{u \sim p(\mathcal{U}) \\ \{\tau^{(i)}\} \sim \pi_{\theta_{\text{old}}}}} \left[\frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \min\Big(\rho_\theta^{(i,t)} A(a_t^{(i)}),\ \text{clip}\big(\rho_\theta^{(i,t)}, 1-\epsilon, 1+\epsilon\big) A(a_t^{(i)})\Big) \cdot \mathbf{1}_{\pi_\theta(a_t^{(i)} \mid s_t^{(i)}, u) \geq \delta} \right.$$
$$\left. - \beta D_{\text{KL}}\big(\pi_\theta(\cdot|u) \,\|\, \pi_{\text{ref}}(\cdot|u)\big). \right] \tag{2}$$

Intrinsic Rewards for Exploration. To further encourage exploration, and inspired by prior work on exploration-targeted online RL (Burda et al., 2018b; Bellemare et al., 2016; Ecoffet et al., 2019), we introduce an intrinsic reward based on the novelty of the current state. A memory list stores distinct states, and for each new state we compute its cosine similarity with existing entries. If the similarity falls below a threshold, the state is added to memory and assigned a reward. The intrinsic reward is defined as $r_{\text{intrinsic}} = \frac{1}{n}$, where n denotes the number of similar past states. This mechanism encourages the agent to explore novel states even when no extrinsic reward is provided by the environment and maintains policy entropy, as shown in Figure 7.

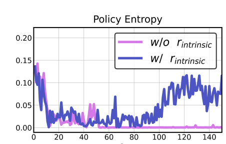


Figure 7: Policy entropy comparison with vs. without intrinsic rewards.

5 Related Work

LLM Agents in Multi-Step Embodied Tasks. LLM agents for multi-step embodied tasks have been studied under different paradigms. Data-driven approaches (Song et al., 2024; Xiong et al., 2024; Qiao et al., 2025; 2024; Tajwar et al., 2025) enhance decision-making through effective data collection methods and imitation learning. Model-based agents (Tang et al., 2024; Zhou et al., 2024) build world models, often by generating code with large closed-source systems such as GPT-4. Other methods (Lin et al., 2023; Choudhury & Sodhi, 2025) strengthen reasoning through model transitions or by leveraging privileged information provided by the simulation environment. In contrast, our approach reduces reliance on such external resources and emphasizes autonomous growth through the agent's own exploration and self-improvement.

Memory for LLM Agents. To enable progressive improvement from past experiences, Reflexion (Shinn et al., 2023) and REMEMBERER (Zhang et al., 2023) leverage external memory. Reflexion stores verbal reflections for later prompting, while REMEMBERER records observations, actions, rewards, and Q-values, retrieving similar cases as few-shot exemplars. These methods show that LLMs can improve without parameter updates. However, with fixed parameters, they cannot expand intrinsic knowledge, so adaptation remains short-term (Zhang et al., 2023), relying on external memory rather than achieving long-term evolution and generalization.

Learning by Knowledge Distillation Our hybrid off-policy update functions as reward-guided knowledge distillation during online training. Snell et al. (2022) introduced context distillation, where the model first solves tasks using a Teacher prompt (with instructions, examples, explanations, and scratch-pad reasoning) and then learns to produce the final answer from a minimal Student prompt via offline, SFT-based distillation. In contrast, we integrate knowledge distillation into online RL, leveraging online adaptability while enhancing exploration for more efficient training.

RL for LLM Agents. RL provides a robust framework for optimizing LLM parameters through observations and reward signals from environment interactions. Prior work, Retrospex (Xiang et al., 2024), showed that offline RL, which learns optimal policies from large logged datasets, can improve LLM agent performance. Recent studies focus on online RL (Shao et al., 2024; Feng et al., 2025b; Wang et al., 2025), where agents learn in real time. GiGPO (Feng et al., 2025b) advanced GRPO by grouping rollouts with similar observations, enabling finer credit assignment and stronger performance. Our work advances this online RL direction by integrating non-parametric memory updates into both on- and off-policy learning, yielding substantially higher sample efficiency.

Enhancing Exploration for Online RL. A central challenge in online RL is effective exploration. Classical methods such as count-based exploration (Bellemare et al., 2016) and Random Network Distillation (Burda et al., 2018b) use intrinsic rewards to encourage novelty. Go-Explore (Ecoffet et al., 2019) stores key states and re-explores from them, solving hard-exploration tasks like Atari games. Its LLM extension, Intelligent Go-Explore (Lu et al., 2025a), achieves strong results in environments such as TextWorld (Côté et al., 2018), but relies on large closed-source models and does not perform parameter updates. In our concurrent work, RLVMR (Zhang et al., 2025) employs warm-start SFT to elicit diverse reasoning types (planning, exploration, and reflection) and provides dense, process-level rewards for each reasoning type during online RL, enhancing exploration and credit assignment. Together, these studies underscore the importance of structured exploration for scaling RL to complex environments.

6 Experiments

To examine the effectiveness of EMPO^2, we conduct extensive experiments on two widely used LLM agent benchmarks: ScienceWorld (Wang et al., 2022) and WebShop (Yao et al., 2022) using Qwen2.5-7B-Instruct (Qwen et al., 2025) as the base model. The EMPO^2 performance we evaluate is the performance of the trained model without memory at test time.

6.1 ScienceWorld

ScienceWorld (Wang et al., 2022) is an interactive text-based benchmark in which an agent performs science experiments at the elementary school level. Successfully completing these experiments requires long-term multi-step planning, hypothesis testing, and interpretation of outcomes, as well as sufficient exploration to determine where the necessary tools are and what appropriate actions should be taken. ScienceWorld includes tasks from diverse topics and in our experiments, we cover 19 tasks spanning chemistry, classification, biology, electricity, and measurement.

