MAGE: META-REINFORCEMENT LEARNING FOR LANGUAGE AGENTS TOWARD STRATEGIC EXPLORATION AND EXPLOITATION

Lu Yang*, **Zelai Xu***, **Minyang Xie, Jiaxuan Gao, Zhao Shok, Yu Wang†, Yi Wu†**
Tsinghua University

ABSTRACT

Large Language Model (LLM) agents have demonstrated remarkable proficiency in learned tasks, yet they often struggle to adapt to non-stationary environments with feedback. While In-Context Learning and external memory offer some flexibility, they fail to internalize the adaptive ability required for long-term improvement. Meta-Reinforcement Learning (meta-RL) provides an alternative by embedding the learning process directly within the model. However, existing meta-RL approaches for LLMs focus primarily on exploration in single-agent settings, neglecting the strategic exploitation necessary for multi-agent environments. We propose MAGE, a meta-RL framework that empowers LLM agents for strategic exploration and exploitation. MAGE utilizes a multi-episode training regime where interaction histories and reflections are integrated into the context window. By using the final episode reward as the objective, MAGE incentivizes the agent to refine its strategy based on past experiences. We further combine population-based training with an agent-specific advantage normalization technique to enrich agent diversity and ensure stable learning. Experiment results show that MAGE outperforms existing baselines in both exploration and exploitation tasks. Furthermore, MAGE exhibits strong generalization to unseen opponents, suggesting it has internalized the ability for strategic exploration and exploitation. Code is available at `https://github.com/Lu-Yang666/MAGE`.

1 INTRODUCTION

Reinforcement Learning (RL) has significantly enhanced the ability of Large Language Model (LLM) agents to solve complex, multi-step tasks Guo et al. (2025); Comanici et al. (2025). However, despite their proficiency in static environments, these agents typically lack the capacity to incorporate real-time feedback or refine their strategies when encountering shifting dynamics. Bridging this gap requires a transition from fixed execution to test-time adaptation. By enabling agents to "learn from experience" during interaction, their behavior can evolve dynamically across several trials. This capability is essential for transforming static task-solvers into adaptive learners that can autonomously adjust to the complexities of non-stationary environments.

Current approaches to achieve such test-time adaptation primarily rely on prompting through In-Context Learning (ICL) Brown et al. (2020); Shinn et al. (2023) or external memory components Xu et al. (2025); Zhou et al. (2025). Although these methods provide some flexibility, they often fail to fundamentally internalize the adaptive capability. Meta-reinforcement learning (meta-RL) Duan et al. (2016); Finn et al. (2017) offers a more robust alternative by embedding the learning mechanism directly within the model itself. While recent efforts have applied meta-RL to LLM agents Jiang et al. (2025), they focus almost exclusively on inducing exploration in single-agent tasks. In multi-agent environments, however, an agent must adapt not only to the environment but also to the diverse behaviors of other participants. This necessitates strategic exploitation, which involves dynamically identifying and capitalizing on opponent-specific patterns. Since a strategy

*Equal contribution: `yanglu25@mails.tsinghua.edu.cn, zelai.eecs@gmail.com`
†Corresponding authors: `yu-wang@tsinghua.edu.cn, jxwuyi@gmail.com`

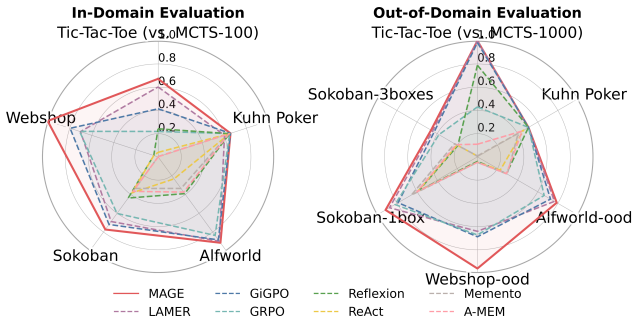

Figure 1: Evaluation of MAGE in exploration and exploitation tasks. MAGE outperforms existing training and training-free baselines in a number of environments.

optimized for one adversary might fail against another Czarnecki et al. (2020), this requires a fundamental shift from environment exploration to agent exploitation.

To bridge this gap, we present **MAGE**, a **M**eta-RL framework that empowers LLM **A**gents for strate**G**ic Exploration and **E**xploitation in multi-agent environments. MAGE adopts a multi-episode training regime where the agent's interaction history and reflections from previous episode are integrated into the LLM's context window. By optimizing the policy across these interaction trajectories via RL, the LLM internalizes the ability to learn from past experience. Unlike prior work Jiang et al. (2025) that focuses on maximizing cumulative rewards to incentivize exploration, MAGE prioritizes the final episode reward as the primary objective. This metric serves as the measure of how effectively an agent has identified the weaknesses of its adversary and adapted its strategy to maximize performance. The core idea of MAGE is to transform the LLM agent into a strategic learner that views its interaction history not just as a record of events, but as the strategic basis for exploiting the opponents' vulnerabilities.

Meta-RL in multi-agent environments for strategic explotation and exploitation presents unique challenges, as training against a single agent is insufficient for developing generalizable exploitation skills and fails to prepare the model for the varied flaws found in diverse opponents Czarnecki et al. (2020). Therefore, we leverage Population-Based Training (PBT) Jaderberg et al. (2017); Vinyals et al. (2019) to expose the LLM to a diverse pool of different agents. In the context of this work, we employ Population-Based Training (PBT) to refer to the process of training the LLM agent against a diverse pool of fixed opponent archetypes. By interacting with a population of agents, MAGE learns to recognize specific behavioral patterns and exploit their inherent strategic vulnerabilities. We further introduce an agent-specific advantage normalization technique to handle divergent reward distributions across opponents, which necessitate separate normalization. Our agent-specific normalization ensures that the agent can effectively use its context window as a strategic buffer to distinguish between different types of adversaries and respond with the appropriate counter-strategy. By integrating population-based training with agent-specific normalization, MAGE establishes a principled meta-RL framework for learning strategic exploration and exploitation in language agents.

We conduct extensive experiments to validate MAGE across diverse benchmarks. In single-agent (Alfworld, Webshop, Sokoban) and multi-agent (Tic-Tac-Toe, Kuhn Poker) settings, MAGE consistently outperforms baselines with a rapid adaptation curve. Notably, it achieves a 100% success rate in Webshop and 91.4% in Alfworld, significantly surpassing the strongest baselines of 79.7% and 88.3%. In Tic-Tac-Toe, MAGE reaches a 67.2% success rate, exceeding LAMER's 60.2%. Evaluation against unseen opponents further demonstrates robustness: MAGE achieves 96.1% in Webshop-ood (vs. 68.8%), maintains top performance in Sokoban, and reaches a 100.0% draw rate against MCTS-1000 in Tic-Tac-Toe. In Kuhn Poker, it hits the theoretical upper bound against CFR opponents. These results suggest MAGE internalizes a fundamental logic for zero-shot adaptation rather than mere pattern memorization. Finally, ablation studies confirm that the synergy between population-based training and agent-specific normalization is crucial for identifying and exploiting opponent vulnerabilities.

We summarize our contributions as follows:

- We propose MAGE, a meta-RL framework that empowers language agents for strategic exploration and exploitation in multi-agent environments.
- We introduce an effective training recipe that combines population-based training with agent-specific advantage normalization to provide diverse opponents and stable training signals for meta-RL.
- We conduct extensive experiments showing that MAGE improves adaptation and achieves higher win rates against in-domain and unseen opponents.

2 Related Work

In-Context Learning The capacity of Large Language Models (LLMs) to adapt via In-Context Learning (ICL) Brown et al. (2020) has catalyzed the development of autonomous agents. To move beyond static prompting, recent works such as *Reflexion* Shinn et al. (2023) and *Self-Refine* Madaan et al. (2023) introduce iterative feedback loops, allowing agents to correct errors based on environment trials. Furthermore, memory-augmented frameworks Xu et al. (2025); Zhou et al. (2025) enable agents to retrieve past experiences from external databases. However, these methods rely on fixed model weights and often fail to internalize the underlying learning logic, leading to sub-optimal adaptation in complex, non-stationary settings.

