MEMORY-DRIVEN SELF-IMPROVING LLM AGENTS FOR SEQUENTIAL DECISION-MAKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have emerged as effective action policies for sequential decision-making (SDM) tasks due to their extensive prior knowledge. However, this broad yet general knowledge is often insufficient for long-term decision-making tasks with limited task-related experience, making it challenging to efficiently adapt LLMs to specific SDM tasks especially for generalizing to unseen tasks. To address this challenge, we propose a *memory-driven self-improvement framework for LLM agents* that combines the general prior knowledge of LLMs with a compact memory of domain-specific experiences. The framework consists of two mutually reinforcing components: memory-driven value estimation and LLM prior refinement. Specifically, the memory stores state-action experiences with Q-values grounded in environmental feedback, enabling retrieval-based, non-parametric value estimation while providing valuable experiences for refining the LLM prior. The refined LLM prior, in turn, generates higher-quality trajectories that further enrich the memory, forming a self-improvement loop. Experiments on multi-step agentic and mathematical reasoning tasks demonstrate that our approach improves sample efficiency and generalization to unseen tasks, substantially outperforming traditional RL and LLM-based baselines.

1 INTRODUCTION

Sequential decision-making (SDM) has a wide range of real-world applications, including robotics Polydoros and Nalpantidis (2017); Brunke et al. (2022); Rana et al. (2023), autonomous driving Naranjo et al. (2005); Song et al. (2022), and human–AI interaction Granter et al. (2017); Li et al. (2019); McTear (2022). Natural language plays a crucial role in many SDM tasks, either in purely language-based settings Granter et al. (2017); Jannala (2025); Jin et al. (2024) or as a tool for understanding and describing the environment Ma et al. (2024a); Wang et al. (2025). Large language models (LLMs), with their broad prior knowledge, demonstrate strong zero-shot reasoning capabilities, making them promising candidates for action policies in such text-based SDM tasks. However, when deployed in specialized domains, their general knowledge is often insufficient for reliable decision-making Yan et al. (2025); Jannala (2025).

To adapt LLM action policies into the target domains, three main approaches have been explored. The first is prompt-based methods, which utilize human-crafted prompts Yao et al. (2022; 2024) or incorporate historical interactions Shinn et al. (2024); Christianos et al. (2023) to provide more task-specific information. However, these prompt-engineering methods heavily depend on the quality of the prompts and the reasoning capabilities of the LLMs. The second approach is fine-tuning, which includes supervised fine-tuning (SFT) and reinforcement learning fine-tuning (RLFT). SFT typically requires substantial high-quality decision-making data Zhou et al. (2024), while on-policy RLFT methods Carta et al. (2023); Tan et al. (2024) suffer from poor sample efficiency Abdolmaleki et al. (2018); Chen et al. (2023). The third pipeline performs RL with fixed LLM priors, using LLMs either to narrow the action search space Yan et al. (2025) or to design reward functions Kwon et al. (2023); Klissarov et al. (2023) that promote efficient exploration. However, these methods remain highly sensitive to the capability of the LLM priors in fulfilling such roles.

Considering these limitations, we propose a **memory-driven self-improvement** framework for text-based SDM. To combine the benefits of LLM general knowledge with task-specific interactions, a memory-driven action policy with an LLM prior is designed, where the LLM prior generates action

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

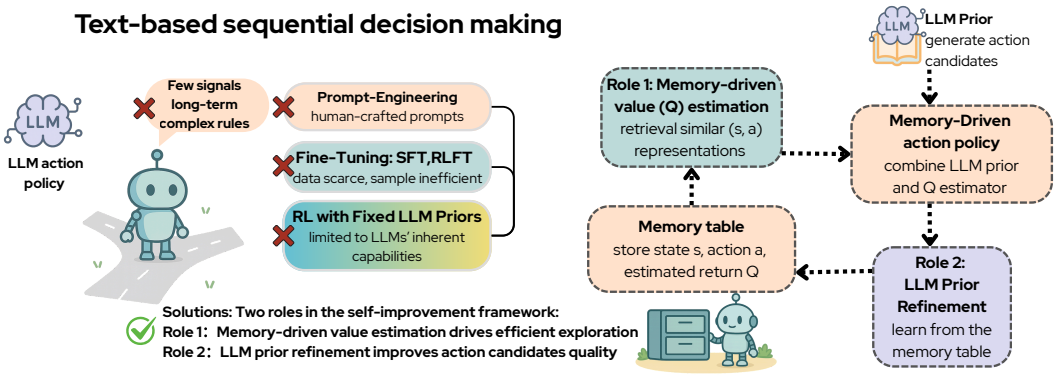


Figure 1: Motivation and overview of our memory-driven self-improvement framework for SDM. Left: existing approaches (prompt-engineering, fine-tuning, and RL with LLM priors) struggle under sparse signals and domain-specific data. Right: Our framework introduces two complementary roles: (1) memory-driven value estimation, which enables efficient exploration, and (2) LLM prior refinement, which biases action generation toward high-quality candidates; together forming a self-improvement loop that resists scarce experience and enables efficient adaptation.

candidates, and memory-driven value estimation guides more precise action posterior selection. In practice, the framework forms a closed loop with two mutually reinforcing roles as illustrated in Figure 1: **Role 1**: **Memory-driven value estimation**, which converts informative interactions into a compact memory table. By retrieving semantically similar past experiences, the model can make non-parametric value estimates for action candidates and enable informed exploration choices. Importantly, the accumulated state-action pairs and their values also provide a compact, value-informed data source for refining the LLM prior. **Role 2**: **Memory-driven LLM prior refinement**, which periodically updates the LLM's decision prior using historical state-action pairs and their Q-values stored in memory. By emphasizing high-value state-action pairs, the refinement biases the LLM prior toward generating higher-quality action candidates, improving both learning efficiency and generalization. Overall, memory supports both Q-value evaluation and policy refinement, while the refined LLM prior generates better actions that further enrich the memory, forming a self-improvement loop for efficient adaptation to target SDM tasks.

In summary, our main contributions are as follows:

1. We propose a **memory-driven self-improvement framework** for text-based SDMs and leverage the Expectation-Maximization (EM) to provide a unified formulation and practical implementation.

2. We develop a **multi-faceted memory utilization mechanism** that associates state-action experiences with Q-values and reuses them for three purposes: non-parametric Q-value estimation for action selection, policy learning toward high-value actions, and Q-guided prompt selection for lightweight, instant prior refinement without parameter updates.

3. Experiments on agentic tasks (ALFWorld, Overcooked, and VirtualHome) and multi-step reasoning tasks (Game of 24 and MiniF2F) demonstrate substantial improvements in sample efficiency and policy performance. Moreover, memory-driven prior refinement improves generalization to unseen tasks and enables experience transfer across LLMs of different scales.

2 PRELIMINARY

Textual Markov Decision Processes. A Markov Decision Process (MDP) is defined as $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1]$ is the discount factor. Particularly, note that r_t denotes the reward received at time step t. In this work, we focus on textual MDP, where both states and actions are represented in natural language, i.e., $\mathcal{S}, \mathcal{A} \subseteq \mathcal{V}^*$, with $\mathcal{V}$ denoting the vocabulary. Textual MDPs present unique challenges, as the state and action spaces can be combinatorially large, and policies must operate over inherently discrete, structured, and semantically rich representations of language.

Q-learning for MDPs A prominent paradigm for solving MDPs with discrete action spaces is Q-learning, where the Q-function $Q(s, a)$ is learned to rank and select actions. For each state–action pair (s, a), the Q-function, defined as $Q(s, a) = \mathbb{E}_\pi \left[\sum_{i \geq t} \gamma^{i-t} r_i \mid s, a\right]$, represents the expected cumulative reward obtained by starting from (s, a) and following policy π thereafter. The DQN algorithm (Mnih, 2013) is a classic instance of Q-learning that maps state–action embeddings to scalar Q-values and trains the Q-network via temporal-difference (TD) learning. Blundell et al. (2016) introduces Episodic Control(EC), a memory-based Q-learning method that leverages retrieval techniques and informative state representations to enable non-parametric Q-value estimation.

Control as Inference. Control-as-Inference framework (Levine, 2018) formulates the solution of MDPs from a probabilistic inference perspective. An optimality random variable $\mathcal{O}$ is introduced, where $\mathcal{O} = 1$ indicates achieving a successful trajectory and $\mathcal{O} = 0$ otherwise. The objective is to maximize the likelihood of achieving a successful trajectory from each state s, which is identical to maximizing the evidence lower bound (ELBO):

$$\log p(\mathcal{O}=1|s) \geq \mathbb{E}_{q(a|s)}[\log p(\mathcal{O}=1|s,a)] - D_{\mathrm{KL}}(q(a|s)\|p(a|s)) \triangleq \mathrm{ELBO}, \tag{1}$$

where $p(a|s)$ is the prior action distribution, $q(a|s)$ is the variational distribution, and $p(\mathcal{O} = 1|s, a)$ is the likelihood that the trajectory will achieve optimality given the current state-action pair (s, a). Abdolmaleki et al. (2018) assume that the optimality likelihood is proportional to Q-value: $p(\mathcal{O} = 1|s, a) \propto \exp(Q(s, a)/\tau)$. This probabilistic formulation offers both conceptual flexibility and methodological richness, while also naturally accommodating the integration of LLMs as sources of prior knowledge (Yan et al., 2025).