Baselines. We compare EMPO^2 with several RL approaches. For non-parametric RL, Reflexion (Shinn et al., 2023) updates memory in a non-parametric manner by incorporating LLM reflections from previous trajectories and using them in the prompt for the subsequent trial. For offline RL, Retrospex (Xiang et al., 2024) leverages an SFT-trained model and uses a Q-function learned via Implicit Q-learning (Kostrikov et al., 2022) to dynamically rescore actions. The official Retrospex paper used the smaller Flan-T5-Large (Chung et al., 2024) (770M) and incorporated human-designed heuristics to assist the agent during evaluation. In contrast, to ensure consistency in our experimental setup, we standardize the base model of Retrospex to Qwen2.5-7B-Instruct and exclude these heuristics. Finally, for online RL, we include GRPO (Shao et al., 2024) as a representative baseline. Further details are provided in Appendix D.

Training Details. Our EMPO^2 implementation is based on verl (Sheng et al., 2024), one of the representative RL-for-LLM libraries. We extended GRPO in verl from a single-step setup to a multi-step setup and incorporated both a memory module and an off-policy loss calculation component. We use the same hyperparameter configuration for GRPO and EMPO^2. The prompt used is provided in Appendix B, and implementation details are given in Appendix D.2.

Table 1: **Comparison results of ScienceWorld.** Each task in ScienceWorld contains multiple variants. We use the first five variants for training and evaluate on the 20 unseen test variants. Bold shows the best performance per task, while red shading marks cases where parametric updates score lower than non-parametric updates.

Qwen2.5-7B-Instruct		Naive	Non-Parametric	Offline RL	Online RL	
Topic	Task		Reflexion	Retrospex	GRPO	EMPO^2
Chemistry	chemistry-mix	-42.0 ± 38.0	1.2 ± 0.7	20.8 ± 10.0	12.4 ± 3.5	$\mathbf{42.7\pm12.4}$
	chemistry-mix-paint-secondary-color	-33.0 ± 47.1	0.0 ± 0.0	27.8 ± 6.3	7.1 ± 2.8	$\mathbf{33.3\pm0.6}$
	chemistry-mix-paint-tertiary-color	-33.9 ± 44.3	36.9 ± 5.7	7.6 ± 4.2	$\mathbf{42.6\pm6.2}$	39.2 ± 8.7
Classification	find-animal	-58.2 ± 50.2	39.5 ± 5.8	25.9 ± 13.5	72.4 ± 6.8	$\mathbf{100.0\pm0.0}$
	find-living-thing	-65.1 ± 48.1	36.6 ± 6.1	20.6 ± 4.8	68.7 ± 7.1	$\mathbf{100.0\pm0.0}$
	find-non-living-thing	-35.9 ± 68.6	4.8 ± 2.0	89.1 ± 11.5	24.7 ± 6.4	$\mathbf{100.0\pm0.0}$
	find-plant	-47.1 ± 66.2	15.1 ± 3.8	23.0 ± 3.5	46.2 ± 7.9	$\mathbf{100.0\pm0.0}$
Biology1	identify-life-stages-1	-48.9 ± 65.4	9.2 ± 2.4	19.0 ± 25.7	17.9 ± 4.7	$\mathbf{36.2\pm11.2}$
	identify-life-stages-2	-50.7 ± 65.0	33.8 ± 5.5	11.0 ± 1.7	39.5 ± 6.0	$\mathbf{56.3\pm8.1}$
Biology2	lifespan-longest-lived	-51.8 ± 64.8	44.6 ± 6.5	55.0 ± 15.0	78.2 ± 7.3	$\mathbf{100.0\pm0.0}$
	lifespan-longest/shortest-lived	-56.2 ± 63.5	34.1 ± 5.1	38.0 ± 15.0	62.3 ± 6.9	$\mathbf{100.0\pm0.0}$
	life-span-shortest-lived	-56.8 ± 63.0	6.1 ± 1.9	67.0 ± 23.8	20.6 ± 4.4	$\mathbf{100.0\pm0.0}$
Electricity	power-component	-90.0 ± 39.4	6.3 ± 1.8	8.2 ± 2.4	15.1 ± 3.9	$\mathbf{94.3\pm3.6}$
	power-component-renewable-vs-nonrenewable-energy	-85.0 ± 49.8	11.7 ± 2.9	10.0 ± 3.2	24.6 ± 5.5	$\mathbf{92.6\pm0.9}$
	test-conductivity	-86.9 ± 42.4	13.2 ± 3.1	60.0 ± 0.0	27.8 ± 6.1	$\mathbf{89.5\pm3.2}$
	test-conductivity-of-unknown-sub	-81.7 ± 48.6	2.6 ± 1.0	65.5 ± 23.7	9.5 ± 3.4	$\mathbf{71.4\pm6.3}$
Measurement	measure-melting-point-known-sub	-97.5 ± 7.5	11.4 ± 3.0	26.5 ± 16.1	19.8 ± 5.0	$\mathbf{27.6\pm4.2}$
	use-thermometer	-83.7 ± 43.6	0.9 ± 0.4	32.5 ± 32.1	7.6 ± 2.5	$\mathbf{82.7\pm13.3}$
	Average	-61.3	17.1	33.8	33.2	**75.9**

Main Results. Table 1 presents the comparison results among baselines. In ScienceWorld, failed tasks lead to negative rewards, producing returns between -100 and 100. The baseline Qwen2.5-7B-Instruct obtains an average return of -61.3, which improves to 17.1 when non-parametric RL (Reflexion) is applied. Offline RL (Retrospex) produces substantial performance gains compared to them, but in some tasks underperforms compared to non-parametric RL (highlighted in red). Online RL with GRPO also achieves considerable improvements, and its average performance is comparable to that of offline RL. However, unlike offline RL, it never underperforms non-parametric RL, indicating that online RL generalizes better to unseen variants. Our EMPO^2 demonstrates substantially higher learning performance compared to all baselines. Among the tasks that initially started with negative rewards, seven reached the maximum score of 100. On average, EMPO^2 achieved more than twice the performance improvement over GRPO, demonstrating its effectiveness in greatly enhancing learning efficiency in online RL.