Agentic RL for LLMs Reinforcement Learning (RL) has shifted from simple preference alignment to enhancing complex reasoning and multi-turn decision-making. High-profile models like OpenAI o1 Lightman et al. (2023) and DeepSeek-R1 Guo et al. (2025) demonstrate the power of RL in scaling computational thought chains. In the agentic context, RL is increasingly applied to multi-step tasks such as web search Jin et al. (2025), software engineering Wei et al. (2025), and GUI interactions Wang et al. (2025). Specialized algorithms like GiGPO Feng et al. (2025) have been proposed to stabilize training for such long-horizon interaction trajectories. MAGE builds upon this agentic RL trend but shifts the focus from mastering a single task to mastering the process of adaptation itself.

Meta-Reinforcement Learning Traditional Meta-RL Duan et al. (2016); Finn et al. (2017) aims to train agents that can rapidly adapt to new tasks by internalizing the learning procedure. Recently, this paradigm has been extended to LLMs; for instance, LAMER Jiang et al. (2025) utilizes meta-RL to incentivize efficient exploration in single-agent environments. However, in multi-agent or competitive scenarios, agents must perform *strategic exploitation*—identifying and capitalizing on the specific vulnerabilities of opponents. Unlike previous works that focus on exploration, MAGE leverages population-based training and agent-specific normalization to ensure the agent can robustly exploit diverse behaviors, representing a novel frontier for meta-RL in language agents.

3 MAGE

In this section, we present **MAGE**, a **M**eta-RL framework designed to optimize LLM **A**gents for strate**G**ic Exploration and **E**xploitation in multi-agent environments. Unlike standard In-Context Learning (ICL), which relies on emergent behaviors, MAGE explicitly trains the model to *learn to learn* by treating a sequence of interaction episodes as an inner optimization loop.

3.1 Problem Setup

We define a *meta-episode* as a sequence of N episodes

$$\mathcal{E} = \{\tau_1, \tau_2, \ldots, \tau_N\},$$

executed against a stationary task or opponent. Each episode τ_n corresponds to a complete trajectory:

$$\tau_n = \{(s_{n,1}, a_{n,1}, r_{n,1}), \ldots, (s_{n,T}, a_{n,T}, r_{n,T})\}, \tag{1}$$

where $s_{n,t}$, $a_{n,t}$, and $r_{n,t}$ denote the state, action, and reward at step t of episode n, respectively.

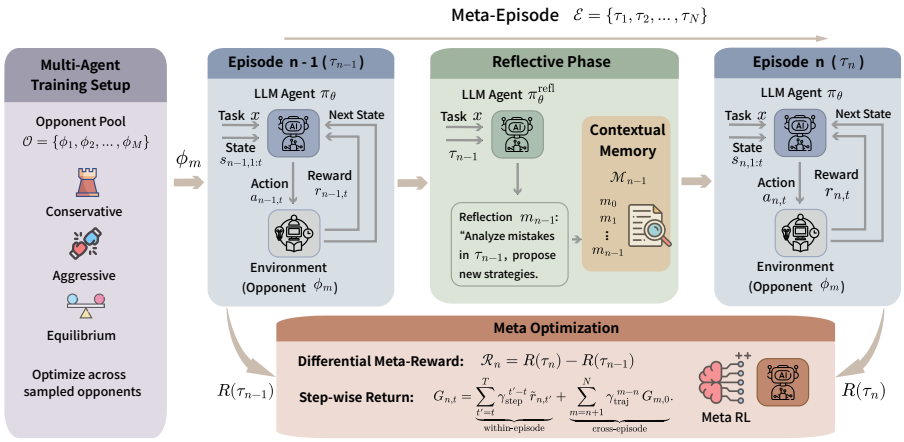

Figure 2: Overview of the MAGE framework. MAGE optimizes an LLM policy π_θ across N episodes using a contextual memory $\mathcal{M}_{n-1}$ updated via self-reflections m_{n-1}. By training against an opponent pool $\mathcal{O}$ and applying agent-specific advantage normalization to the final episode reward, MAGE facilitates stable strategic adaptation and learning-to-learn.

3.2 The MAGE Framework

Unlike standard ICL, our framework introduces a *Reflective Inner Loop* in which the model explicitly generates and exploits its own high-level feedback across episodes. At the conclusion of episode τ_{n-1}, the model produces a self-generated reflection:

$$m_{n-1} \sim \pi_\theta^{\text{refl}}(\cdot \mid \tau_{n-1}, x), \tag{2}$$

where π_θ^{refl} denotes the model's reflection-generation behavior and x denotes the specific task description. The reflection m_{n-1} is expressed in natural language and is intended to summarize failure modes, diagnose strategic errors, and propose corrective actions.

The sequence of past reflections is organized into a *contextual memory*:

$$\mathcal{M}_{n-1} = \{m_0, m_1, \ldots, m_{n-1}\}, \tag{3}$$

which serves as a compact, high-level abstraction of accumulated experience across episodes.

Within episode n, the model generates actions based on the task description, the contextual memory formed by all prior reflections, and the state history observed so far in the current episode. Formally, let

$$s_{n,1:t} = \{s_{n,1}, \ldots, s_{n,t}\}$$

denote the state history up to step t in episode n. The action distribution is then defined as:

$$a_{n,t} \sim \pi_\theta(\cdot \mid s_{n,1:t}, \mathcal{M}_{n-1}, x)\,. \tag{4}$$

For the initial episode ($n = 1$), $\mathcal{M}_0 = \{m_0\}$ is initialized as an empty reflection.

3.2.1 Step-wise Return

To facilitate fine-grained policy updates, we compute a step-wise return $G_{n,t}$ for each action $a_{n,t}$. In our setting, the environment provides a *sparse* reward: all task-related reward is emitted only at the final step of each episode. Consequently, the learning signal is defined primarily at the episode level, and is injected into the step-wise return in a structured manner.

Episode-wise differential meta-reward To explicitly optimize learning progress across episodes, we define the episode-wise differential meta-reward:

$$\mathcal{R}_n = R(\tau_n) - R(\tau_{n-1}), \tag{5}$$

with $R(\tau_0) \equiv 0$. $R(\tau_n)$ denotes the cumulative task reward of episode n. This signal measures the improvement achieved by the policy update induced by the reflection from the previous episodes.

The step-wise reward is then formulated as:

$$\tilde{r}_{n,t} = \begin{cases} 0, & t < T, \\ \mathcal{R}_n, & t = T. \end{cases} \tag{6}$$

Inspired by LaMER, the step-wise return for action $a_{n,t}$ is defined as:

$$G_{n,t} = \underbrace{\sum_{t'=t}^{T} \gamma_{\text{step}}^{t'-t}\, \tilde{r}_{n,t'}}_{\text{within-episode}} + \underbrace{\sum_{m=n+1}^{N} \gamma_{\text{traj}}^{m-n}\, G_{m,0}}_{\text{cross-episode}}. \tag{7}$$

where γ_{step} and γ_{traj} denote the discount factors for within-episode and cross-episode returns, respectively. For more information on the composition of episode reward, see Appendix **??**.

As for the return $G_{m_{n-1}}$ of the reflection m_{n-1} is equal to the return of the first step of the n-th episode:

$$G_{m_{n-1}} = G_{n,0} \tag{8}$$

Reward design The total episode reward in our framework is structured as a composite signal consisting of three key terms: (i) **Task reward**, which measures primary objective success, assigning $+10$ for wins/successes, -10 for losses/failures, and 0 for neutral outcomes; (ii) **Invalid action penalty**, a fixed deduction of 0.5 applied whenever the model generates inadmissible or malformed actions to enforce rule adherence; and (iii) **Length penalty**, designed to prevent verbosity by penalizing responses that exceed half of the maximum length $L_{\max}$. Specifically, the length penalty r_{length} is defined as:

$$r_{\text{length}} = \begin{cases} 0, & L < \frac{1}{2} L_{\max}, \\ \dfrac{L - \frac{1}{2} L_{\max}}{0.5 L_{\max}}, & \frac{1}{2} L_{\max} \le L < L_{\max}, \\ 1, & L \ge L_{\max}. \end{cases} \tag{9}$$

This combined reward structure guides the model to produce correct, valid, and concise responses while maintaining a strong focus on task performance.