3 Memory-Driven value estimation with LLM priors

In this section, we present a memory-driven approach to Q-value estimation that leverages LLM-based semantic representations. By explicitly exploiting LLM representations through retrieval techniques, our method builds upon the well-established episodic control (EC) (Blundell et al., 2016) to enable non-parametric Q-value estimation. Specifically, our method maintains a memory table $\mathcal{M}$ storing Q-values for visited state–action pairs, which is continuously updated during online exploration. At inference time, actions are selected through memory queries. The framework is thus defined by two core operations: **memory update** and **memory query**.

Memory update. A memory table $\mathcal{M}$ is constructed to store information about previously visited state-action pairs (s, a). Specifically, it includes the natural language descriptions of (s, a), the corresponding vectorized embeddings $f(s, a)$, and the associated Q-values $Q(s, a)$. The Q-values stored in memory $\mathcal{M}$ are then updated according to:

$$Q(s_t, a_t) \leftarrow \begin{cases} R_t, & \text{if } (s_t, a_t) \notin \mathcal{M}, \\ \max\{Q(s_t, a_t), R_t\}, & \text{otherwise,} \end{cases} \tag{2}$$

where $R_t = \sum_{i \geq t} \gamma^{i-t} r_i$ represents a Monte Carlo estimate of the cumulative discounted reward (i.e., the return-to-go).

Algorithm 1 Memory-Driven Q-learning (Mem-Q)

Input: Embedding function f_{LLM}, empty memory table $\mathcal{M} = \emptyset$.
Output: Updated memory table $\mathcal{M}$.

for each episode **do**
  for $t = 1, 2, 3, \ldots, T$ **do**
    Obtain all feasible actions $\mathcal{A}_{s_t}$ for current s_t.t
    Estimate the return $\widehat{Q}$ for each $a \in \mathcal{A}_{s_t}$ via Eq. 3.
    Select action a_t using the ϵ-greedy strategy.
    Execute action a_t, receive reward r_{t+1}, and observe the next state s_{t+1}.
  end for
  for $t = T, T-1, \ldots, 1$ **do**
    Write and update memory table $\mathcal{M}$ via Eq. 2.
  end for
end for

Memory query. Given the memory $\mathcal{M}$ and the current state s_t, the policy is then determined using the kernel-based Q-value estimator $\widehat{Q}$, which is defined as

$$\widehat{Q}(s_t, a) = \sum_{i \in \mathcal{N}_M(f(s_t,a))} w_i Q(s^{(i)}, a^{(i)}), \quad w_i = \frac{k(h, h_i)}{\sum_{j \in \mathcal{N}_M} k(h, h_j)}, h = f(s_t, a), h_i = f(s^{(i)}, a^{(i)}), \tag{3}$$

where $\mathcal{N}_M(f(s_t, a))$ denotes the M nearest neighbors of $f(s_t, a)$, and the inverse distance kernel is used with $k(h, h_j) = \frac{1}{\|h - h_j\|_2^2 + \delta}$, which measures similarity in the embedding space. With the estimated Q-values $\widehat{Q}$, action selection can be carried out using the ϵ-greedy strategy, analogous to the approach in DQN (Mnih, 2013).

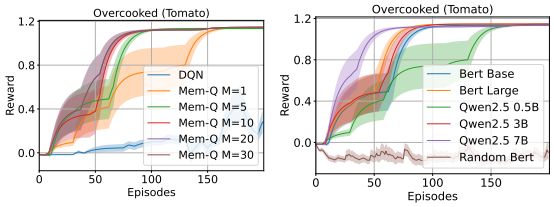


Figure 2: Results of memory-drive Q-learning on Overcooked. Left: effect of the number of retrieved (s, a) pairs for value estimation; Right: effect of different LLMs on representations.

Vectorized Memory Query with LLM Priors. While conceptually simple, the effectiveness of the proposed memory-driven Q-learning critically depends on the choice of embedding function f. Inspired by the recent success of Retrieval-Augmented Generation (RAG) (Wiratunga et al., 2024), which demonstrates the ability of LLMs to yield semantically rich vector representations. Hence, we adopt the LLM encoder as the embedding function f_{LLM}. The overall memory-driven decision-making procedure with LLM embeddings is summarized in Algorithm 1, referred to as Memory-Driven Q-learning (Mem-Q).

In Figure 2, we evaluate Mem-Q on Overcooked (Tan et al., 2024), a textual decision-making task. The results show that Mem-Q significantly outperforms DQN. The ablation study on the retrieval size M, introduced in Eq. (3), shows that incorporating more similar (s, a) embeddings into kernel-based Q estimation leads to faster convergence, which may be attributed to more accurate value estimation with semantically meaningful representations (Han et al., 2023). Furthermore, larger LLMs, which capture richer semantic structures, consistently yield stronger performance. Mem-Q applies directly to predefined, enumerable action spaces, but becomes intractable for free-form actions. We therefore introduce generative LLMs to propose candidate actions. Beyond value estimation, the state-action memory with Q-values also provides a natural foundation for action policy learning and prompt example selection, as introduced in the next section.

4 Memory-Driven Policy Optimization with LLM Priors

The success of memory-driven Q-learning highlights a key insight: combining the capabilities of LLMs with experience storage can substantially enhance the efficiency of RL algorithms. Motivated by this, we propose a **memory-driven action policy** that uses the LLM prior to narrow the action search space, and leverage memory to further refine the LLM prior with domain-specific knowledge, thereby improving sample efficiency.

4.1 Memory-Driven Policy with LLM Priors

A straightforward approach to incorporating LLMs into the action policy is through a probabilistic inference framework (Li et al., 2024), wherein the policy is cast as the posterior distribution:

$$p_\theta(a|s, \mathcal{O} = 1) \propto p(\mathcal{O} = 1|s, a)p_{\text{LLM}_\theta}(a|s), \tag{4}$$

where $p_{\text{LLM}_\theta}(a|s)$ is the LLM prior parametrized by θ and $p(\mathcal{O} = 1|s, a)$ denotes the likelihood of optimality[1], which can be estimated using Q-value, as described in Eq 3. Therefore, the memory-driven policy can be conducted using self-normalized importance sampling as follows:

- Sample K candidate actions from the LLM prior: $\mathcal{C}^K(s) = \{a_1, \ldots, a_K\}, a_k \sim p_{\text{LLM}_\theta}(\cdot|s)$.
- For each action candidate a_k, approximate the optimality likelihood using the kernal-based Q-value estimator $\widehat{Q}(s, a_i)$ according to Eq 3.
- Select an action via: $a \sim \text{Multinormial}\left(\exp(\widehat{Q}(s, a_1)/\tau)/\sum_k \exp(\widehat{Q}(s, a_k)/\tau)\right)$.

Although conceptually straightforward, this method is sensitive to the specification of the LLM prior p_θ. A poorly aligned prior may bias the candidate set toward suboptimal actions, thereby degrading policy performance. To overcome this limitation, the following section introduces a principled optimization procedure based on the Expectation–Maximization framework, which refines the memory-driven policy by adaptively optimizing the underlying LLM priors.

[1]Notably, the likelihood exhibits no explicit parametric dependence, as the proposed memory-driven Q-learning framework is inherently non-parametric.

4.2 Optimizing Memory-Driven Policy with Expectation Maximization

To optimize the memory-driven policy with the LLM prior and the likelihood estimation involved, we adopt the Expectation–Maximization (EM) algorithm. Starting from an arbitrary initialization θ_0, the EM procedure performs the following iterative update:

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{p_{\theta_k}(a,s|\mathcal{O}=1)} \left[\log p(\mathcal{O}=1|s,a) + \log p_{\text{LLM}_\theta}(a|s)\right]. \tag{5}$$

By construction, the EM update can be interpreted as maximizing the ELBO in Eq. 1. In practice, however, the expectation in Eq. 5 is analytically intractable. To address this challenge, we employ an on-policy E-step for likelihood expectation approximation and a memory-driven off-policy E-step for prior expectation approximation, followed by maximization based on these tractable estimates.

4.2.1 Expectation Step.

In the E-step, a simple yet tractable approach for the expectation approximation is the Monte Carlo (MC) estimation. Concretely, for the current state s, we draw an action from the posterior $a \sim p_\theta(a|s, \mathcal{O}=1)$, which can be approximated using importance sampling introduced in Sec. 4.1. For the likelihood expectation term in Eq. 5, the likelihood is assumed to be proportional to the Q value, and one can directly apply the kernel-based Q value estimation in Eq. 3.