Adaptation in New Tasks with Memory Updates. An agent post-trained on a single task may exhibit limited ability to generalize to new scenarios. However, EMPO^2, which acquires the ability to explore by leveraging memory, demonstrates significantly stronger adaptability to novel situations compared to GRPO, which is trained without learning to utilize memory. Figure 8 illustrates how a model trained on one task adapts when memory is introduced in a new task. In particular, we demonstrate cases with varying levels of topic difference. For a relatively similar transition, we examine Biology 1 (*identify-life-stages-2*) → Biology 2 (*life-span-shortest-lived*). For a more distinct transition, we examine Biology 2 (*lifespan-longest-lived*) → Electricity (*test-conductivity*), and Electricity (*power-component*) → Chemistry (*chemistry-mix-paint-secondary-color*).

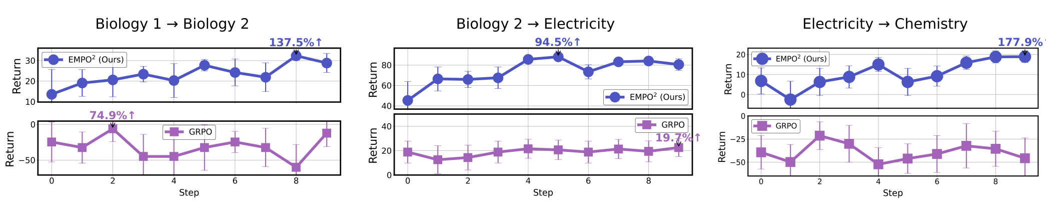


Figure 8: Comparison of GRPO and EMPO^2 adapting to new tasks. Step 0 has no memory, while later steps use accumulated memory as in EMPO^2 training.

As shown in Figure 8, without memory (step 0), EMPO^2 achieves stronger baseline performance on novel tasks than GRPO. When memory is enabled, EMPO^2 adapts rapidly, yielding an average improvement of 136% across three scenarios within 10 steps. GRPO, by contrast, demonstrates notable gains in some cases but exhibits greater variability and, in other instances, fails to adapt to unfamiliar tasks. In certain situations, its performance even falls below that of the Qwen2.5-7B-Instruct base model. Though these findings are preliminary, they indicate that EMPO^2 has strong potential as an RL framework for developing agents that are both more general and adaptable.

6.2 WebShop

WebShop (Yao et al., 2022) is an HTML-based online shopping environment where agents search, navigate, and purchase items according to user instructions. When the “buy” action is selected, a final reward is given based on how well the product’s attributes and price match the criteria.

Baselines. For the WebShop experiments, we use the same baselines as in the ScienceWorld experiments, with the addition of one more online RL baseline, GiGPO (Feng et al., 2025b), as GiGPO does not cover ScienceWorld but provides benchmarking results on WebShop. The scores of Naive, Reflexion, GRPO, and GiGPO are taken from Feng et al. (2025b), while Retrospex results are re-run using the official Retrospex code with the Qwen2.5-7B-Instruct model.

Training Details. The WebShop EMPO^2 implementation builds on the official GiGPO (Feng et al., 2025b) code with the same hyperparameters. Further details are provided in Appendix D.3.

Main Results. Table 2 presents the baseline comparison results on WebShop. Consistent with the findings in ScienceWorld, EMPO^2 once again delivers the strongest performance. Although offline RL, online GRPO, and GiGPO each outperform non-parametric RL, GiGPO further enhances GRPO by leveraging additional advantage estimation through grouping similar observations within rollout groups. Despite these gains, EMPO^2 surpasses all baselines, achieving both higher scores and success rates than GiGPO. Taken together, these results indicate that EMPO^2 consistently demonstrates superior performance in web-based environments due to its improved exploration.

Table 2: **Comparison results of WebShop.** Following Feng et al. (2025b), we average results over three random seeds and report both the mean score and the mean success rate (%). GiGPO$_{\text{w/ std}}$ denotes the use of the normalization factor $F_{\text{norm}} = \text{std}$, whereas GiGPO$_{\text{w/o std}}$ uses $F_{\text{norm}} = 1$, as specified in Feng et al. (2025b). The EMPO^2 performance we evaluate is the performance of the trained model without memory at test time.

Qwen2.5-7B-Instruct	Naive	Non-Parametric	Offline RL	Online RL			
		Reflexion	Retrospex	GRPO	GiGPO w/ std	GiGPO w/o std	**EMPO^2**
Score	26.4	58.1	73.1±4.1	79.3±2.8	84.4±2.9	86.2±2.6	**88.3±2.6**
Succ.	7.8	28.8	60.4±3.9	66.1±3.7	72.8±3.2	75.2±3.8	**76.9±4.1**

6.3 Ablation study on Mode Combinations

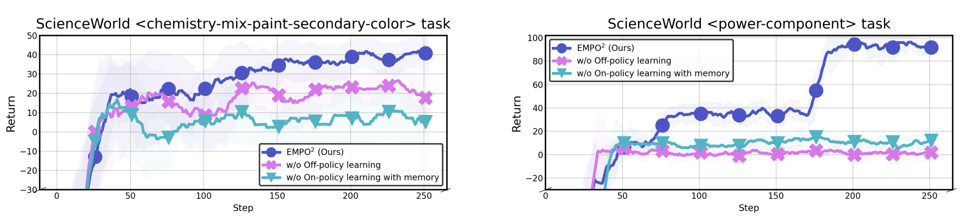


Figure 9: Comparison of training curves between EMPO^2 and variants that exclude either off-policy learning or on-policy learning with memory.

As shown in Figure 5, EMPO^2 incorporates three mode combinations: on-policy learning without memory, on-policy learning with memory, and off-policy learning, and we further analyze how leveraging each affects performance on two ScienceWorld tasks, where EMPO^2 shows significant improvements over GRPO. Figure 9 presents training curves comparing EMPO^2 with variants that exclude either off-policy or on-policy learning with memory. As shown in the graphs, removing either component results in suboptimal learning, indicating that a balanced integration of on-policy and off-policy updates is most effective for performance improvement. This highlights their complementary roles: on-policy updates contribute to stable learning, while off-policy updates enable reasoning as if guided by additional tips, and their combination yields both faster convergence and stronger final performance.

7 CONCLUSION

In this work, we propose EMPO^2, a novel RL method that enhances exploration in parametric RL by leveraging non-parametric memory updates. EMPO^2 integrates both on-policy and off-policy learning, thereby improving training efficiency and stability. Our experiments demonstrate that EMPO^2 achieves remarkable gains in training efficiency on ScienceWorld and WebShop, and further shows the ability to adapt rapidly to new domains in a few-shot manner by incorporating additional memory. An ablation study confirms the importance of the three distinct modes of EMPO^2.