3.2.2 Optimization Objective

The overall objective of MAGE is to maximize the expected cumulative meta-reward across a meta-episode:

$$\max_{\theta}\ \mathbb{E}_{\tau_1,\ldots,\tau_N \sim \pi_\theta} \left[\sum_{n=1}^{N} \mathcal{R}_n \right], \tag{10}$$

where π_θ denotes the LLM policy parameterized by θ.

Using the derived advantages $\hat{A}_{n,t}$ calculated from the step returns $G_{n,t}$, the policy is optimized via a generalized policy gradient objective. By utilizing the advantage as a weighting factor for the log-probabilities of actions and reflections, the meta-learning objective is formulated as:

$$\mathcal{L}_{\text{episode}}(\theta) = -\sum_{n=1}^{N} \sum_{t=1}^{T} \hat{A}_{n,t} \log \pi_\theta(a_{n,t} \mid s_{n,1:t}, \mathcal{M}_{n-1}, x), \tag{11}$$

$$\mathcal{L}_{\text{reflection}}(\theta) = -\sum_{n=2}^{N} \hat{A}_{n,0} \log \pi_\theta(m_{n-1} \mid \tau_{n-1}, x) \tag{12}$$

$$\mathcal{L}_{\text{MAGE}}(\theta) = \mathbb{E}_{\mathcal{E} \sim \pi_\theta} \left[\mathcal{L}_{\text{episode}}(\theta) + \mathcal{L}_{\text{reflection}}(\theta) \right]. \tag{13}$$

This objective explicitly encourages the model to acquire strategies that improve its own learning dynamics over successive episodes. It is worth noting that the MAGE framework is algorithm-agnostic; while the basic objective is presented as a policy gradient, it is fully compatible with advanced reinforcement learning algorithms such as GiGPO Feng et al. (2025). In these cases, the advantage $\hat{A}_{n,t}$ serves as the core signal for the respective surrogate loss functions, effectively training the model to optimize its strategy refinement process across the meta-episode.

3.2.3 AGENT-SPECIFIC ADVANTAGE NORMALIZATION

In multi-agent setting, the LLM agent must identify and exploit the behavioral patterns of diverse opponents. We implement two games (*Tic-Tac-Toe*, *Kuhn Poker*) that represent different strategic challenges.

During the training phase, the LLM agent interacts with a population of opponents $\mathcal{O} = \{\phi_1, \phi_2, \ldots, \phi_M\}$, where each ϕ_m represents a distinct fixed strategy. A meta-episode consists of N episodes against a single opponent ϕ_m sampled from $\mathcal{O}$. The agent is not explicitly told which strategy it is facing; instead, it must infer the opponent's play pattern from the contextual memory $\mathcal{M}_{n-1}$ and the current episode's state history $s_{n,1:t}$, and adjust its policy π_θ accordingly.

Population-Based Training (PBT) in MAGE In the context of this work, we employ Population-Based Training (PBT) to refer to the process of training the LLM agent against a diverse pool of fixed opponent archetypes $\mathcal{O}$. Unlike traditional PBT settings that involve evolutionary meta-optimization or periodic weight replacement, our approach utilizes the population to ensure strategic robustness and to force the model to perform high-level inference.

Let $\hat{A}_{n,t}^{(m)}$ denote the normalized step-wise advantage derived from interactions with opponent ϕ_m. The multi-agent MAGE objective maximizes the expected advantage-weighted log-likelihood of reflections and actions across all steps and episodes within the meta-episode, and across sampled opponents:

$$\mathcal{L}_{\phi_m}(\theta) = -\sum_{n=1}^{N}\sum_{t=1}^{T} \hat{A}_{n,t}^{(m)} \log \pi_\theta\big(a_{n,t} \mid s_{n,1:t}, \mathcal{M}_{n-1}, x\big), \tag{14}$$

$$\mathcal{L}_{\phi_m,\, reflection}(\theta) = -\sum_{n=2}^{N} \hat{A}_{n,0}^{(m)} \log \pi_\theta(m_{n-1} \mid \tau_{n-1}, x), \tag{15}$$

$$\mathcal{L}_{\text{MAGE}}^{\text{multi-agent}}(\theta) = \mathbb{E}_{\phi_m \sim \mathcal{O}}\left[\mathcal{L}_{\phi_m}(\theta) + \mathcal{L}_{\phi_m,\, reflection}(\theta)\right]. \tag{16}$$

This objective encourages the agent to select actions that maximize learning progress against a heterogeneous population of opponents, effectively promoting strategy adaptation and generalization across different play patterns.

3.2.4 TRAINING PROCEDURE

MAGE is trained using a meta-episode reinforcement learning framework where the agent iteratively improves its behavior through self-generated reflections. In each episode, the LLM policy interacts with the environment conditioned on the task description and the accumulated contextual memory. After each episode, the agent computes a differential meta-reward based on the performance improvement over the previous episode. If the episode is not the last in the meta-episode, the model generates a reflection from the trajectory and stores it in memory to guide future episodes. After the meta-episode ends, MAGE performs backward credit assignment across both time steps and episodes to compute advantages for actions and reflections, and updates the policy by jointly optimizing the action and reflection losses. The complete training procedure algorithm is provided in Appendix A.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We employ Qwen3-4B as our base large language model, utilizing its native Thinking capabilities to facilitate complex reasoning during both the reflection and action generation phases.The training is conducted using the GiGPO algorithm. We adopt GiGPO as our optimization algorithm because it provides a dual-level (episode and step-wise) relative advantage estimation, enabling fine-grained credit assignment in multi-turn agent tasks while maintaining the high memory efficiency and stability of critic-free group-based RL. For the meta-learning objective, we set the cross-episode discount

Category	Method	Multi-Agent Tasks		Single-Agent Tasks		
		Kuhn Poker	Tic-Tac-Toe	ALFWorld	Sokoban	WebShop
In-Context Learning	ReAct	0.648	0.039	0.234	0.383	0.039
	Reflexion	0.648	0.242	0.391	0.438	0.039
Memory-based	A-MEM	0.641	0.016	0.375	0.367	0.000
	Memento	0.641	0.031	0.336	0.336	0.000
RL	GRPO	0.648	0.219	0.836	0.602	0.711
	GiGPO	**0.656**	0.414	0.883	0.719	0.797
Meta-RL	LAMER	0.594	0.602	0.898	0.688	0.703
	MAGE	**0.656**	**0.672**	**0.914**	**0.773**	**1.000**

Table 1: **In-domain evaluation performance of the final episode.** MAGE consistently reaches or outperforms existing methods.

factor $\gamma_{\text{traj}} = 0.6$. Each meta-episode consists of $N = 3$ episodes. During MAGE training, we utilize a group size of 8 meta-episodes per batch. In contrast, for the standard RL baselines, we expand the group size to 24 to maintain an equivalent number of total trajectories per update, ensuring a fair comparison of sample efficiency. In multi-agent environments, we utilize population-based training: for Tic-Tac-Toe, the agent interacts with MCTS-based, preferred-pattern, and random strategies; for Kuhn Poker, the training distribution consists of conservative, aggressive, and intermediate opponent archetypes.

Environments. We evaluate MAGE across diverse strategic benchmarks. Multi-agent tasks include *Tic-Tac-Toe*, a perfect-information game for assessing rapid adaptation to deterministic optimal play, and *Kuhn Poker*, an imperfect-information variant testing strategic reasoning and bluffing. Single-agent tasks include *ALFWorld* (interactive household planning), *WebShop* (goal-oriented web navigation), and *Sokoban* (long-horizon spatial puzzles).

Baselines. We compare MAGE against a broad set of baselines encompassing heuristic agent frameworks (*ReAct* Yao et al. (2022), *Reflexion* Shinn et al. (2023)), memory-augmented agents (*A-MEM* Xu et al. (2025), *Memento* Zhou et al. (2025)), foundational reinforcement learning methods (*GRPO* Shao et al. (2024), *GiGPO* Feng et al. (2025)), and related meta-learning approaches (*LAMER* Jiang et al. (2025)).