However, for the LLM prior expectation term, such an "on-policy" approach is inefficient, since LLMs typically involve a very large number of parameters, and a limited set of MC samples is insufficient to provide reliable gradient estimates for the subsequent M-step. Consequently, we consider importance sampling approximation using examples stored in the memory table to explore and exploit the posterior distribution effectively, thereby yielding a robust "off-policy" estimate of the expectation. Specifically, we employ self-normalized importance sampling (SNIS) to estimate the LLM prior expectation as follows:

$$\begin{aligned}\mathbb{E}_{p_\theta(a,s|\mathcal{O}=1)} \left[\log p_{\text{LLM}_\theta}(a|s)\right] &= \mathbb{E}_{q(s,a)} \left[\frac{p_\theta(a,s|\mathcal{O}=1)}{q(s,a)} \log p_{\text{LLM}_\theta}(a|s)\right] \\ &\approx \sum_i \frac{w(s^{(i)},a^{(i)})}{\sum_j w(s^{(j)},a^{(j)})} \log p_{\text{LLM}_\theta}(a^{(j)}|s^{(j)}),\ (s,a) \sim q(s,a)\end{aligned}$$

where $w(s,a) = \frac{p(\mathcal{O}=1|a,s)p_{\text{LLM}_\theta}(a,s)}{q(s,a)}$ denotes the importance weight. While the proposal $q(s,a)$ offers substantial flexibility, the statistical efficiency of SNIS is highly sensitive to its choice: suboptimal proposals yield high-variance importance weights, thereby impeding effective state–action exploration. In theory, the optimal proposal, which leads to zero variance, is $q(s,a) \propto p(\mathcal{O}=1|a,s)p_{\text{LLM}_\theta}(a,s)$. While intractable, empirically, the memory table $\mathcal{M}$ provides a practical basis for designing the proposal distribution $q(s,a)$, as it stores state-action pairs accumulated from previous posterior samples. We thus adopt a uniform distribution over the memory table $\mathcal{M}$ as the proposal, providing an empirical approximation to the posterior.

At each refinement iteration k, the memory table $\mathcal{M}_k$ has sampling distribution $q_k(s,a) = q_k(s)q_k(a \mid s)$. We use the memory-induced state marginal $q_k(s)$ as the fixed state distribution, rather than explicitly modeling the current prior's state visitation distribution. For the posterior restricted to the support of $\mathcal{M}_k$, the unnormalized importance weight is:

$$\begin{aligned}w_k(s,a) &= \frac{p_k(\mathcal{O}=1 \mid a,s)p_{\text{LLM}_{\theta_k}}(a,s)}{q_k(s,a)} = \frac{p_k(\mathcal{O}=1 \mid a,s)q_k(s)p_{\text{LLM}_{\theta_k}}(a \mid s)}{q_k(s,a)} \\ &= \frac{p_k(\mathcal{O}=1 \mid a,s)p_{\text{LLM}_{\theta_k}}(a \mid s)}{q_k(a \mid s)} \propto \rho_k(s,a)\exp(Q(s,a)/\tau),\end{aligned} \tag{6}$$

where $\rho_k(s,a) = p_{\text{LLM}_{\theta_k}}(a \mid s)/q_k(a \mid s)$. In implementation, we disentangle the effects of the prior correction ρ_k and value Q in the importance weights, as shown in Table 10. Empirically, Q-only weighting performs comparably to the full ρ-corrected weighting while avoiding repeated LLM queries to estimate the prior. We therefore adopt Q-only weighting in our main experiments for efficiency while maintaining stable training.

4.2.2 MAXIMIZATION STEP.

In the M-step, we maximize the expectation in Eq. 5, which involves the optimality likelihood $p(\mathcal{O} = 1 \mid s, a)$ and the LLM prior p_{LLM_θ}. To maximize the expectation of the non-parametric likelihood, we update the memory according to Eq. 2 using the sample drawn from the posterior. For optimizing the LLM prior, building on the memory-driven expectation estimation, we apply the following memory-based reweighted training objective:

$$\theta_{k+1} = \arg\min_{\theta} \sum_{(s,a)\sim\mathcal{M}_k} \left[\frac{\exp(Q(s,a)/\tau)}{\sum_{(s',a')\sim\mathcal{M}_k} \exp(Q(s',a')/\tau)} \log p_{\text{LLM}_\theta}(a|s) \right]. \tag{7}$$

We summarize the detailed training procedure in Algorithm 2. Alternatively, we consider **prompt-based LLM prior refinement**, which keeps the LLM parameters fixed and performs a shallow M-step through in-context demonstrations. At each iteration k, given the current state s_t, we first retrieve the top-N most similar memory entries, $\mathcal{N}_k(s_t) = \text{TopN}_{(s_i,a_i,Q_i)\in\mathcal{M}_k} \text{sim}(s_t, s_i)$, and then sample the prompt set P_t from $\mathcal{N}_k(s_t)$ according to the Q-based distribution $p_k(i \mid s_t) = \exp(Q_i/\tau)/\sum_{j\in\mathcal{N}_k(s_t)} \exp(Q_j/\tau),\ P_t \sim p_k(\cdot \mid s_t)$. The selected demonstrations condition the LLM prior as $p_{\text{LLM}_{\theta_k}}(a \mid s_t, P_t)$, enabling lightweight prior refinement without parameter updates; see Appendix C.2 for details. It is noteworthy that the memory-driven Q estimation and policy refinement interact in a bootstrapping manner. Specifically, the updated memory table continuously refines the non-parametric estimate of the optimality likelihood, which in turn provides more accurate feedback for updating the LLM prior. Conversely, the improved LLM prior constrains the action search space toward high-quality candidates, thereby accelerating convergence and improving memory-driven Q estimation. This closed-loop interaction establishes a self-improvement cycle in which the two components iteratively enhance each other, leading to progressively stronger policies.

5 EXPERIMENTS

5.1 ENVIRONMENTS

We evaluate our method on three textual agentic decision-making tasks and two multi-step mathematical reasoning tasks. **Overcooked** (Tan et al., 2024) requires sequential actions to prepare and deliver dishes, where we consider the Tomato and Salad tasks with state-dependent action spaces of up to 8 actions. **VirtualHome** (Puig et al., 2018) evaluates food preparation across four rooms with sparse rewards and up to 10 possible actions. **ALFWorld** (Shridhar et al., 2020) involves textual navigation and interaction with larger state-dependent action spaces of up to 50 actions; we consider the Pick and Examine tasks. Beyond SDM tasks, we evaluate our method on **two multi-step math reasoning tasks** to further assess its generalization ability. **Game of 24** (Yao et al., 2024) evaluates three-step arithmetic reasoning using 175 training and 229 test instances, while **MiniF2F** (Zheng et al., 2022) evaluates formal theorem proving in Lean **with free-form tactic generation**, using 244 training and 244 test problems. Detailed environment descriptions are provided in Appendix D.1.

5.2 BASELINES

We compare against the following baselines:
LLM prior: We first assess the ability of LLMs to solve decision-making tasks in a zero-shot manner, without relying on deliberately designed prompts.
DQN-based methods: We compare against **DQN** and its variant **DQN-prior** Yan et al. (2025), which performs deep Q-learning within a narrowed sub-action space generated by LLM priors.
Our memory-driven algorithms: **Mem-Q** (Algorithm 1) leverages language models' embedding capabilities and the kernel-based value estimation to perform non-parametric Q-learning.
PPO is a policy-based LLM-tuning approach, where the LLM acts as the action policy and is fine-tuned following the PPO algorithm.
We also consider **two retrieval-based memory prompting baselines**. **MemoryBank** Zhong et al. (2024) retrieves semantically similar memories based on the current state as in-context examples; we adapt it to SDM tasks by storing experienced state-action pairs as memory entries. **MemRL** Zhang et al. (2026) instead summarizes trajectory-level experiences and estimates their utility from the final reward when used as in-context prompts. During inference, examples are selected based jointly on

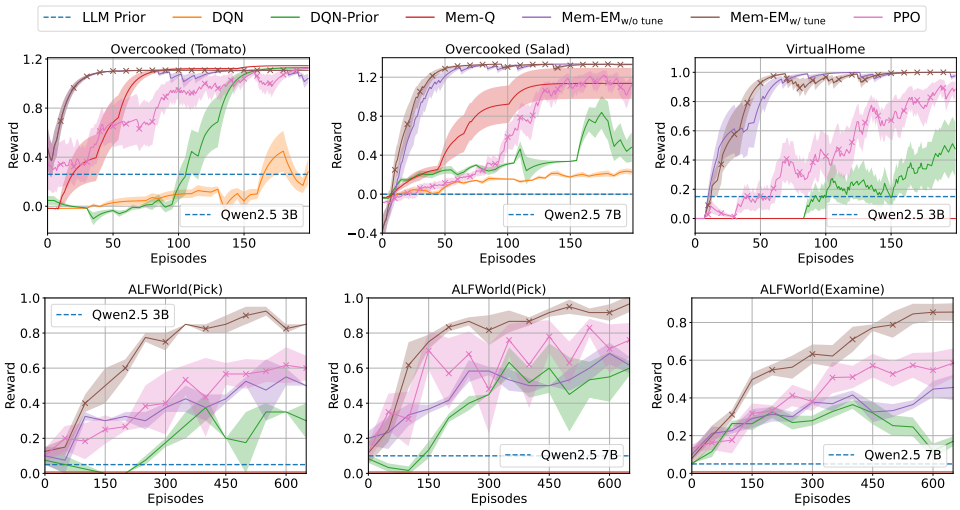


Figure 3: Results of comparison with baselines. We plot the mean and standard error of the cumulative reward across 5 random seeds. The dashed line represents directly prompting the LLM prior to generating actions given state information, with the corresponding LLM version specified. The '$\times$' markers for Mem-EM$_{\text{w/ tune}}$ indicate the time steps when the LLM prior is fine-tuned.

initial-state similarity and their estimated Q-values.