While our study demonstrates the potential of EMPO^2 as a RL framework for general agents, our current implementation for memory employs a simple similarity-based search for memory retrieval. More advanced retrieval mechanisms may further enhance performance. Moreover, although our experiments primarily utilize Qwen2.5-7B-Instruct, extending EMPO^2 to a broader range of model families and sizes could yield deeper insights into its generality and robustness. In particular, scaling to larger models may further amplify the benefits of our approach. Beyond model scaling, applying EMPO^2 to new domains such as mathematics, coding, multi-hop question answering, and multi-modal RL represents an exciting and challenging direction for future research. In addition, exploring other off-policy techniques beyond importance sampling could be of interest to achieve more stable and efficient hybrid optimization.

ETHICS STATEMENT

This work evaluates EMPO^2 on ScienceWorld and WebShop, which are publicly available research benchmarks that do not include private data or sensitive information. We complied with dataset licenses and community standards for responsible use and citation, and no additional data collection or modification of the environments was performed.

Although our method exhibits strong adaptability in exploration and reasoning tasks, online RL systems may be misapplied in safety-critical real-world contexts. To reduce such risks, we confine our study to benchmark environments, and for real-world applications, responses generated by LLMs will require more careful scrutiny. We hope that future research will further address safety and broader societal impacts when extending embodied reasoning agents beyond simulation.

REPRODUCIBILITY STATEMENT

We provide detailed training information, including pseudocode in Appendix A, the hyperparameters used in our experiments, the hyperparameters for the baseline experiments, the GPU resources utilized, and code snippets for the additional components we implemented in Appendix D.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018a.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018b.

Sanjiban Choudhury and Paloma Sodhi. Better than your teacher: LLM agents that learn from privileged AI feedback. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=st7XqFgbAH`.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532, 2018.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *CoRR*, 2025a.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025b.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jeonghye Kim, Sojeong Rhee, Minbeom Kim, Dohyung Kim, Sangmook Lee, Youngchul Sung, and Kyomin Jung. Reflact: World-grounded decision making in llm agents via goal-state reflection. *arXiv preprint arXiv:2505.15182*, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=68n2s9ZJWF8`.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *Advances in Neural Information Processing Systems*, 36:23813–23825, 2023.

Cong Lu, Shengran Hu, and Jeff Clune. Intelligent go-explore: Standing on the shoulders of giant foundation models. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL `https://openreview.net/forum?id=apErWGzCAA`.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025b.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K Qiu, and Yuqing Yang. Agent lightning: Train any ai agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*, 2025.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agent planning with world knowledge model. *Advances in Neural Information Processing Systems*, 37:114843–114871, 2024.

Shuofei Qiao, Zhisong Qiu, Baochang Ren, Xiaobin Wang, Xiangyuan Ru, Ningyu Zhang, Xiang Chen, Yong Jiang, Pengjun Xie, Fei Huang, et al. Agentic knowledgeable self-awareness. *arXiv preprint arXiv:2504.03553*, 2025.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, 2024.

Fahim Tajwar, Yiding Jiang, Abitha Thankaraj, Sumaita Sadia Rahman, J Zico Kolter, Jeff Schneider, and Ruslan Salakhutdinov. Training a generally curious agent. *arXiv preprint arXiv:2502.17543*, 2025.

Weihao Tan, Wentao Zhang, Shanqi Liu, Longtao Zheng, Xinrun Wang, and Bo An. True knowledge comes from practice: Aligning large language models with embodied environments via reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=hILVmJ4Uvu`.

Hao Tang, Darren Key, and Kevin Ellis. Worldcoder, a model-based llm agent: Building world models by writing code and interacting with the environment. *Advances in Neural Information Processing Systems*, 37:70148–70212, 2024.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*, 2022.

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

Yufei Xiang, Yiqun Shen, Yeqin Zhang, and Nguyen Cam-Tu. Retrospex: Language agent meets offline reinforcement learning critic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4650–4666, 2024.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch every step! llm agent learning via iterative step-level process refinement. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1556–1572, 2024.

Zhihe Yang, Xufang Luo, Zilong Wang, Dongqi Han, Zhiyuan He, Dongsheng Li, and Yunjian Xu. Do not let low-probability tokens over-dominate in rl for llms. *arXiv preprint arXiv:2505.12929*, 2025.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

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

Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. Large language models are semi-parametric reinforcement learning agents. *Advances in Neural Information Processing Systems*, 36:78227–78239, 2023.

Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. Rlvmr: Reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents. *arXiv preprint arXiv:2507.22844*, 2025.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*, 2024.

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. Wall-e: World alignment by rule learning improves world model-based llm agents. *arXiv preprint arXiv:2410.07484*, 2024.

A PSEUDO CODE

Algorithm 1 presents the pseudocode of EMPO^2. Compared to the original GRPO algorithm, EMPO^2 introduces several new components: a memory buffer, and tip retrieval and addition, and two rollout modes and two update modes.