Baseline Fairness and Input Protocol To ensure a rigorous evaluation of the meta-learning objective, we maintain strict context parity between MAGE and the comparison baselines. The input provided to each model at step t of episode n is structured as follows:

- **Standard RL Baselines (GRPO, GiGPO):** The context includes the task prompt, the initial environment state ($s_{n,1}$), and the current episode's trajectory history. To manage context length while preserving local dynamics, the history consists of the full state-action sequence for the most recent 7 steps, and state-only observations for the most recent 2 steps.
- **Meta-RL Baselines (LaMER, MAGE):** In addition to the standard RL context described above, these models are granted access to the *contextual memory* $\mathcal{M}_{n-1}$. This memory contains the sequence of self-generated reflections $\{m_1, \ldots, m_{n-1}\}$ derived from prior episodes within the same meta-episode.

Importantly, the underlying task information, initial observations, and the sliding-window protocol for the current trajectory are identical across all models. The primary distinction lies in the inclusion of self-generated reflections, allowing us to isolate the impact of strategic abstraction on cross-episode adaptation.

Metrics. The primary evaluation metric across all environments is the *Success Rate*, reported under the *Pass@k* formulation. Specifically, Pass@k measures the probability that an agent successfully completes the task at least once within the first k episodes of a meta-episode.

Our evaluation consists of three parts to systematically assess strategic plasticity and the Final-Episode Optimization objective:

- **In-Domain Evaluation:** Assesses fundamental learn-to-learn performance and the conversion of early interactions into exploitative strategies under training-consistent distributions.
- **Generalization and Cross-Domain Plasticity:** Tests robustness against out-of-domain (OOD) tasks and unseen opponents to verify a generalizable probing logic rather than pattern memorization.
- **Ablation Studies:** Deconstructs the framework to analyze the impact of Final-Episode Optimization, Population-Based Training (PBT), and Opponent-Specific Advantage Normalization.

4.2 In-Domain Evaluation

We evaluate the effectiveness of MAGE under in-domain conditions, where evaluation environments and opponent distributions align with the training phase. This isolates the impact of the proposed Final-Episode Optimization objective and its ability to foster strategic plasticity in LLM agents. The results are shown in Table 1.

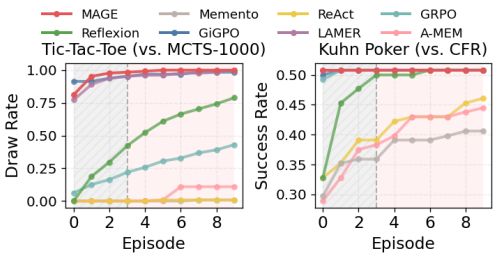

Figure 3: **Multi-Agent Evaluation.** Performance in Tic-Tac-Toe (vs. MCTS-1000) and Kuhn Poker (vs. CFR).

Our training objective explicitly optimizes the final episode return within a three-episode trajectory. Consequently, the policy is not incentivized to maximize rewards in the initial two episodes, where performance may naturally trail baselines. The effectiveness of MAGE is best observed from the third episode onward, as this phase reflects the completed adaptation process targeted by our objective. Therefore, we emphasize performance in the third episode and beyond to provide a faithful evaluation of the method's strategic optimization.

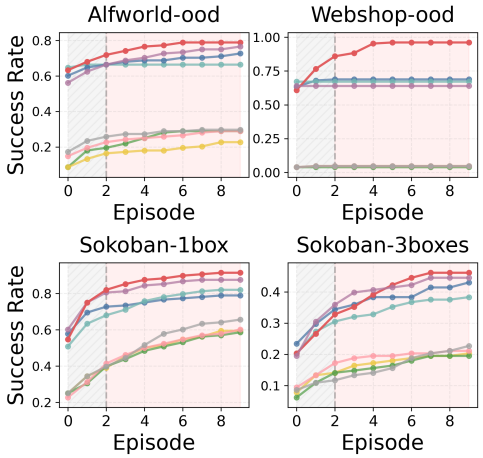

Figure 4: **Single-Agent Evaluation.** Performance in ALFWorld, Sokoban, and WebShop.

Multi-Agent Strategic Exploitation. The evaluation in multi-agent settings (Tic-Tac-Toe and Kuhn Poker) highlights MAGE's capacity for **strategic exploitation** of opponent-specific idiosyncrasies.

In Tic-Tac-Toe, MAGE achieves a dominant 67.2% terminal success rate against MCTS-100, significantly outperforming LAMER (60.2%) and GiGPO (41.4%). In Kuhn Poker, MAGE hits the 65.6% theoretical upper bound, matching the performance ceiling despite the task's stochasticity. These results validate that Opponent-Specific Advantage Normalization effectively stabilizes meta-learning across heterogeneous strategy populations.

Single-Agent Exploration. In complex single-agent tasks (ALFWorld, Sokoban, WebShop), MAGE consistently achieves superior terminal performance, proving that prioritizing terminal success over cumulative reward fosters more effective *in-context adaptation*.

In WebShop, MAGE transitions from a 66.4% initial success rate to 100% **by the 5th episode**, outperforming baselines like GiGPO and LAMER by $20 - 30\%$. This highlights its ability to convert early feedback into flawless execution. In Sokoban, MAGE demonstrates a "slow-start, high-finish"

pattern, improving from 40.6% to 77.3% ($+36.7\%$). This confirms that Final-Episode Optimization incentivizes strategic probing over conservative play. In ALFWorld, MAGE reaches a 91.4% Pass@10, surpassing LAMER (89.8%) and GiGPO (88.3%), while leaving pure prompting methods like Reflexion below 40%.

Discussion on Meta-Learning Capability. Unlike static methods (ReAct, Reflexion) that fail to improve across episodes, MAGE treats interaction history as a "meta-context." By optimizing for terminal success, it systematically transitions from early *information-gathering* to late-episode *exploitation*, achieving true strategic plasticity.

4.3 Generalization and Cross-Domain Plasticity

We evaluate MAGE under out-of-domain (OOD) conditions to assess its strategic exploitation across shifted task complexities and novel opponent behaviors.

Multi-agent Evaluation. Facing MCTS-1000 in Tic-Tac-Toe—where winning is nearly impossible—MAGE's draw rate ascends from 81.2% to 100.0% by the final episode. This demonstrates its ability to identify perfect defensive patterns and recalibrate to prevent exploitation. Against CFR opponents in Kuhn Poker, MAGE reaches the 50.8% theoretical ceiling. This stability validates that Opponent-Specific Advantage Normalization prevents policy collapse when encountering near-optimal behaviors.

Single-agent Evaluation. In Sokoban, despite being trained only on 2-box configurations, MAGE achieves 91.4% in 1-box and 46.1% in 3-boxes variants, outperforming GiGPO. In WebShop, MAGE maintains a 96.1% success rate (vs. 68.8% for GiGPO), and in AlfWorld, it preserves a high terminal performance of 78.9%. These results suggest MAGE develops a robust information-gathering mechanism that generalizes to idiosyncratic features under distributional shifts rather than merely memorizing patterns.

4.4 Ablation Studies

We conduct controlled ablations on MAGE's core components: reward design, population-based training, and agent-specific advantage normalization, maintaining fixed hyperparameters and budgets.

4.4.1 Reward Design

We compare three reward formulations while preserving the meta-episode structure: **Differential Return (MAGE):** Uses episode-wise progress, $\hat{r}_{n,T} = R(\tau_n) - R(\tau_{n-1})$. **Cumulative Return (LAMER-style):** Uses absolute performance, $\hat{r}_{n,T} = R(\tau_n)$, with cross-episode propagation. **Single-episode Return:** Uses $\hat{r}_{n,T} = R(\tau_n)$ without cross-episode propagation.

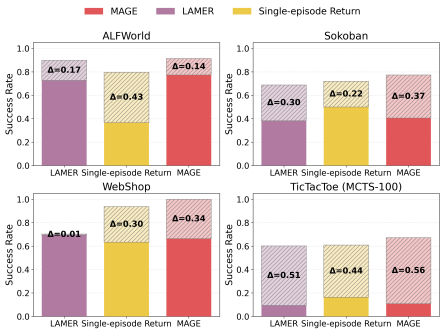

Figure 5: **Reward design ablation results.** Using single-episode return and LAMER variants causes the success rate to drop in various environments.