Mem-EM (Algorithm 2) formulates decision-making as an EM procedure that integrates memory-driven Q-value estimation with LLM prior refinement. We consider two variants: **Mem-EM**$_{\text{w/ tune}}$ periodically fine-tunes the LLM prior using examples from the memory table, while **Mem-EM**$_{\text{w/o tune}}$ keeps the LLM prior fixed. We further consider **Mem-EM**$_{\text{w/o tune+Prompt}}$, which performs a shallow M-step through retrieval-based prompting, with the LLM prior given by $p_{\text{LLM}}(a \mid s_t, P_t)$, where P_t denotes the retrieved in-context examples. For a fair comparison, we use three in-context examples for all prompt-based baselines, including **MemoryBank**, **MemRL**, and **Mem-EM**$_{\text{w/o tune+Prompt}}$.

Experimental Settings We use Qwen2.5-3B (Team, 2024), Qwen2.5-7B, Qwen3-8BYang et al. (2025) and Deepseek-V4-FlashXu et al. (2026) as the LLM prior. The retrieval size M is set to 20, and the number of action candidates K is set to 5 or 10. The fixed BERT-base model (Devlin et al., 2019) is used to obtain semantic representations for time consideration. Detailed hyperparameter settings are provided in the Appendix D.

5.3 Results

5.3.1 Comparison with Baselines on Agentic Decision-Making Tasks

The comparison results of baselines are shown in Figure 3. These results demonstrate that Mem-EM$_{\text{w/ tune}}$ outperforms all other baselines, remarkably achieving over 40% improvement on complex ALFWorld environments. We observe that memory-based Mem-Q significantly outperforms DQN on Overcooked, and memory-based Mem-EM$_{\text{w/o tune}}$ achieves comparable or better performance than DQN-prior, demonstrating that the memory-driven approaches provide satisfactory value estimation and improve sample efficiency. However, for the complex ALFWorld, where both state and action spaces are extremely large, neither DQN nor Mem-Q alone can solve the task, highlighting the necessity of incorporating LLM priors.

By leveraging LLM priors to generate valuable action candidates, the baselines DQN-Prior and Mem-EM$_{\text{w/o tune}}$ consistently outperform their vanilla counterparts (DQN and Mem-Q) that explore the full original action space. Nevertheless, since pretrained LLMs lack task-specific knowledge, fixed LLM priors inherently limit performance and may even degrade it.

Importantly, results show that Mem-EM$_{\text{w/ tune}}$ substantially outperforms Mem-EM$_{\text{w/o tune}}$ on complex ALFWorld. This demonstrates that our memory-driven policy optimization effectively integrates domain-specific knowledge into the LLM prior, thereby improving its decision-making ability. Furthermore, as illustrated in the ALFWorld experiments, the EM-based framework Mem-EM$_{\text{w/ tune}}$ requires only six time LLM tuning throughout training, with LoRA Hu et al. (2021) used for parameter-efficient fine-tuning. This keeps the computational overhead tolerable while yielding

Table 1: **Game of 24 results.** LLM-based methods use Qwen3-8B, with BERT embeddings unless otherwise noted, on a 175/229 train/test split. The three blocks compare memory-only decision making, LLM-based baselines, and Mem-EM variants. Replacing BERT with Qwen3-8B embeddings examines the role of representation in value estimation; prompt technique and fine-tuning examine two ways of incorporating experience into the action policy. The fine-tuned greedy variant removes test-time memory access, isolating the capability distilled in the policy. **Bold** indicates the highest test win rate; underlining highlights memory-free performance after fine-tuning.

Method	Test-time decision	Train win rate	Test win rate
Random Sample	All legal actions	–	0.88%
Mem-Q	All legal actions + Q value	62.29%	10.48%
Mem-Q (Qwen Emb.)	All legal actions + Q value (Qwen Emb.)	60.00%	14.41%
Qwen3-8B zero-shot	One generated action	–	10.67%
MemoryBank	One generated action + Prompt	15.43%	11.79%
MemRL	One generated action + Prompt	20.57%	13.10%
Mem-EM$_{\text{w/o tune}}$	Ten proposals + Q value	18.86%	11.35%
Mem-EM$_{\text{w/o tune}}$ (Qwen Emb.)	Ten proposals + Q value (Qwen Emb.)	18.29%	12.66%
Mem-EM$_{\text{w/o tune + Prompt}}$	Ten proposals + Q value + Prompt	22.29%	15.28%
Mem-EM$_{\text{w tune}}$-Greedy	One generated action from tuned LLM	37.14%	18.78%
Mem-EM$_{\text{w tune}}$	Ten proposals + Q value	**39.57%**	**19.65%**

significant performance improvements. Additionally, we demonstrate the superior **generalization ability** to unseen tasks of our refined LLM prior and memory-based Q-value estimator ; please refer to Appendix E.2 for details.

5.3.2 Generalization on Mathematical Reasoning Tasks

Results on Game of 24. Table 1 compares the generalization performance of different methods. Mem-Q achieves strong training performance but exhibits a substantial train–test gap. Replacing BERT with Qwen embeddings improves test performance, suggesting that representation quality matters for value transfer. Retrieving similar experiences as prompts improves over the zero-shot baseline, but the gains remain limited. The experience-reuse ablation in Appendix (Figure 7) further shows that prompt-based reuse yields modest training gains while test performance plateaus, suggesting limitations of a frozen LLM prior in leveraging additional experience. In contrast, policy fine-tuning gradually improves generalization. Comparing Mem-EM$_{\text{w/tune}}$ with Mem-EM$_{\text{w/tune}}$-Greedy shows that the fine-tuned policy retains most of its performance under memory-free greedy decoding. Together, these results suggest that the M-step consolidates experience into a transferable policy, while memory-guided selection provides a smaller, complementary decision-time benefit.

Results on MiniF2F. MiniF2F features a free-form action space and horizons of up to 20 steps, with each LLM-generated action verified by Lean. This makes it challenging, with Qwen3-8B achieving only 1.23% success. Table 2 shows that Mem-EM achieves the highest test win rate in the DeepSeek block. More importantly, distilling the memory table collected by Mem-EM$_{\text{w/o tune}}$ with DeepSeek into Qwen3-8B improves its test win rate from 1.23% to 18.85%, without test-time memory access. This distilled prior further reaches 24.18% with Mem-EM and 25.41% with continued fine-tuning. These results show that feedback-grounded memory can transfer experience across models of different scales and support continued policy refinement in free-form, long-horizon math reasoning.

5.3.3 Ablation Studies

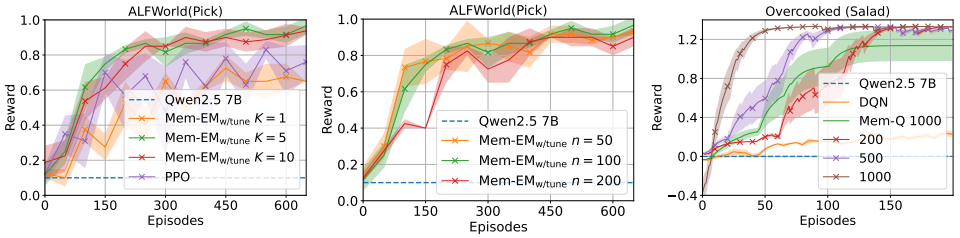
The ablation studies analyze the following aspects:

Effect of the number of action candidates. Figure 4(a) illustrates the impact of the number of action candidates K for Mem-EM$_{\text{w/ tune}}$. Our method with $K = 5$ or 10 significantly outperforms the setting with $K = 1$ and PPO. It is worth noting that Mem-EM$_{\text{w/ tune}}$ with $K = 1$ resembles a traditional actor–critic RL setting, where the action policy samples a single action at each step while a critic model, trained on environment returns, guides policy learning. These results highlight the superiority of our EM-based training framework over traditional RL approaches, demonstrating the clear benefits of Q-value-guided posterior action sampling.