Algorithm 1 EMPO^2: Exploratory Memory-Augmented On- and Off-Policy Optimization

Inputs: Initial policy $\pi_{\theta_{\text{old}}}$, memory buffer $\mathcal{M}$, task distribution $p(\mathcal{U})$, group size N, batch size B, max episode length T
for each training iteration **do**
  {// Multi-step rollout}
  Sample B tasks $u \sim p(\mathcal{U})$ and initialize N identical environments (total $B \times N$)
  Sample $m_{\text{rollout}} \sim \{\text{Prompting Without Memory} : p,\ \text{Memory-Augmented Prompting} : 1-p\}$
  Initialize state $s_0^{(i)} \leftarrow u^{(i)}$ for $i = 0, \ldots, B \times N - 1$
  for $t = 0$ to $T-1$ **do**
    for $i = 0$ to $B \times N - 1$ **do**
      if $m_{\text{rollout}} =$ Memory-Augmented Prompting **then**
        $\boxed{\text{tips}}_t \leftarrow \text{Retr}(s_t^{(i)}; \mathcal{M})$
        Sample $a_t^{(i)} \sim \pi_\theta^{\text{old}}(\cdot \mid s_t^{(i)}, \boxed{\text{tips}}_t, u^{(i)})$
      else
        Sample $a_t^{(i)} \sim \pi_\theta^{\text{old}}(\cdot \mid s_t^{(i)}, u^{(i)})$
      end if
      Execute $a_t^{(i)}$, observe $r_t^{(i)}$, $s_{t+1}^{(i)}$
    end for
  end for
  for $i = 0$ to $B \times N - 1$ **do**
    Sample $\boxed{\text{tips}} \sim \pi_\theta^{\text{old}}(\cdot \mid s^{(i)}, u^{(i)}, \text{tip-generation prompt})$
    Append $\boxed{\text{tips}}$ to $\mathcal{M}$
  end for
  {// Policy update}
  if $m_{\text{rollout}} =$ Memory-Augmented Prompting **then**
    Sample $m_{\text{update}} \sim \{\text{On-Policy} : q,\ \text{Off-Policy} : 1-q\}$
    if $m_{\text{update}} =$ Off-Policy **then**
      for $i = 0$ to $B \times N - 1$ **do**
        $\log \pi_{\theta_{\text{old}}}(a \mid s_t^{(i)}, \boxed{\text{tips}}_t, u^{(i)}) \leftarrow \log \pi_{\theta_{\text{old}}}(a \mid s_t^{(i)}, u^{(i)})$
      end for
    end if
  end if
  Update policy θ using the loss function in Eq. 2.
end for

B PROMPTS

The following prompts were used in our experiments. The ScienceWorld and WebShop prompts were used identically for both the online RL baseline and EMPO^2, with the WebShop prompt adapted from Feng et al. (2025b). The content inside the curly brackets ({}) is dynamically filled based on the current progress at each episode step.

Tip Generation Prompt

Thanks for your playing. Now you have ended a trajectory and collect some meaningless or valuable information from the interactions with the environment. Please summary the trajectory, and also summary what information you get from this trajectory, and how far this trajectory is from fully completing the task. Please response with only one sentence with only one line, do not include any extra words. You sentence should be less than 100 words.

Prompt for ScienceWorld

You have done a few science experiments successfully and below are the action history of your experiments with similar tasks. Here is 2 examples: {example_1} {example_2} Follow the report of the two example tasks shown to you previously, try to solve a similar new task.

Task Description: {task_description}
All your possible action formats are: {available_action_description}

If you enter an unfamiliar room for the first time, you can use the action 'look around' to discover the objects in it. Items in your inventory: {inventory}

Important! You can only use FOCUS actions on these items: {focus_items}. You cannot FOCUS on any other things. Please only use FOCUS as required by the task description. Also, please FOCUS more directly, try not to focus on the container. You could try to explore different actions, especially when you are not sure what the best action for your current observation.
{current_observation}

Prompt for WebShop

You are an expert autonomous agent operating in the WebShop e-commerce environment.

Your task is to: {task_description}.

Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}
You are now at step current_step and your current observation is: {current_observation}.

Your admissible actions of the current situation are:
[{available_actions}].

Now it's your turn to take one action for the current step.

You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <think> </think> tags.

Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

C Detailed Explanation of Importance Sampling Ratios in Policy Updates

To further clarify our policy update mechanism, this section details the calculation of the importance sampling ratio ρ_θ. The specific calculation depends on whether tips were used during the rollout and update phases. This leads to three distinct scenarios, as summarized in Table 3. The importance sampling ratio ρ_θ is defined as the ratio of the probability of an action under the current policy π_θ to its probability under the old policy $\pi_{\theta_{\text{old}}}$, used to correct for the distributional shift in off-policy learning.

Table 3: Calculation of the importance sampling ratio ρ_θ for different policy update modes. The ratio is computed as $\rho_\theta = \frac{\pi_\theta(a_t|\cdot)}{\pi_{\theta_{\text{old}}}(a_t|\cdot)}$, which in practice is often calculated using log-probabilities.

Update Mode	Rollout Condition	Current Log Prob	Old Log Prob
Regular On-Policy	No tips	$\log \pi_\theta(a_t \mid s_t, u)$	$\log \pi_{\theta_{\text{old}}}(a_t \mid s_t, u)$
On-Policy w/ Tips	With tips_t	$\log \pi_\theta(a_t \mid s_t, u, \text{tips}_t)$	$\log \pi_{\theta_{\text{old}}}(a_t \mid s_t, u, \text{tips}_t)$
Off-Policy	With tips_t (rollout)	$\log \pi_\theta(a_t \mid s_t, u)$	$\log \pi_{\theta_{\text{old}}}(a_t \mid s_t, u, \text{tips}_t)$

The three update modes shown in the table cover all scenarios. An update is considered on-policy when the policy used to generate actions ($\pi_{\theta_{\text{old}}}$) and the policy being updated (π_θ) are conditioned on the same information. This applies to the first two modes:

- **Regular On-Policy**: This is the standard on-policy update. The conditioning context for both the current and old policies is identical (s_t, u), with no tips involved.
- **On-Policy w/ Tips**: This mode is also on-policy because both policies are consistently conditioned on the provided tips $(s_t, u, \boxed{\text{tips}}_t)$.

The **Off-Policy** update is the key mechanism through which the model learns from external guidance. In this scenario, actions are sampled from the old policy augmented with tip information: $\pi_{\theta_{\text{old}}}(\cdot \mid s_t, u, \boxed{\text{tips}}_t)$. Consequently, the old log-probabilities are computed using this tip-conditioned policy. However, to "internalize" this guidance, the current log-probabilities for the new policy π_θ are recomputed without the tips, using only $\pi_\theta(a_t \mid s_t, u)$. This mismatch in conditioning between the new and old policies makes the update off-policy and allows the base policy to absorb the knowledge contained in the tips.