Results in Figure 5 show that MAGE's **Differential Return** is the primary driver for steep learning curves, reaching the highest success rates (e.g., 91.4% in Alfworld; 100% in Webshop) with stable gains. **Cumulative Return** is inconsistent; while competitive in Alfworld (89.8%), it fails in Webshop ($\Delta \approx 0.8\%$), suggesting that absolute return targets can be brittle. **Single-episode Return** achieves relative improvements but lower final averages, indicating it lacks the cross-episode exploitation strength of the differential formulation.

4.4.2 Population-Based Training

We ablate the training distribution and sampling structure in Tic-Tac-Toe to assess their impact on meta-learning.

Multi-Opponent vs. Fixed Opponent. While the Fixed-Opponent baseline slightly outperforms MAGE against MCTS-100, this advantage is limited to task-specific memorization. When tested against MCTS-1000, MAGE demonstrates superior zero-shot generalization, achieving a 100% success rate faster than the baseline. This confirms that MAGE's multi-opponent training fosters more robust, scalable policies compared to the specialized, brittle strategies developed through single-opponent training.

Varied Distribution. Comparing a *Balanced Distribution* (50% MCTS, 50% patterns/random) to a *Pattern-Skewed* version shows that MAGE (Balanced) outperforms the Skewed variant (67.2% vs. 57.8%). Exposure to diverse, patterned agents acts as a necessary curriculum for building robust opponent models.

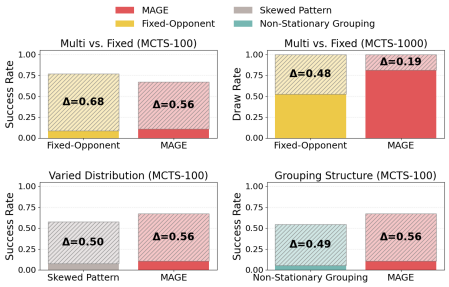

Figure 6: **Opponent diversity ablation results.** Using fixed-opponent, non-stationary grouping or skewed opponent weighting causes success rate to drop in Tic-Tac-Toe.

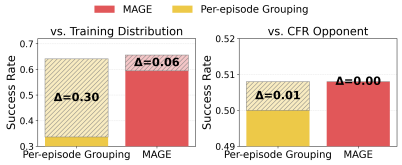

Figure 7: **Advantage normalization ablation results.** Using per-episode grouping causes success rate to drop or fluctuate in Kuhn Poker.

Grouping Structure. We compare *Stationary Grouping* (standard MAGE), where normalization groups share a single opponent archetype, against *Non-Stationary Grouping* (mixed archetypes). Stationary grouping achieves 67.2%, while the non-stationary variant drops to 54.7%. By isolating archetype-specific variance, Stationary Grouping provides a cleaner credit assignment signal, enabling the agent to refine strategies effectively from interaction history.

4.4.3 Agent-specific Advantage Normalization

We evaluate two grouping strategies for GiGPO-style normalization: **Cross-episode grouping (Global Anchor):** Aggregates all actions at state s across the entire meta-episode into a single anchor group, $\mathcal{G}_{\text{global}}(s)$. **Per-episode grouping (Local Anchor):** Normalizes actions at state s only within each individual episode n, $\mathcal{G}_n(s)$. The global anchor captures inter-episode dependencies for long-term improvement, while the local anchor preserves episode-specific context. MAGE (global anchor) outperforms the local variant in Kuhn Poker (Figure 7), starting at 59.4% and rapidly hitting the 65.6% ceiling versus 33.6%. Global normalization stabilizes the baseline, linking early actions to exploitation. While both eventually reach the $\approx 50.8\%$ CFR theoretical limit, MAGE minimizes variance and ensures more consistent policy updates.

5 Conclusion

In this work, we introduced MAGE, a meta-reinforcement learning framework designed to equip Large Language Model agents with the capability for strategic exploration and exploitation in multi-agent environments. By shifting the paradigm from static execution to dynamic adaptation, MAGE enables agents to actively identify and capitalize on opponent vulnerabilities through multi-episode interactions. Our integration of population-based training with agent-specific advantage normalization effectively addresses the challenges of opponent diversity and reward instability, fostering robust "learning-to-learn" behaviors. Empirical results demonstrate that MAGE not only outperforms existing baselines in both exploration and exploitation tasks but also exhibits strong zero-shot generalization against unseen strategies. These findings highlight the necessity of internalizing meta-learning mechanisms within LLMs, paving the way for more autonomous agents capable of navigating the complexities of non-stationary, real-world interactions without relying on external scaffolding.

IMPACT STATEMENT

This paper presents work whose goal is to advance the field of meta-reinforcement learning for language agents. Our framework, MAGE, empowers large language model agents to perform strategic exploitation in multi-agent environments, which could be applied to areas such as adaptive educational tools and complex resource allocation scenarios. Our method may be adapted to other domains requiring rapid adaptation, like human-computer interaction. It is also essential to ensure the responsible deployment of our approach to avoid potential misuse.

REFERENCES

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Wojciech M Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. Real world games look like spinning tops. *Advances in Neural Information Processing Systems*, 33:17443–17454, 2020.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

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

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

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.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Yulun Jiang, Liangze Jiang, Damien Teney, Michael Moor, and Maria Brbic. Meta-rl induces exploration in language agents. *arXiv preprint arXiv:2512.16848*, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

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.

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.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

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*, 2022.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, et al. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*, 2025.

A End-to-end MAGE Algorithm

The end-to-end training procedure of MAGE is summarized in Algorithm 1.

Algorithm 1 End-to-End MAGE Training Process

Require: Initialized LLM policy π_θ, task description x, opponent population $\mathcal{O}$ (if multi-agent), meta-episode length N, episode horizon T, discount factors $\gamma_{\text{step}}, \gamma_{\text{traj}}$.

while training **do**
  Sample an opponent $\phi_m \sim \mathcal{O}$ (or initialize stationary task).
  Initialize **Contextual Memory** $\mathcal{M}_0 = \{m_0\}$ (empty reflection).
  Set initial baseline reward $R(\tau_0) = 0$.
  % Phase 1: Rollout & Memory Assembly
  for episode $n = 1, 2, \ldots, N$ **do**
    Initialize state $s_{n,1}$ and episode trajectory $\tau_n = \emptyset$.
    for step $t = 1, 2, \ldots, T$ **do**
      Sample action $a_{n,t} \sim \pi_\theta(\cdot \mid s_{n,1:t}, \mathcal{M}_{n-1}, x)$.
      Execute $a_{n,t}$, observe next state $s_{n,t+1}$ and task reward $r_{n,t}$.
      Append $(s_{n,t}, a_{n,t}, r_{n,t})$ to τ_n.
    end for
    Receive cumulative task reward $R(\tau_n) = \sum r_{n,t}$.
    Compute episode-wise differential meta-reward: $\mathcal{R}_n = R(\tau_n) - R(\tau_{n-1})$.
    if $n < N$ **then**
      Reflection Construction:
      Generate reflection $m_n \sim \pi_\theta^{\text{refl}}(\cdot \mid \tau_n, x)$.
      Memory Assembly: $\mathcal{M}_n \leftarrow \mathcal{M}_{n-1} \cup \{m_n\}$. ▷ Store self-generated reflection
    end if
  end for
  % Phase 2: Advantage Computation
  for episode $n = 1, \ldots, N$ **do**
    Assign step-wise reward $\tilde{r}_{n,T} = \mathcal{R}_n$ and $\tilde{r}_{n,t} = 0$ for $t < T$.
  end for
  for episode $n = N$ down to 1 **do** ▷ Backward pass for returns
    for step $t = T$ down to 1 **do**
      $G_{n,t} = \sum_{t'=t}^{T} \gamma_{\text{step}}^{t'-t} \tilde{r}_{n,t'} + \sum_{k=n+1}^{N} \gamma_{\text{traj}}^{k-n} G_{k,0}$.
      Compute normalized advantage $\hat{A}_{n,t}^{(m)}$ from $G_{n,t}$.
    end for
    if $n > 1$ **then**
      Set $G_{m_{n-1}} = G_{n,0}$ and compute reflection advantage $\hat{A}_{n,0}^{(m)}$.
    end if
  end for
  % Phase 3: Policy Update
  $\mathcal{L}_{\text{episode}}(\theta) = -\sum_{n,t} \hat{A}_{n,t}^{(m)} \log \pi_\theta(a_{n,t} \mid s_{n,1:t}, \mathcal{M}_{n-1}, x)$
  $\mathcal{L}_{\text{reflection}}(\theta) = -\sum_{n=2}^{N} \hat{A}_{n,0}^{(m)} \log \pi_\theta(m_{n-1} \mid \tau_{n-1}, x)$
  Update θ via gradient descent on $\nabla_\theta \left(\mathcal{L}_{\text{episode}}(\theta) + \mathcal{L}_{\text{reflection}}(\theta)\right)$.
end while

B Training and Evaluation Details

B.1 Shared Training Hyperparameters

Unless otherwise specified, all experiments use the shared hyperparameters listed in Table 2. We use an AdamW optimizer with a constant learning rate and employ GiGPO with mean-normalization for advantage stabilization.