Effect of the LLM prior update interval. Fig. 4(b) examines the effect of the fine-tuning interval n,

Table 2: **MiniF2F results.** The DeepSeek block compares Mem-EM with prompt-based baselines after 400 training episodes. The Qwen block evaluates memory-to-policy distillation using the memory table collected by DeepSeek Mem-EM, followed by 400 additional episodes of Mem-EM refinement with or without further fine-tuning. The fine-tuning variant performs two updates, one every 200 episodes.

Method	Test-time decision	Train win rate	Test win rate
DeepSeek-V4-Flash-284B			
Zero-shot	One generated tactic	–	30.74%
MemoryBank	One generated tactic + Prompt	36.48%	32.79%
MemRL	One generated tactic + Prompt	38.11%	33.20%
Mem-EM$_{\text{w/o tune}}$	Five proposals + Q value	36.07%	**34.43%**
Qwen3-8B			
Zero-shot	One generated tactic	–	1.23%
Memory-distilled prior	One generated tactic from tuned LLM	15.98%	18.85%
Mem-EM$_{\text{w/o tune}}$	Five proposals + Q value	24.59%	24.18%
Mem-EM$_{\text{w tune}}$	Five proposals + Q value	26.23%	**25.41%**



(a) Action candidates (b) Tuning interval (c) Memory capacity

Figure 4: Ablation studies. (a) Effect of the number of action candidates K generated by the LLM. The '$\times$' markers indicate the time steps when the LLM-prior is updated. (b) Impact of the LLM fine-tuning interval, where n denotes that the LLM policy is fine-tuned every n episodes. (c) Influence of the memory table capacity N on Mem-EM$_{\text{w/tune}}$, where at most N (s, a) pairs are stored, with the least-recently-used (LRU) strategy applied for replacement.

where the LLM prior is updated every n episodes. Results show that Mem-EM$_{\text{w/ tune}}$ is robust to the update interval and does not require frequent fine-tuning to achieve strong performance.

Effect of memory capacity. Figure 4(c) shows the effect of memory table capacity N of Mem-EM$_{\text{w/tune}}$ on Overcooked (Salad). Mem-EM$_{\text{w/tune}}$ demonstrates robustness to limited memory capacity with a simple least-recently-used (LRU) replacement strategy to retain crucial (s, a) pairs. This robustness is attributed to the LLM's role in reducing the exploration space, allowing the model to converge effectively with only 200 memory items, despite a total of approximately 1,000 possible state-action pairs in the environment.

6 Conclusions

In this work, we propose a memory-driven self-improvement framework for decision-making tasks. The framework consists of two mutually reinforcing components: memory-driven value estimation and memory-driven LLM prior refinement. The memory-driven value estimation approach maintains historical interactions and their Q-values, performing non-parametric value estimation through retrieval of similar representations. Building on the EM formulation, we design a practical and stable memory-driven LLM prior refinement algorithm, which adapts task-specific knowledge into the LLM prior by learning from the memory table. Experimental results show that our EM-based self-improvement framework delivers substantial performance gains while avoiding extensive fine-tuning. A current limitation is that the accuracy of memory-driven Q estimation depends on the quality of state-action representations. We currently use embeddings from general-purpose LLMs, which are not explicitly optimized for decision making. Future work could develop decision-aware embedding models that better preserve action semantics and value-relevant structure, thereby improving retrieval and Q-value estimation.

AI USE STATEMENT

In this work, we used large language models (LLMs) to assist with writing and polishing the manuscript. Specifically, LLMs were employed to improve the grammar, clarity, and overall readability of the text. All scientific content, ideas, and experimental results were generated and verified by the authors. The authors take responsibility for the final content.

ETHICS STATEMENT

We develop an algorithm for sequential decision-making and evaluate it using publicly available benchmarks and environments. We do not identify specific ethical concerns arising from the experiments conducted in this work.

REPRODUCIBILITY STATEMENT

The main text describes our method. The appendix provides additional derivations, implementation details, experimental settings, and hyperparameters. We will publicly release our code upon acceptance.

REFERENCES

Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):411–444, 2022.

Krishan Rana, Ming Xu, Brendan Tidd, Michael Milford, and Niko Sünderhauf. Residual skill policies: Learning an adaptable skill-based action space for reinforcement learning for robotics. In *Conference on Robot Learning*, pages 2095–2104. PMLR, 2023.

José Eugenio Naranjo, Carlos González, Ricardo García, Teresa de Pedro, and Rodolfo E Haber. Power-steering control architecture for automatic driving. *Ieee transactions on intelligent transportation systems*, 6(4):406–415, 2005.

Qingyuan Song, Weiping Fu, Wen Wang, Yuan Sun, Denggui Wang, and Jincao Zhou. Quantum decision making in automatic driving. *Scientific reports*, 12(1):11042, 2022.

Scott R Granter, Andrew H Beck, and David J Papke Jr. Alphago, deep learning, and the future of the human microscopist. *Archives of pathology & laboratory medicine*, 141(5):619–621, 2017.

Ziming Li, Julia Kiseleva, and Maarten De Rijke. Dialogue generation: From imitation learning to inverse reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6722–6729, 2019.

Michael McTear. *Conversational ai: Dialogue systems, conversational agents, and chatbots*. Springer Nature, 2022.

Sai Dhanush Jannala. *Chess LLM Arena: A Framework for Evaluating Strategic Decision-Making in Large Language Models*. PhD thesis, Dublin, National College of Ireland, 2025.

Xuanfa Jin, Ziyan Wang, Yali Du, Meng Fang, Haifeng Zhang, and Jun Wang. Learning to discuss strategically: A case study on one night ultimate werewolf. *Advances in Neural Information Processing Systems*, 37:77060–77097, 2024.

Weiyu Ma, Qirui Mi, Yongcheng Zeng, Xue Yan, Runji Lin, Yuqiao Wu, Jun Wang, and Haifeng Zhang. Large language models play starcraft ii: Benchmarks and a chain of summarization approach. *Advances in Neural Information Processing Systems*, 37:133386–133442, 2024a.

Jiaqi Wang, Enze Shi, Huawen Hu, Chong Ma, Yiheng Liu, Xuhui Wang, Yincheng Yao, Xuan Liu, Bao Ge, and Shu Zhang. Large language models for robotics: Opportunities, challenges, and perspectives. *Journal of Automation and Intelligence*, 4(1):52–64, 2025.

Xue Yan, Yan Song, Xidong Feng, Mengyue Yang, Haifeng Zhang, Haitham Bou Ammar, and Jun Wang. Efficient reinforcement learning with large language model priors. In *The Thirteenth International Conference on Learning Representations*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 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, 2024.

Filippos Christianos, Georgios Papoudakis, Matthieu Zimmer, Thomas Coste, Zhihao Wu, Jingxuan Chen, Khyati Khandelwal, James Doran, Xidong Feng, Jiacheng Liu, Zheng Xiong, Yicheng Luo, Jianye Hao, Kun Shao, Haitham Bou-Ammar, and Jun Wang. Pangu-agent: A fine-tunable generalist agent with structured reasoning, 2023.

Runlong Zhou, Simon Du, and Beibin Li. Reflect-rl: Two-player online rl fine-tuning for lms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 995–1015, 2024.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR, 2023.

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.

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Xing Chen, Dongcui Diao, Hechang Chen, Hengshuai Yao, Haiyin Piao, Zhixiao Sun, Zhiwei Yang, Randy Goebel, Bei Jiang, and Yi Chang. The sufficiency of off-policyness and soft clipping: Ppo is still insufficient according to an off-policy measure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7078–7086, 2023.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*, 2023.

Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Nirmalie Wiratunga, Ramitha Abeyratne, Lasal Jayawardena, Kyle Martin, Stewart Massie, Ikechukwu Nkisi-Orji, Ruvan Weerasinghe, Anne Liret, and Bruno Fleisch. Cbr-rag: case-based reasoning for retrieval augmented generation in llms for legal question answering. In *International Conference on Case-Based Reasoning*, pages 445–460. Springer, 2024.

Chi Han, Ziqi Wang, Han Zhao, and Heng Ji. Explaining emergent in-context learning as kernel regression. *arXiv preprint arXiv:2305.12766*, 2023.

Kenneth Li, Samy Jelassi, Hugh Zhang, Sham M Kakade, Martin Wattenberg, and David Brandfonbrener. Q-probe: A lightweight approach to reward maximization for language models. In *International Conference on Machine Learning*, pages 27955–27968. PMLR, 2024.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8494–8502, 2018.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. minif2f: A cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2109.00110.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731, 2024.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026.