D Experiments Details

D.1 Retrospex

In Retrospex (Xiang et al., 2024), the base models differ by environment: Flan-T5-Large (Chung et al., 2024) is used for ScienceWorld, while Llama-3-8B-Instruct (Grattafiori et al., 2024) is used for WebShop. To ensure consistency in our experiments, we standardized the base model to Qwen2.5-7B-Instruct. For this purpose, we utilized the offline trajectories provided by Retrospex and conducted SFT with LLaMA-Factory (Zheng et al., 2024). For the IQL (Kostrikov et al., 2022) Q-function, we employed the model released by Retrospex. During SFT training, we tuned the hyperparameters over learning rates $1.0 \times 10^{-5}, 5.0 \times 10^{-5}, 1.0 \times 10^{-6}$ and epochs $3, 8$, and adopted the configuration that yielded the best performance. Each run was conducted using two NVIDIA A100 GPUs with 80GB memory.

Moreover, in our Retrospex ScienceWorld evaluation, we remove human-designed heuristics to reduce reliance on manual rules. Retrospex normally skips any "focus on" action unless repeated three times or explicitly mentioned in the task, and replaces step-by-step "go to" actions with direct "teleport" moves. Removing these heuristics ensures the evaluation better reflects the agent's inherent capabilities.

D.2 Online RL: ScienceWorld

We base our EMPO^2 implementation on the GRPO framework provided in verl (Sheng et al., 2024), while introducing the following key modifications:

- **Multi-step implementation:** In the original GRPO implementation in verl, an LLM rollout terminates after generating a single response to a given problem. We extend this to a multi-step setting, where the agent continues interacting with the environment until either a maximum episode length is reached or the environment issues a termination signal. This modification allows the agent to perform sequential reasoning and adapt its responses across turns.
- **Memory buffer integration:** To support EMPO^2's memory-based mechanism, we incorporate an explicit memory buffer. During multi-step rollouts, the agent can retrieve tips from memory and append newly generated tips to it. The code snippet for this part is as follows:

```python
# Memory compression and storage utilities
def do_compress(text):
    response = requests.post("http://127.0.0.1:8000/key_cal/", json={"text": text})
    return response.json()

def retrieve_memory(idx, key):
    response = requests.post("http://127.0.0.1:8001/mem/", json={"key": key, "idx": idx})
    return response.json()

def add_memory(idx, key, content, score):
    requests.post("http://127.0.0.1:8001/mem/", json={
```

```python
        "key": key, "idx": idx, "content": content, "score": score
    })

# Use memory retrieval depending on training phase
if phase in ["on-policy-with-memory", "off-policy"]:
    text = "␣".join([f"{c['role']}:␣{c['content']}" for c in pure_chats[i]])
    key = np.array(do_compress(text)['key']).reshape(-1,).tolist()
    count, mem_list = retrieve_memory(random_var, key)
else:
    count, mem_list = 0, []
```

Listing 1: Implementation of memory buffer integration.

- **Off-policy log probability refinement:** To support off-policy updates, we introduce off-policy log probability refinement. For each action region, the on-policy log probabilities are replaced with their off-policy counterparts. The code snippet for this part is as follows:

```python
# Create an off-policy batch by replacing inputs
off_policy_batch = deepcopy(batch)
off_policy_batch.replace_inputs_with_off_policy()

# Compute log probabilities for both on- and off-policy data
off_policy_log_probs = actor.compute_log_prob(off_policy_batch)
on_policy_log_probs = actor.compute_log_prob(batch)

# For each action region, update the original log probs
# with corresponding off-policy values
for gen_id in range(num_generations):
    for region in action_regions[gen_id]:
        loc_l, loc_r = region.on_policy_range
        loc_l_off, loc_r_off = region.off_policy_range
        range_len = loc_r_off - loc_l_off

        # substitute log probs with off-policy values
        on_policy_log_probs[gen_id, loc_l:loc_l+range_len] = \
            off_policy_log_probs[gen_id, loc_l_off:loc_l_off+range_len]

# Update the batch with refined log probs
batch.update_log_probs(on_policy_log_probs)
```

Listing 2: Implementation of off-policy log probability refinement.

Hyperparameters. All online RL algorithms (GRPO, EMPO^2) use the same hyperparameter configuration. The maximum response length is set to 4,500 tokens, and each episode is limited to 30 steps. The actor learning rate is configured as 1×10^{-6}. For GRPO, the group size is fixed at 8, and the mini-batch size is 16. The KL-divergence loss coefficient is set to 0.0. In addition, the actor rollout parameters are specified as follows: the clipping upper bound is set to 0.30, the clipping lower bound to 0.20, and the clipping ratio coefficient to 10.0.

Computing Resources. All experiments were conducted using eight NVIDIA A100 40GB GPUs.

D.3 Online RL: WebShop

We base our EMPO^2 implementation on the GRPO framework provided in verl-agent (Feng et al., 2025b), and the modifications for EMPO^2 are the same as those described in Appendix D.2.

Hyperparameters. All online RL algorithms (GRPO, GiGPO, EMPO^2) use the same hyperparameter configuration, following Feng et al. (2025b). The maximum response length is set to 512 tokens, and each episode is limited to 15 steps. The actor learning rate is configured as 1×10^{-6}. For GRPO, the group size is fixed at 8. The rollout temperature is set to 1.0, while the validation temperature is set to 0.4. The mini-batch size is 64, and the KL-divergence loss coefficient is 0.01. Finally, the discount factor γ is set to 0.95.

Computing Resources. All online RL experiments were conducted using eight NVIDIA A100 GPUs (40GB each).

E QUALITATIVE ANALYSIS ON TIPS

E.1 MORE EXAMPLES OF GENERATED TIPS

Following the example of the generated tips in Section 4.1, below are more detailed examples of how the tips evolve as the task progresses.