Hyperparameter	Value
Actor Learning Rate	1×10^{-6}
PPO Mini-batch Size	64
PPO Micro-batch Size (per GPU)	8
Log-prob Micro-batch Size (per GPU)	16
Sampling Temperature	0.7
Top-p / Top-k	0.8 / 20
Invalid Action Penalty Coef.	0.5
Step-wise Discount (γ_{step})	0.95
Trajectory Discount (γ_{traj})	0.6
GiGPO Step Advantage Weight	1.0
GiGPO Normalization Mode	`mean_norm`
Total Training Epochs	150
Evaluation Seed	0

Table 2: Shared training and rollout hyperparameters across all environments.

B.2 Environment-Specific Configurations

Task-specific constraints and architectural parameters are detailed below. For all tasks, the maximum prompt and response lengths during testing remain consistent with training unless otherwise noted.

AlfWorld The environment uses `alfworld/AlfredTWEnv` with the standard `train` split for optimization. In-domain and out-of-domain evaluations utilize the `eval_in_distribution` and `eval_out_of_distribution` sets, respectively. Parameters include 10 maximum turns, a maximum prompt length of 4096, a maximum response length of 1024, a maximum of 16384 batched tokens, and a reference log-probability micro-batch size of 16.

Sokoban Training is conducted over 300 epochs on 6×6 rooms with two boxes. Constraints include a maximum of 7 turns, a search depth of 100, a maximum of 21 solution steps, and 3 actions per turn. The configuration uses a maximum prompt length of 4096, a maximum response length of 4096, 32768 maximum batched tokens, and a length penalty coefficient of 1.0.

WebShop Optimization is performed over 150 epochs with a maximum of 12 turns, a maximum prompt length of 8192, a maximum response length of 1024, 32768 maximum batched tokens, and a reference log-probability micro-batch size of 32. For out-of-domain evaluation, the maximum prompt length is extended to 10240 tokens.

Tic-Tac-Toe This environment utilizes KL-divergence regularization (coefficient 0.1 with the `low_var_kl` variant) and a length penalty coefficient of 2.0. The setup includes 150 training epochs, a maximum of 8 turns, a 3×3 board, a maximum prompt length of 4096, a maximum response length of 3072, and 16384 maximum batched tokens.

Kuhn Poker Training consists of 150 epochs with a maximum of 6 turns. Parameters include a maximum prompt length of 4096, a maximum response length of 4096, 16384 maximum batched tokens, and a length penalty coefficient of 2.0.

C Prompt Templates

This section provides the full prompt templates used for the decision-making (Play) and the reflection (Reflect) stages across all evaluated environments. Placeholders such as {`init_observation`} and {`current_trajectory`} are dynamically populated during the interaction.

C.1 ALFWORLD PROMPTS

ALFWorld Play Prompt

```
You are an expert agent operating in the ALFRED Embodied
Environment.
{init_observation}{past_trajectories_reflections}{current_trajectory}
Your admissible actions of the current situation are:
[{admissible_actions}]
Now it's your turn to take an action.
- Your response should first by step-by-step reasoning about the
current situation.
- Once you've finished your reasoning, you should choose an
admissible action for current step and present it within <action>
</action> tags.
```

ALFWorld Reflect Prompt

```
You are an expert agent operating in the ALFRED Embodied
Environment.
{init_observation}
You will be given the history of a past experience.  Your job is to
**reflect on the past sequence**, identify any **mistakes or
inefficiencies**, and then devise a **concise, improved plan**
starting from the original initial state.
Below are the actions you took and the corresponding observations:
{current_trajectory}
The task is NOT successfully completed.
Now it's your turn to reflect on the past experience and come up
with a new plan of action.
- Your response should first be step-by-step reasoning about the
strategy and path you took to attempt to complete the task.
Identify where things went wrong or could be better.
- Then devise a concise, new plan of action that accounts for your
mistake with reference to specific actions that you should have
taken.
- Finally, end the response with your reflection and improved plan
inside <remark> </remark> tags, to guide the next trial.
```

C.2 SOKOBAN PROMPTS

Sokoban Play Prompt

```
You are an expert agent operating in the Sokoban environment.
# Symbols and Their Meaning
- Walls (#):  These block movement.  You can't move through or push
anything into walls.
- Floor (_):  Open spaces where you can walk and move boxes.
- Targets (O): The spots where boxes need to go.
- Boxes (X): These are what you need to push onto the targets.
- Player (P): That's you!  You'll move around the grid to push
boxes.
- Box on Target (✓):  A box successfully placed on a target.
- Player on Target (S): You standing on a target.
# Goal
```

```
Your goal is to push all the boxes (X) onto the target spots (O).
Once all boxes are on the targets, you win!
# Rules
Your admissible actions are ["up", "down", "left", "right"].
You can only push one box at a time.  You can't pull boxes, so plan
ahead to avoid getting stuck.
You can't walk through or push boxes into walls (#) or other boxes.
To avoid traps, do not push boxes into corners or against walls
where they can't be moved again.
# Observations
The initial state of the game is:
{init_observation}{past_trajectories_reflections}{current_trajectory}
Now it's your turn to make moves (choose the next
{num_actions_per_turn} actions).
- Your response first be step-by-step reasoning about the current
situation | observe the positions of boxes and targets, plan a path
to push a box toward a target, and avoid traps like corners or
walls.
- Then choose {num_actions_per_turn} admissible actions and present
them within <action> </action> tags (separated by comma).
```

C.3 WebShop Prompts

WebShop Play Prompt

```
You are an expert autonomous agent operating in the WebShop
e-commerce environment.
Your task is to:
{task_description}.{past_trajectories_reflections}{current_trajectory}
Your admissible actions of the current situation are:
[ {admissible_actions} ].
Now it's your turn to take one action for the current step.
Your response should first be step-by-step reasoning about the
current situation, then think carefully which admissible action best
advances the shopping goal.
Once you've finished your reasoning, you should choose an admissible
action for current step and present it within <action> </action>
tags.
```

C.4 Tic-Tac-Toe Prompts

Tic-Tac-Toe Play Prompt

```
You are an expert agent playing Tic-Tac-Toe on a {board_size} by
{board_size} board.
The rows and columns are indexed from 1 to {board_size}.
# Cell States
- Empty cells (.):  cells that are not yet taken
- Player X (X): cells taken by player X
- Player O (O): cells taken by player O
# Game Rules
- A valid action is placing your mark on an empty cell (.).
- Each action is represented as a coordinate (row, col).
- At the INITIAL state, all of the following actions are valid:
(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)
```

```
- A player WINS if they place THREE marks consecutively in a
straight line (ROW, COLUMN, or DIAGONAL).
# Your Goal
Your goal is to win or prevent the opponent from winning.  You play
as {player_symbol}, opponent plays as {opponent_symbol}.
# Observation
{init_observation}{past_trajectories_reflections}{current_trajectory}
IMPORTANT: The action history contains ONLY YOUR actions.  Opponent
acts automatically.
- Reason step-by-step about threats/opportunities.
- Choose ONE EMPTY cell (.)  and put the index in "(row, col)"
format within <action> </action> tags.
```