Qwen Team. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. In *Forty-first International Conference on Machine Learning*, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Nat McAleese, Rai Michael Pokorny, Juan Felipe Ceron Uribe, Evgenia Nitishinskaya, Maja Trebacz, and Jan Leike. Llm critics help catch llm bugs. *arXiv preprint arXiv:2407.00215*, 2024.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024a.

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montserrat Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. In *Conference on Robot Learning*, pages 374–404. PMLR, 2023.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *The Twelfth International Conference on Learning Representations*, 2024b.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *International conference on machine learning*, pages 2827–2836. PMLR, 2017.

Zhuo Li, Derui Zhu, Yujing Hu, Xiaofei Xie, Lei Ma, Yan Zheng, Yan Song, Yingfeng Chen, and Jianjun Zhao. Neural episodic control with state abstraction. *arXiv preprint arXiv:2301.11490*, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pages 1861–1870. PMLR, 2018.

Xue Yan, Jiaxian Guo, Xingzhou Lou, Jun Wang, Haifeng Zhang, and Yali Du. An efficient end-to-end training approach for zero-shot human-ai coordination. *Advances in Neural Information Processing Systems*, 36, 2024.

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

Chengrui Wang, Qingqing Long, Meng Xiao, Xunxin Cai, Chengjun Wu, Zhen Meng, Xuezhi Wang, and Yuanchun Zhou. Biorag: A rag-llm framework for biological question reasoning. *arXiv preprint arXiv:2408.01107*, 2024b.

Muning Wen, Ziyu Wan, Jun Wang, Weinan Zhang, and Ying Wen. Reinforcing llm agents via policy optimization with action decomposition. *Advances in Neural Information Processing Systems*, 37: 103774–103805, 2024.

APPENDIX CONTENTS

A Related Work **15**

B Additional Explaination on Memory-driven value estimation **15**

C Additional Explanation of Memory-Driven Policy Optimization **16**

C.1 Monotonic improvement of EM framework 16

C.2 Prompt-based refinement 17

D Detailed Experimental Settings **17**

D.1 Environments 17

D.2 LLM prior implementation 19

D.3 Computational Resources and Hyperparameters 20

E Additional Results **20**

E.1 Effect of the training objective. 20

E.2 Generalization Ability on ALFWorld. 20

E.3 Additional baselines on agentic tasks 21

E.4 Testing across Different Foundation LLMs 22

E.5 Ablation on Experience Reuse 23

E.6 Computational Resource Requirements 23

E.7 Additional results on the hyperparameter tuning 23

Algorithm 2 Memory-Driven Expectation Maximization (Mem-EM) with LLM Prior Refinement

Input: LLM action prior p_{LLM_θ}, memory table $\mathcal{M} = \emptyset$
Output: Refined LLM prior p_{LLM_θ} and memory table $\mathcal{M}$

for episode $i = 1$ to N **do**
  for step $t = 1, 2, \ldots, T$ **do**
    Sample action candidates from LLM prior $\mathcal{C}^K(s_t) \sim p_{\text{LLM}_\theta}(\cdot|s_t)$.
    Estimate Q-values $\widehat{Q}(s_t, a)$ for $a \in \mathcal{C}^K(s_t)$ via Eq. 3.
    Select action $a \sim \text{Multinormial}(\exp(\widehat{Q}(s, a_1)/\tau)/\sum_k \exp(\widehat{Q}(s, a_k)/\tau))$.
    Execute a_t, observe reward r_{t+1} and next state s_{t+1}
  end for
  for step $t = T$ down to 1 **do**
    Write and update memory table $\mathcal{M}$ via Eq. 2.
  end for
  if i reaches update interval **then**
    Refine LLM prior p_{LLM_θ} using stored (s, a) pairs from $\mathcal{M}$ via Eq. 7.
  end if
end for

Note: Orange text highlights LLM-guided exploration steps that differ from the Mem-Q in Algorithm 1; green text indicates prior refinement operations.

A Related Work

LLM Priors in Decision-Making Recent works leverage LLMs to enhance sequential decision-making (SDM) in three main ways: action generation, value estimation, and reward function design. First, LLMs can act as the action policy, generating satisfactory actions either through deliberate prompting Yao et al. (2022); Shinn et al. (2024) or RL-based fine-tuning Carta et al. (2023); Tan et al. (2024). Second, LLMs can serve as the value function to guide the search. Examples include reasoning-path search Wang et al. (2022); Yao et al. (2023) and Monte Carlo Tree Search guided by LLM evaluations Hao et al.; Wan et al. (2024). In addition, LLMs can be fine-tuned to act as process or outcome reward models with detailed explanations Lightman et al. (2023); McAleese et al. (2024); Wang et al. (2024a). Third, LLMs are used to generate reward signals for RL, either directly by prompting with historical interactions Kwon et al. (2023) or by producing executable reward code for continuous-control tasks Yu et al. (2023); Ma et al. (2024b).

Memory-based Decision-Making Episodic Control (EC) Blundell et al. (2016) and its extensions Pritzel et al. (2017); Li et al. (2023) represent a classic family of memory-based methods. These approaches maintain $|\mathcal{A}|$ separate memory tables, one for each feasible action, and apply kernel-based estimation over state representations. Yet, EC relies solely on state similarity and ignores semantic relationships among actions. Off-policy methods such as DQN Mnih (2013) and SAC Haarnoja et al. (2018) can also be viewed as memory-based RL, with the replay buffer serving as memory. Yan et al. (2024) further combine DQN with LLM embeddings, but their approach compresses stored experiences into Q-networks that map high-dimensional embeddings to scalar values, potentially discarding semantic information and limiting sample efficiency. More recently, memory-based methods have been integrated with LLMs under the retrieval-augmented generation (RAG) paradigm Zhou et al. (2025); Wang et al. (2024b). These approaches retrieve relevant cases to enhance LLM outputs via in-context learning Han et al. (2023), but they primarily focus on one-step tasks such as question answering Wiratunga et al. (2024) or high-level planning Zhou et al. (2025). By contrast, our work applies retrieval techniques to sequential decision-making, and explicitly incorporates domain-specific memory to refine LLM-based policies.

B Additional Explaination on Memory-driven value estimation

In the main paper, we presented memory-driven Q-learning estimation without training. Here, we explore an alternative approach that involves tuning the embedding function. As described above, we use the BERT-base model as the embedding function f_θ, which can be further fine-tuned by minimizing the distance between the predicted Q-value and the Monte Carlo estimate:

$$\ell = -(\hat{Q}_\theta(s_t, a) - y_t)^2,$$

where $y_t = \sum_{i=t}^{T} \gamma^{i-t} r_t$. The results of fine-tuning the BERT embedding model are shown in Figure 5. We observe that our model, both with and without fine-tuning, outperforms DQN-Prior. All three baselines use the LLM prior to narrow the action search space, and fine-tuning versus keeping the BERT embedding model fixed yields similar performance.

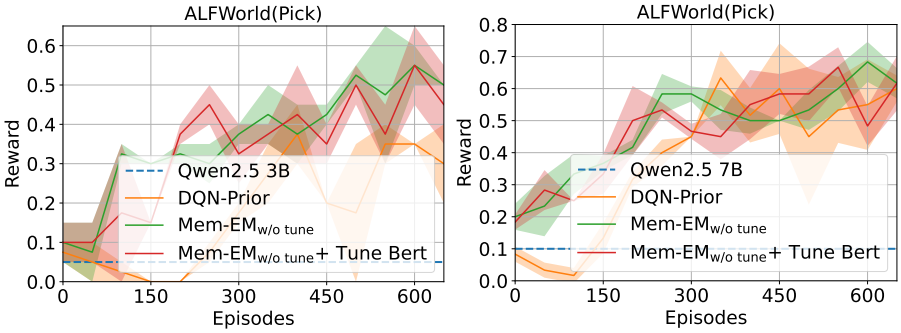


Figure 5: Ablation study on finetuning the embedding model bert.