ScienceWorld <power-component> task

You focused on the red light bulb but did not complete the task of turning on the red light bulb. You are in the hallway and need to find a way; At that timestep, The specific action your took was focus on red light bulb. **score -100.0**

Trajectory far from completion; connected battery to red wire but not in correct configuration; gained insights on available objects but missed key steps in circuit creation.; At that timestep, The specific action your took was connect battery cathode to red wire terminal 1. **score 0.0**

Connected green wire to green light bulb but did not find the battery to complete the circuit. Trajectory is incomplete as the task requires powering the green light bulb.; At that timestep, The specific action your took was connect battery to green wire terminal 1, connect green wire terminal 2 to green light bulb, connect red wire terminal 1 to battery. **score 7.0**

Connected green wire to green light bulb, but task not fully completed due to missing battery connection.; At that timestep, The specific action your took was connect green wire to green light bulb. **score 13.0**

Task incomplete, terminal connections incorrect; got red wire, battery, and red light bulb, but terminal connections not successfully made to power red light bulb.; At that timestep, The specific action your took was connect battery cathode to red wire terminal 1. **score 27.0**

I persisted in connecting the green wire to the green light bulb but the circuit was interrupted by other wires, affecting the task completion.; At that timestep, The specific action your took was connect green wire terminal 2 to green light bulb. **score 33.0**

You moved to the workshop, connected the circuit, but the green light bulb is still not powered on due to the lack of a power source and proper connections.; At that timestep, The specific action your took was turn on green light bulb. **score 83.0**

The green light bulb was turned on, but the door to the workshop was closed repeatedly.; At that timestep, The specific action your took was open door to workshop. **score 83.0**

ScienceWorld <chemistry-mix-paint-secondary-color> task

You moved to the greenhouse but did not find the necessary materials to create green paint, indicating the task cannot be completed with the current information and location.; At that timestep, The specific action your took was go to kitchen. **score -100.0**

Failed to execute the task due to unknown actions; collected information about paint colors and mixing process but unable to complete the task of creating green paint.; At that timestep, The specific action your took was pour cup containing yellow paint in art studio in bowl, pour cup containing blue paint in art studio in bowl, mix bowl. **score 0.0**

Failed to find a way to the kitchen or other useful areas; picked up an orange but it's not suitable for making green paint. At that timestep, The specific action your took was open door to kitchen. **score 10.0**

Failed to create green paint; mixed blue and yellow paint but action was ambiguous. Collected info on room layout and objects but not sufficient to complete task.; At that timestep, The specific action your took was open door to workshop. **score 30.0**

The agent attempted to pour yellow paint into the bowl in the art studio but could not find the appropriate action, indicating a need to directly interact with the paint containers; At that timestep, The specific action your took was pour cup containing yellow paint in art studio in bowl and pour cup containing blue paint in art studio in bowl and mix bowl. **score 40.0**

The task is to create green paint by mixing blue and yellow paint, but I repeatedly tried to pour blue paint into the bowl without success.; At that timestep, The specific action your took was pour cup containing blue paint in art studio in bowl. **score 40.0**

Created green paint by mixing yellow and blue paint, but focus action was unnecessary as the paint was already in the bowl. At that timestep, The specific action your took was pour cup containing yellow paint in art studio in bowl. **score 80.0**

E.2 Effects of Tips on Exploration Behavior

This section provides a qualitative analysis of how tips promote exploration.

ScienceWorld <power-component> task

[Without Memory]

Task: Your task is to turn on the red light bulb. First, focus on the red light bulb. Then, create an electrical circuit that powers it on.
Observation: This room is called the hallway. In it, you see:
- the agent
- a picture
You also see:
- A door to the workshop (that is open)
...
Action: focus on red bulb

 # Task Failed

[With Memory-Augmented Prompting]

Task: Your task is to turn on the red light bulb. First, focus on the red light bulb. Then, create an electrical circuit that powers it on.
Observation: This room is called the hallway. In it, you see:
- the agent
- a picture
You also see:
- A door to the workshop (that is open)
...
tips: Here are some memories you collected in your previous exploration:
Focus on red light bulb but cannot find it in the hallway.; At that timestep, the specific action your took was focus on red light bulb; Eventually you got the score -100.0/100.
Action: go to workshop
Observation: You move to the workshop.
...

ScienceWorld <chemistry-mix-paint-secondary-color> task

[Without Memory]

Task: Your task is to use chemistry to create green paint. When you are done, focus on the green paint.
Observation: This room is called the hallway. In it, you see:
- the agent
- a picture
You also see:
- A door to the art studio (that is open)
...
Action: pour cup containing yellow paint in hallway in bowl

 # Task Failed

[With Memory-Augmented Prompting]

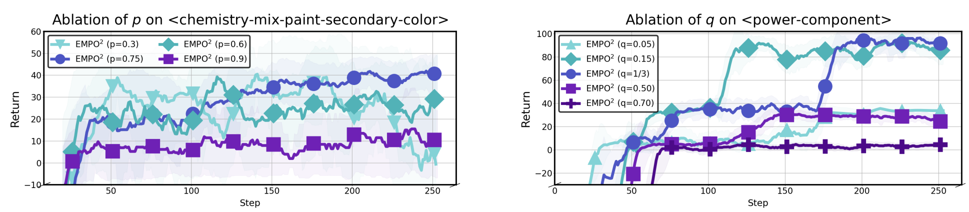
Task: Your task is to use chemistry to create green paint. When you are done, focus on the green paint.
Observation: This room is called the hallway. In it, you see:
- the agent
- a picture
You also see:
- A door to the art studio (that is open)
...
tips: Here are some memories you collected in your previous exploration:
Need to find a green pigment or mixture to create green paint.; At that timestep, the specific action your took was pour cup containing yellow paint in hallway in bowl; Eventually you got the score -100.0/100.
Need to find a green pigment or mixture to create green paint.; At that timestep, the specific action your took was open door to art studio; Eventually you got the score -100.0/100.
Action: go to art studio
Observation: You move to the art studio.
...

As shown in the examples above, an agent without memory tends to repeat the same mistakes because it cannot incorporate feedback from previous failures into future attempts. In contrast, with memory-augmented prompting, the agent can refer to its past unsuccessful attempts, use them as guidance, and actively avoid repeating those errors. This enables the agent to explore novel and more effective behaviors, ultimately expanding its search capabilities and boosting learning performance.

F MORE ABLATION STUDY

F.1 MODE SELECTION PROBABILITY

As discussed in Section 4.2, EMPO^2 employs a memory-rollout probability p during the rollout phase and an off-policy update probability q during the update phase. We conduct comprehensive ablation studies to systematically investigate the effects of these hyperparameters p and q.