C.5 KUHN POKER PROMPTS

Kuhn Poker Play Prompt

```
You are an expert Kuhn Poker agent.
# Game Rules
- Deck includes:  King (K) > Queen (Q) > Jack (J).
- Both players place 1 chip blind ante.  Each is dealt a private
card.
- Players choose:  PASS or BET (1 additional chip).
- If a player PASSES after a BET, the opponent wins.  If both PASS
or both BET, the higher card wins.
# Player Information
- You are playing as Player {agent_player_id}, opponent is Player
{opponent_player_id}.
# Your Goal
Maximize your total chips over the long run.
# Observation
{init_observation}{past_trajectories_reflections}{current_trajectory}
Now it's your turn:
- Reason step-by-step about the current situation.
- Choose ONE action:  PASS or BET inside <action> </action> tags.
```

D COMPUTATIONAL EFFICIENCY AND TRADE-OFF ANALYSIS

Method	WebShop	Tic-Tac-Toe	Kuhn Poker
GRPO	14:16:34	29:08:19	25:34:26
GiGPO	16:10:31	28:00:52	25:10:46
LaMER	26:39:55	33:21:48	23:45:09
MAGE	26:14:13	31:56:03	25:02:03

Table 3: Total wall-clock training time (hh:mm:ss) on 8× H200 GPUs.

To investigate the relationship between computational overhead and performance gains, we report the total wall-clock training time for MAGE and relevant baselines in Table 3. All models were trained on a single node with 8× NVIDIA H200 GPUs.

Our analysis reveals several key insights regarding the efficiency of the MAGE framework:

- **Minimal Overhead in Short-Trajectory Tasks:** In environments with shorter trajectory lengths, such as *Kuhn Poker* and *Tic-Tac-Toe*, the additional compute required for MAGE is marginal.

- **Significant Performance ROI in Complex Tasks:** For interaction-heavy environments like *WebShop*, the wall-clock time for MAGE (approx. 26 hours) is higher than the GiGPO baseline (approx. 16 hours). However, this additional time is justified by the substantial performance breakthrough, where the success rate improves from 0.797 (GiGPO) to 1.0 (MAGE). This represents a major qualitative leap in agent capability that standard RL fails to achieve regardless of training duration.

In conclusion, while the reflective stage introduces a non-negligible time cost in long-horizon tasks, the results demonstrate that this cost is well-compensated by the agent's improved ability to exploit past experiences and avoid repetitive strategic errors.

E EXPERIMENTS

E.1 META-EPISODE LENGTH ABLATION

Method	Ep0	Ep1	Ep2	Ep3	Ep4	Ep5	Ep6	Ep7	Ep8	Ep9
MAGE_3	0.109	0.211	0.289	0.352	0.438	0.492	0.523	0.562	0.586	0.672
MAGE_4	0.086	0.180	0.258	0.305	0.336	0.414	0.500	0.547	0.578	0.617
MAGE_5	0.203	0.367	0.453	0.516	0.562	0.625	0.672	0.727	0.766	**0.820**

Table 4: Performance under different meta-episode lengths. MAGE_n denotes the configuration where the meta-episode length is set to n.

The experimental results in Table 4 demonstrate that the MAGE framework exhibits strong scalability and robustness across different meta-episode horizons. Although the model is trained with a meta-episode length of 3, it maintains a consistent upward performance trajectory when evaluated at longer lengths. Notably, MAGE_5 consistently outperforms MAGE_3 and MAGE_4 across all episodes, achieving the highest success rate of 0.820 by the final episode. This indicates that the framework effectively leverages extended meta-episode to refine its strategies, confirming its stability and generalization capability over longer horizons than those encountered during training.

E.2 PERFORMANCE ANALYSIS ON QWEN3-8B

Method	Ep0	Ep1	Ep2	Ep3	Ep4	Ep5	Ep6	Ep7	Ep8	Ep9
ReAct	$0.011_{\pm.004}$	$0.013_{\pm.004}$	$0.013_{\pm.004}$	$0.021_{\pm.004}$	$0.021_{\pm.004}$	$0.028_{\pm.004}$	$0.042_{\pm.004}$	$0.042_{\pm.004}$	$0.050_{\pm.008}$	$0.050_{\pm.008}$
Reflexion	$0.008_{\pm.004}$	$0.078_{\pm.008}$	$0.161_{\pm.007}$	$0.250_{\pm.010}$	$0.309_{\pm.004}$	$0.354_{\pm.010}$	$0.427_{\pm.018}$	$0.456_{\pm.015}$	$0.508_{\pm.010}$	$0.542_{\pm.004}$
A-MEM	$0.003_{\pm.004}$	$0.003_{\pm.004}$	$0.003_{\pm.004}$	$0.003_{\pm.004}$	$0.003_{\pm.004}$	$0.005_{\pm.008}$	$0.008_{\pm.012}$	$0.016_{\pm.012}$	$0.018_{\pm.009}$	$0.023_{\pm.006}$
GiGPO	$0.185_{\pm.024}$	$0.185_{\pm.030}$	$0.279_{\pm.028}$	$0.320_{\pm.043}$	$0.385_{\pm.024}$	$0.404_{\pm.029}$	$0.447_{\pm.004}$	$0.453_{\pm.008}$	$0.490_{\pm.019}$	$0.500_{\pm.029}$
LAMER	$0.250_{\pm.017}$	$0.430_{\pm.022}$	$0.529_{\pm.018}$	$0.633_{\pm.007}$	$0.698_{\pm.010}$	$0.768_{\pm.003}$	$0.799_{\pm.009}$	$0.833_{\pm.017}$	$0.872_{\pm.008}$	$0.886_{\pm.008}$
MAGE	$0.193_{\pm.007}$	$0.346_{\pm.016}$	$0.471_{\pm.013}$	$0.578_{\pm.017}$	$0.656_{\pm.028}$	$0.685_{\pm.023}$	$0.719_{\pm.014}$	$0.776_{\pm.004}$	$0.823_{\pm.026}$	$0.854_{\pm.016}$

Table 5: Performance on TicTacToe using Qwen3-8B across three random seeds. Results are reported as mean $\pm$ standard deviation.

The results presented in Table 5 summarize the performance of MAGE and various baselines on the Tic-Tac-Toe task using the Qwen3-8B model across three random seeds. Our analysis reveals several key insights:

- **Leading Performance:** Both LAMER and MAGE emerge as the superior methods, significantly outperforming other baselines. MAGE demonstrates a robust learning curve, ascending from an initial success rate of 0.193 ± 0.007 to a final performance of 0.854 ± 0.016. In contrast, baselines such as ReAct and A-MEM fail to achieve significant traction on this task, with success rates remaining below 0.06. While Reflexion and GiGPO show visible improvement over time, they plateaus at approximately 0.542 and 0.500 respectively, trailing behind MAGE by a substantial margin.
- **Scalability Potential:** Although the 8B model already provides a strong signal for the efficacy of the MAGE framework, the consistent performance gains observed here provide a promising justification for future evaluations on larger-scale models.

Reflection	Agent Response
Reflection summarizing errors and strategic guidance. The **key mistakes** in the past trial were: 1. Invalid moves (e.g., (-1,-1)) that disrupted the game state. 2. Poor initial strategy ... **Improved Plan:** Start with the center (2,2) to maximize control and opportunities for winning. ...	*Action generated by the agent after internalizing the reflection.* </think> ... **The improved plan says** to start with the center (2,2) because that's the best move for maximizing control and opportunities. So first move should be (2,2). ... </think>

Table 6: Qualitative behavior analysis in Tic-Tac-Toe.

Agent Response
The agent explicitly considers multiple plausible opponent strategies rather than assuming a fixed policy, and selects an action that is robust under strategic uncertainty. </think> ... So, what's the possible scenario here? Case 1: Player 0 has King. ... Case 2: Player 0 has Jack. ... So, in this case, **it's better to bet**. ... **Wait, but that's not necessarily the case.** If Player 0 has a King, then they have a strong hand. If they bet, that's a strong move. But if Player 0 has a Jack, then **they might bet as well**. ... </think>

Table 7: Qualitative behavior analysis in Kuhn Poker illustrating multi-policy reasoning.