C Additional Explanation of Memory-Driven Policy Optimization

C.1 Monotonic improvement of EM framework

In this section, we briefly review the Expectation–Maximization (EM) algorithm. We start by introducing the evidence lower bound (ELBO) for the log-likelihood:

$$\log p_\theta(\mathcal{O}=1|s) = \sum_a \log p_\theta(\mathcal{O}=1|s,a) p_\theta(a|s) \tag{8}$$

$$= \sum_a \log \left[q_\phi(a|s) \frac{p_\theta(\mathcal{O}=1|s,a) p_\theta(a|s)}{q_\phi(a|s)} \right] \tag{9}$$

$$\geq \mathbb{E}_{q_\phi(a|s)}[\log p_\theta(\mathcal{O}=1|s,a)] - D_{\mathrm{KL}}(q_\phi(a|s) \| p_\theta(a|s)) \triangleq \mathrm{ELBO}(\phi,\theta) \tag{10}$$

In variational inference, maximizing the likelihood can be reformulated as the following two-step optimization:

$$\phi_{k+1} \leftarrow \arg\max_\phi \mathrm{ELBO}(\phi, \theta_k) \tag{11}$$

$$\theta_{k+1} \leftarrow \arg\max_\theta \mathrm{ELBO}(\phi_{k+1}, \theta). \tag{12}$$

It can be shown that when $q_{\phi_{k+1}} = p_{\theta_k}(a|s, \mathcal{O}=1)$, the ELBO attains its maximum value, which coincides with the true log-likelihood:

$$\mathbb{E}_{p_\theta(a|s,\mathcal{O}=1)}[\log p_\theta(\mathcal{O}=1|s,a)] - D_{\mathrm{KL}}(p_\theta(a|s,\mathcal{O}=1) \| p_\theta(a|s)) \tag{13}$$

$$= \mathbb{E}_{p_\theta(a|s,\mathcal{O}=1)} \log \left[\frac{p_\theta(a, \mathcal{O}=1|s)}{p_\theta(a|s,\mathcal{O}=1)} \right] \tag{14}$$

$$= \mathbb{E}_{p_\theta(a|s,\mathcal{O}=1)} \log \left[\frac{p_\theta(a|s,\mathcal{O}=1) p_\theta(\mathcal{O}=1|s)}{p_\theta(a|s,\mathcal{O}=1)} \right] \tag{15}$$

$$= \mathbb{E}_{p_\theta(a|s,\mathcal{O}=1)}[\log p_\theta(\mathcal{O}=1|s)] \tag{16}$$

$$= \log p_\theta(\mathcal{O}=1|s). \tag{17}$$

Thus, EM maximizes the following objective:

$$\theta_{k+1} = \arg\max_\theta \mathbb{E}_{p_{\theta_k}(a|s,\mathcal{O}=1)}[\log p_\theta(\mathcal{O}=1|s,a)] - D_{\mathrm{KL}}(p_{\theta_k}(a|s,\mathcal{O}=1) \| p_\theta(a|s), \tag{18}$$

which is equivalent to

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{p_{\theta_k}(a,s|\mathcal{O}=1)} \left[\log p_\theta(\mathcal{O}=1|s,a) + \log p_\theta(a|s)\right]. \tag{19}$$

Although this optimization is generally intractable, in the E-step one can approximate the expectation using Monte Carlo estimation, followed by the M-step to update the parameter θ. Finally, since the ELBO lower bounds the log-likelihood, each EM iteration guarantees monotonic improvement $\log p_{\theta_{k+1}}(\mathcal{O}=1|s) \geq \log p_{\theta_k}(\mathcal{O}=1|s)$.

ρ-corrected weighting computation Eq. 6 show the importance weights combined by Q-value and ratio ρ_k. For uniform sampling over a memory table containing one entry per distinct state-action pair, we have $q_k(a \mid s) = 1/n_k(s)$, where $n_k(s) = |\{a : (s,a) \in \mathcal{M}_k\}|$. Consequently, the importance ratio becomes $\rho_k(s,a) = \frac{p_{\mathrm{LLM}_\theta}(a|s)}{q_k(a|s)} = n_k(s) p_{\mathrm{LLM}_{\theta_k}}(a \mid s)$. This factor corrects the mismatch between the conditional action sampling distribution in memory and the current LLM prior. All importance weights are computed using θ_k and held fixed during the M-step.

C.2 Prompt-based refinement

Fine-tuning large LLMs can be computationally expensive. To broaden the applicability of our framework, we alternatively keep the LLM parameters fixed and perform a shallow M-step through in-context demonstrations. Following retrieval-based memory prompting methods such as MemoryBankZhong et al. (2024) and MemRLZhang et al. (2026), we first restrict demonstrations to memories relevant to the current state, since semantically related experiences generally provide more useful in-context guidance. Specifically, given a query state s, we retrieve the top-N memory entries according to state similarity,

$$\mathcal{N}_k(s) = \mathrm{TopN}_{(s^{(i)},a^{(i)},Q^{(i)})\in\mathcal{M}_k} \mathrm{sim}(s, s^{(i)}), \tag{20}$$

where we use $N = 10$ in our experiments.

Within this state-relevant subset, we sample K demonstrations according to their normalized Q-value weights. For each $i \in \mathcal{N}_k(s)$, we define

$$\bar{w}_k^{(i)} = \frac{\exp(Q^{(i)}/\tau_p)}{\sum_{j\in\mathcal{N}_k(s)} \exp(Q^{(j)}/\tau_p)}, \tag{21}$$

and independently sample

$$I_\ell \sim \mathrm{Categorical}\left(\{\bar{w}_k^{(i)}\}_{i\in\mathcal{N}_k(s)}\right), \qquad \ell = 1, \ldots, K. \tag{22}$$

The resulting prompt is

$$P_{k+1}(s) = \left((s^{(I_1)}, a^{(I_1)}), \ldots, (s^{(I_K)}, a^{(I_K)})\right), \tag{23}$$

which induces the prompt-conditioned prior $p_{\mathrm{LLM}_\theta}(a \mid s; P_{k+1}(s))$, with θ unchanged. Since each demonstration is independently sampled according to $\bar{w}_k^{(i)}$, memories with larger Q-values are more likely to appear in the prompt. The resulting demonstrations therefore provide the frozen LLM with state-relevant, value-weighted experiences, yielding a lightweight approximation to the M-step through in-context learning. This does not imply that in-context conditioning exactly maximizes the M-step objective; rather, it leverages the LLM's in-context learning capability to bias the action prior toward high-value experiences without updating its parameters. Fig. 7 shows that prompt-based methods are largely constrained by the LLM's inherent capabilities, tend to plateau quickly, and generalize less effectively than the fine-tuning-based M-step.

D Detailed Experimental Settings

D.1 Environments

We consider **three textual agentic decision-making tasks**:
Overcooked The textual Overcooked environment (Tan et al., 2024) benchmarks the ability of taking

a sequence of actions to deliver dishes. We consider two Overcooked tasks: Overcooked(Tomato), which requires delivering a dish of chopped tomato, and Overcooked(Salad), which requires delivering a salad containing chopped tomato and lettuce. The feasible actions vary with state changes, and the maximum number of feasible actions for one state is 8. Besides the reward of 1 for successfully delivering a dish, this environment also provides the following reward shaping: +0.2 for correctly chopping an ingredient, +1 terminal reward for successfully delivering the correct dish, -0.1 for delivering any incorrect item, and -0.001 for each time step.

VirtualHome (Puig et al., 2018) is widely used to evaluate LLM-based decision-making (Tan et al., 2024; Wen et al., 2024). We consider the food preparation task, which requires the agent to complete the task within 4 rooms. The reward is sparse, i.e., the agent receives a reward of 1 only upon completing the target and 0 otherwise. The action space consists of up to 10 possible actions.

ALFWorld (Shridhar et al., 2020) is a popular and complex decision-making benchmark for navigating and completing tasks within rooms. Both observations and actions are textual, and the action space consists of high-level plans such as "go to a room" that can be understood using LLMs' prior knowledge. The action space is state-varying and large, with the maximum possible admissible action space for one step reaches up to 50, making exploration from scratch challenging. We consider two classes of ALFWorld tasks: ALFWorld(pick) and ALFWorld(examine). There are no auxiliary rewards except for a reward of 1.0 for reaching the final goal.

Beyond SDM tasks, we evaluate our method on **two multi-step math reasoning tasks** to further assess its generalization ability. **Game of 24** (Yao et al., 2024) requires solving each instance in three steps. At each step, the agent combines two remaining numbers using one of the four basic arithmetic operations. An episode succeeds if the final number is 24. We use 175 training and 229 disjoint test instances, with numbers ranging from 1 to 13.

MiniF2F (Zheng et al., 2022) is a benchmark of formal mathematical problems in Lean. At each step, the agent selects a tactic(i.e. action) based on the current proof state, and Lean returns the updated state or an error. **Tactics are generated in free form** rather than selected from a predefined action set, making the task challenging. An episode succeeds if Lean verifies the proof within 20 steps. We use 244 training and 244 test problems.

Examples of observations and admissible actions for ALFWorld and Overcooked are shown below:

For ALFWorld(Pick)

Observation: Task: Your task is to: put some knife on sidetable. Current observation:You open the drawer 2. The drawer 2 is open. In it, you see nothing..
Admissible actions You are allowed to take the following actions: close drawer 2, examine drawer 2, go to cabinet 1, go to cabinet 2, go to cabinet 3, go to cabinet 4, go to coffeemachine 1, go to countertop 1, go to drawer 1, go to drawer 3, go to drawer 4, go to drawer 5, go to drawer 6, go to drawer 7, go to drawer 8, go to fridge 1, go to garbagecan 1, go to microwave 1, go to sidetable 1, go to sinkbasin 1, go to stoveburner 1, go to stoveburner 2, go to stoveburner 3, go to stoveburner 4, go to toaster 1, inventory, look.