(a) (b)

Figure 10: (a) EMPO² learning curves with varying p, (b) with varying q

Ablation on p (Memory Rollout Probability): We evaluated $p \in \{0.3, 0.6, 0.9\}$ on the `chemistry-mix-paint-secondary-color` task. When $p = 0.9$, performance degrades significantly because EMPO^2 effectively collapses to GRPO, confirming the importance of memory. Both $p = 0.3$ and $p = 0.6$ show faster initial learning due to more aggressive knowledge internalization, although $p = 0.3$ exhibits minor fluctuations in the later stages. Our choice of $p = 0.75$ provides stable convergence across diverse tasks.

Ablation on q (Off-Policy Update Probability): We tested $q \in \{0.05, 0.15, 0.5, 0.70\}$ on the `power-component` task. Extreme values ($q = 0.05$ or $q = 0.70$) underperform: very small q overemphasizes distillation at the expense of training the memory policy, while large q slows knowledge internalization. Notably, $q = 0.15$ achieves faster early exploration than our default $q = 1/3$. This aligns with our expectations, as the default hyperparameters prioritize overall robustness rather than task-specific optimization. Therefore, it is natural that more optimal settings exist for particular tasks, highlighting the robustness of EMPO² within a reasonable hyperparameter range.

These results confirm that EMPO^2 performs effectively across a broad hyperparameter range ($p \in [0.6, 0.75]$, $q \in [0.15, 0.5]$). Our default settings represent a well-balanced configuration that generalizes across multiple tasks without task-specific tuning, while the algorithm remains adaptable when further optimization is desired.

F.2 ROLE OF INTRINSIC REWARD

To further investigate the role of the intrinsic reward in our proposed algorithm, EMPO², we conduct an ablation study to examine its impact. We compare our full method against variants with different intrinsic reward coefficients (0.5× and 2×), a complete removal of the intrinsic reward, and its replacement with a standard exploration bonus based on Random Network Distillation (RND) (Burda et al., 2018a). For the RND baseline, we adopt the same hyperparameter configuration as in the original work. The results of these experiments are presented in Figure 11.

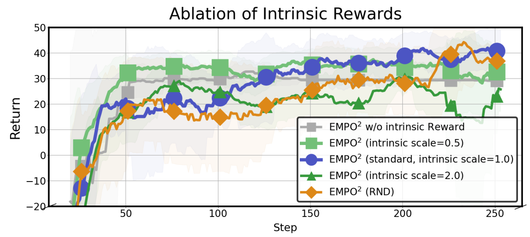


Figure 11: EMPO^2 learning curves with different intrinsic reward configurations on ScienceWorld `chemistry-mix-paint-secondary-color` task. We compare our full method against four variants: scaling the intrinsic reward coefficient by 0.5× and 2×, substituting it with a Random Network Distillation (RND) bonus, and its complete removal (w/o Intrinsic Reward).

Altering the intrinsic reward's scale mainly affects the learning dynamics. A smaller coefficient (0.5×) leads to a smoother but slower convergence, whereas a larger one (2×) introduces minor instabilities. Notably, all variants using an intrinsic reward—including the RND-based one—converge to a similar level of final performance. However, removing the intrinsic reward entirely causes learning to plateau at a lower level, suggesting its necessity in preventing the policy from collapsing into homogeneous behaviors by encouraging sufficient exploration. Overall, these results indicate that EMPO² is robust to the specific mechanism and scale of the intrinsic reward, which primarily influence the stability and speed of learning rather than the final outcome.

G Analysis of Computational Cost

G.1 Cost Analysis of Memory-Augmented Rollouts

We analyzed the additional computational overhead introduced by the memory mechanism in EMPO². During the rollout phase, this mechanism incurs extra costs related to tip generation, retrieval, and storage. For the analysis, we conducted experiments using the Qwen2.5-7B-Instruct model on 8 A100 40GB GPUs.

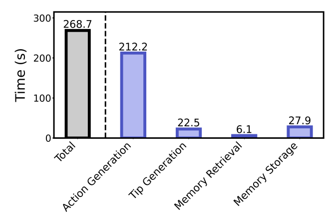


Figure 12: A breakdown of the time each component spends during the rollout of each training step.

As reported in Figure 12, the memory mechanism adds approximately 50.4 seconds per iteration, which accounts for about 19% of the total rollout time. Among these, tip generation and the subsequent storage of tips in memory account for a substantial portion of the cost. Therefore, while we have verified that the memory mechanism substantially aids exploration and significantly improves learning efficiency, it is more desirable to internalize these benefits within the model parameters themselves rather than relying on the mechanism continuously—both to enhance the model's inherent capabilities and to improve overall efficiency.

G.2 Cost Analysis of Total Training Time

Compared to GRPO, the training time of EMPO^2 is primarily influenced by two factors:

- **The memory component**: As discussed in the previous section, the memory component accounts for 19% of the total rollout time. Since memory-augmented prompting is selected with probability ($1 - p = 0.25$) (as described in Section 4.2), this implies that, on average, 19% of the rollout time is incurred with a 25% probability.
- **The response length:** In LLM-based RL training, rollout time constitutes a major portion of the total cost. As the response length increases, the rollout itself becomes slower, and the time

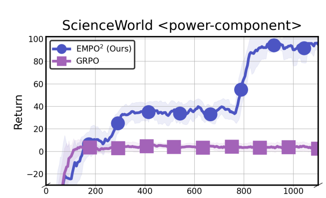


Figure 13: Time–performance curves for EMPO^2 and GRPO on ScienceWorld `power-component` task.

required for log-probability computation and actor updates increases accordingly. In our experiments, we found that the response length of EMPO^2 is generally longer than that of GRPO. We attribute this to the model spending more time reasoning and exploring when given the tips, which we believe enhances its exploration behavior and ultimately improves performance.

To ensure a fair comparison with GRPO from a training-time perspective, we plot the performance in Figure 13 using training time on the x-axis. The results show that, even under this perspective, EMPO^2 exhibits substantially higher efficiency than GRPO.

H The Use of Large Language Models

We used a LLM to polish the writing of the manuscript. The LLM was not employed in any aspect of research ideation, experimental design, or analysis.