F Mechanistic Behavior Analysis

To understand how MAGE shapes decision-making, we qualitatively analyze representative responses focusing on reflection-based learning and multi-opponent policy recognition.

Learn-to-Learn from Reflection. To provide qualitative evidence that MAGE explicitly trains the agent to learn to learn rather than relying on emergent in-context learning, we analyze a representative interaction from the Tic-Tac-Toe environment.

As shown in Table 6, after an episode of errors (e.g., invalid actions, poor positioning), the agent consolidates these failures into a structured reflection. Unlike passive memory, MAGE optimizes the policy to exploit this feedback. In the subsequent episode, the agent demonstrates causally grounded correction: it reasons about prior mistakes, validates the action space, and selects the optimal center opening (2,2), translating reflection into immediate behavioral improvement.

Multi-opponent Policy Recognition and Generalization. Beyond learning from reflection, MAGE enables agents to adapt to environments involving strategic opponents with heterogeneous behaviors. This capability is particularly critical in multi-agent settings, where optimal actions depend not only on the current state but also on opponent policies.

In Kuhn Poker (Table 7), when facing a bet while holding a Queen, the agent does not assume a fixed adversarial strategy. Instead, it reasons over diverse opponent profiles—from aggressive to conservative—evaluating outcomes under multiple scenarios before acting. This behavior demonstrates internalized opponent modeling that is robust to policy variation rather than reliant on a single assumed equilibrium.

G Qualitative Analysis of Exploration and Policy Convergence

To further investigate the internal dynamics of policy evolution, we visualize the state-action space across episodes in a 3D manifold (X: Episode, Y: State Index, Z: Action Index). The diameter of each sphere represents the frequency of a specific state-action pair (s, a) within an episode.

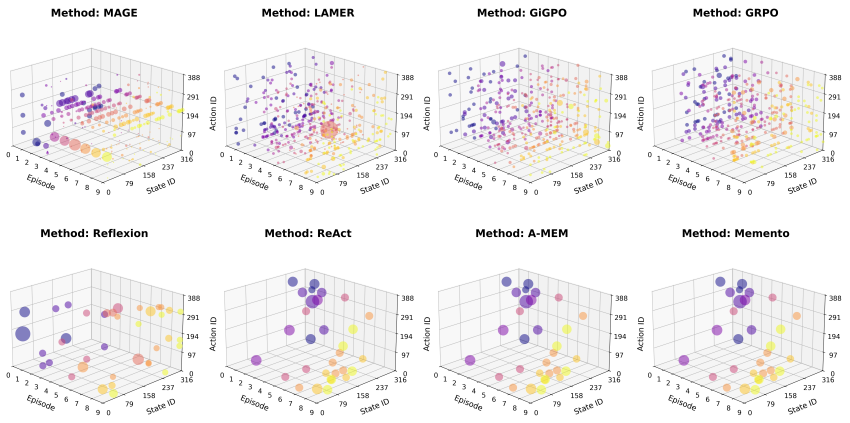

Figure 8: 3D visualization of the state-action manifold across episodes for the WebShop task. The axes represent the Episode index (X), Shuffled State Index (Y), and Action Index (Z), with sphere size indicating the relative frequency of each (s, a) pair.

G.1 WEBSHOP

In the WebShop task, which requires complex multi-step reasoning and environment interaction, we observe distinct behavioral patterns (Figure 8).

MAGE exhibits a transition from broad exploration to structured exploitation. From $Episode \approx 4$, the state-action distribution becomes highly stabilized and regular. The spheres are distributed with remarkable homogeneity, suggesting that MAGE has identified a robust and repeatable trajectory. This visual regularity corresponds to its superior performance, where the success rate reaches 100.0% at $Episode\ 4$ and maintains perfect execution thereafter. LAMER, GiGPO, GRPO methods show a persistently scattered distribution throughout the training process. The lack of concentrated "strategy tunnels" in the 3D space indicates that these agents fail to converge on an optimal path, resulting in suboptimal success rates. The visualizations for Reflexion, ReAct, A-MEM, Memento methods are dominated by a few static, large circles with almost no surrounding exploration. This reflects a "frozen" policy that repeats fixed actions regardless of environmental feedback. Consequently, they suffer from premature stagnation, with success rates often flatlining near zero.

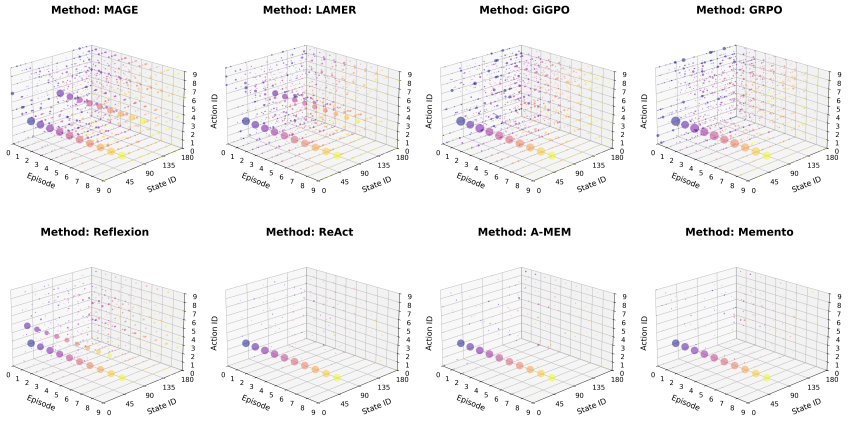

Figure 9: 3D visualization of the state-action manifold across episodes for the Tic-Tac-Toe task. The axes represent the Episode index (X), Shuffled State Index (Y), and Action Index (Z), with sphere size indicating the relative frequency of each (s, a) pair.

G.2 TIC-TAC-TOE

The Tic-Tac-Toe task highlights the agent's ability to balance tactical focus with state-space coverage (Figure 9.

MAGE demonstrates a unique "dual-core" concentration, with two prominent state-action clusters representing key tactical responses. Crucially, these are accompanied by a wide and regular distribution of smaller spheres, indicating a healthy level of auxiliary exploration. This balanced profile allows MAGE to achieve the highest terminal success rate of 67.2%. LAMER shows a similar dual-cluster pattern but with a smaller secondary core compared to MAGE, leading to a slightly lower 60.2%. Reflexion also displays two clusters, yet its peripheral exploration (small dots) is significantly sparser, suggesting it lacks the necessary search breadth to handle diverse opponent moves. GiGPO, GRPO methods converge on only a single state-action cluster. While they show wide exploration, the dots are distributed irregularly, lacking the structured "policy lines" seen in MAGE. This implies that while they search the space, they fail to synthesize this experience into a coherent, multi-faceted strategy. ReAct, A-MEM, Memento methods exhibit a singular large cluster with virtually no visible secondary points. Such extreme sparsity indicates a failure to explore alternative states, leading to near-zero success rates.

G.3 SUMMARY OF FINDINGS

The 3D visualization confirms that MAGE's advantage stems from its ability to rapidly stabilize its core policy (observed as consistent "tunnels" along the episode axis) and maintain structured exploration (observed as the wide, regular distribution of smaller spheres), which prevents the rigid stagnation seen in baselines like ReAct or Reflexion.

H LIMITATIONS AND FUTURE WORK

Although MAGE establishes a robust foundation for in-context adaptation, several promising directions remain for future exploration. While we focus on text-based environments, integrating multimodal feedback represents a natural evolution for cross-domain plasticity. Additionally, exploring dynamic, co-evolutionary training regimes where the opponent population evolves in response to the agent's progress could yield even more sophisticated strategic behaviors. Finally, while MAGE excels in discrete strategic tasks, evaluating its performance in open-ended, real-world environments with high-dimensional action spaces remains an important next step for verifying its broader applicability.

I USE OF LLMS

We acknowledge the use of large language models (LLMs) to assist in the preparation of this manuscript. Specifically, LLMs are employed to improve the conciseness of the technical descriptions, ensure consistent LaTeX formatting across sections, and refine the grammatical flow of the analysis.