For Overcooked(Salad)

Observation: There are two fixed cutting boards in the room. You notice a tomato and an onion on the different tables. Currently you are carrying an unchopped lettuce in hand. To serve the dish of a bowl only containing chopped tomato and lettuce, what action should you take next?
Admissible actions You are allowed to take the following actions: 'pick up the tomato', 'pick up the lettuce', 'pick up the onion', 'take the empty bowl', 'put the lettuce on the first cutting board', 'put the lettuce on the second cutting board', 'serve the dish', 'chop nothing

For mathematical reasoning tasks, we include Game of 24, with a well-defined action set, and MiniF2F, where free-form tactics are generated and verified by Lean. Examples are shown below.

For Game of 24

Observation:
Initial numbers: 4, 7, 8, 8. Remaining numbers: 4, 7, 8, 8. Target: 24.
Admissible actions:
The agent selects two remaining numbers and one arithmetic operator (+, −, ⋆, or /). For example: `combine 8 * 4 = 32`, `combine 8 - 4 = 4`, or `combine 8 / 4 = 2`.
After the selected action is executed, the environment returns the updated remaining numbers. The episode ends after three actions, and succeeds if the final number is 24.

For MiniF2F

Observation:
Theorem: prove that for every natural number n, $n + 0 = n$.
Current Lean proof state:

```
n : Nat
|- n + 0 = n
```

```
simp
```

Action:
The agent freely generates a Lean tactic, rather than selecting from a predefined action set. For example:

```
rfl
```

Lean executes the tactic and returns the updated proof state or an error. The episode succeeds when the proof is completed and verified by Lean, within the 20-step limit.

D.2 LLM PRIOR IMPLEMENTATION

Following Yan et al. (2024), we use the LLM prior to generate K action candidates by sampling K free-form outputs in parallel from the LLM, given the current state and all admissible actions. These outputs are then mapped to executable actions via a simple rule-based projection $\mathcal{P}$. Mathematically, this can be described as: $a \sim p_{\text{LLM}}(\cdot \mid s_t) \longleftrightarrow o \sim \text{LLM}(\cdot \mid s_t),\ a = \mathcal{P}(o)$, where $\mathcal{P}$ is a rule-based mapping that selects the most frequently occurring action in the LLM output o.

An Example of LLM output and rule-based mapping is given as:

For ALFWorld(Pick) from Qwen2.5 7B

Input: Task: Your task is to: put some alarmclock on desk. Current observation:You turn on the desklamp 1.. You are allowed to take the following actions: examine desk 1, examine keychain 3, go to bed 1, go to drawer 1, go to drawer 2, go to drawer 3, go to drawer 4, go to drawer 5, go to dresser 1, go to garbagecan 1, inventory, look, put keychain 3 in/on desk 1, use desklamp 1.
Please select an action from the admissible actions. Please just output the selected action:
LLM Output to Action: 1: put keychain 3 in/on desk → put keychain 3 in/on desk
2: examine keychain 3 → examine keychain 3
3: I will choose "put keychain 3 in/on desk" → put keychain 3 in/on desk
4: You just picked up the desklamp 1 → Randomly select an action, since no feasible action appears in the output
5: use desklamp 1 → use desklamp 1
We map the LLM's output to an executable action using a simple rule-based method, extracting the executable actions directly from the LLM's output.

D.3 Computational Resources and Hyperparameters

Our experiments are conducted on a single machine equipped with eight 48 GB A6000 GPUs, using PyTorch 2.1 with CUDA 12.4. Tables 3, 4, 5, 6, 7 and 8 report the main hyperparameters of our algorithms. For the likelihood, we approximate it as $p(\mathcal{O} = 1 \mid s, a) \propto \exp(Q(s, a)/\tau)$, where the hyperparameter τ is used in two contexts: (1) value-function-guided action posterior selection, and (2) reweighting the log-likelihood during LLM prior fine-tuning, as shown in Eq. 7. In practice, we tune these two parameters separately and denote them as τ_1 and τ_2, respectively. The horizon and memory capacity of environments are shown in Table 9.

Table 3: The hyperparameters on Overcooked(Tomato)

Baselines	Learning Rate	Epochs	Batch Size	Update Frequency	τ_1	τ_2	K	γ
Mem-EM$_{\text{w/ tune}}$	5e-4	3	16	10	0.1	0.5	5	0.8

Table 4: The hyperparameters on Overcooked(Salad)

Baselines	Learning Rate	Epochs	Batch Size	Update Frequency	τ_1	τ_2	K	γ
Mem-EM$_{\text{w/ tune}}$	5e-4	3	16	10	0.1	0.5	10	0.8

Table 5: The hyperparameters on ALFWorld(Pick)

Baselines	Learning Rate	Epochs	Batch Size	Update Frequency	τ_1	τ_2	K	γ
Mem-EM$_{\text{w/ tune}}$	5e-4	3	16	100	0.1	0.2	5	0.9

Table 6: The hyperparameters on ALFWorld(Examine)

Baselines	Learning Rate	Epochs	Batch Size	Update Frequency	τ_1	τ_2	K	γ
Mem-EM$_{\text{w/ tune}}$	5e-4	3	16	100	0.1	0.5	10	0.9

E Additional Results

E.1 Effect of the training objective.

Table 10 compares training objectives using the 2,285 memory entries collected at the 4,000-episode Mem-EM$_{\text{w/tune}}$ checkpoint. Given the same memory and initialization, Q-only weighting achieves comparable or better win rates than ρ-corrected weighting under both evaluation strategies, supporting its use without costly prior-probability estimation. Positive-only SFT uses only 204 positive-value entries, whereas Q-weighted fine-tuning uses all 2,285 entries with value-dependent weights. Both Q-weighted objectives outperform the fresh-adapter SFT baseline, although differences in initialization confound this comparison. Memory-guided action selection substantially improves SFT but offers smaller gains for the Q-weighted policies, suggesting that our training objective effectively distills information from the memory table into the action policy.

E.2 Generalization Ability on ALFWorld.

The generalization ability evaluation results on ALFWorld(Pick) are shown in Table 11. We evaluate the generalization ability of the finetuned LLM policy and the Q-value estimators including the Q-network in and the memory-driven Q-estimator defined in Eq. 3. In general, we consider the components below and their combinations: the pretrained Qwen2.5-7B, denoted as **LLM**; the Q network trained with DQN-Prior, denoted as $\mathbf{Q}_{\textbf{DQN-Prior}}$; the fine-tuned LLM following Mem-EM$_{\text{w/ tune}}$, denoted as $\mathbf{LLM}_{\textbf{Mem-EM}_{\textbf{w/ tune}}}$; and the Q estimator of Mem-EM$_{\text{w/ tune}}$, denoted as $\mathbf{Q}_{\textbf{Mem-EM}_{\textbf{w/ tune}}}$. For example, the Mem-EM$_{\text{w/tune}}$ generates a refined LLM and a memory table of (s, a, Q) tuples. The LLM$_{\text{Mem-EM}_{\text{w/tune}}}$ uses the refined LLM to directly generate a single action for

Table 7: Hyperparameters of Mem-EM$_{\text{w/tune}}$ on Game of 24.

Method	Learning rate	Epochs	Batch size	Update frequency	τ_1	τ_2	K	γ
Mem-EM$_{\text{w/tune}}$	10^{-4}	4	64	250	0.1	0.1	5	1.0

Table 8: Hyperparameters of Mem-EM$_{\text{w/tune}}$ on MiniF2F.

Method	Learning rate	Epochs	Batch size	Update frequency	τ_1	τ_2	K	γ
Mem-EM$_{\text{w/tune}}$	10^{-4}	2	16	200	0.1	0.1	10	0.9

execution. The LLM$_{\text{Mem-EM}_{\text{w/tune}}}$ + Q$_{\text{Mem-EM}_{\text{w/tune}}}$ uses the refined LLM to generate K action candidates and then selects one action based on value estimation following the memory table.

The results show that while all methods perform well on seen tasks, the Q-estimators achieve only modest improvements over the pretrained LLM on unseen tasks. This exposes the generalization limitations of Q-estimators, which are constrained by the BERT-base representations despite being effective on seen tasks. In contrast, LLM fine-tuning shows superior generalization ability on unseen tasks, demonstrating the necessity of LLM prior refinement. Combining the fine-tuned LLM with the memory-based Q-estimator further improves performance, achieving a over 75% performance gain than the pretrained LLM.

On unseen tasks, the phenomenon of the better performance of LLM$_{\text{Mem-EM}_{\text{w/ tune}}}$ + Q$_{\textbf{Mem-EM}_{\text{w/ tune}}}$ with $K = 5$ (following the training setting) compared to larger K may stem from the introduction of low-quality, noisy samples from the tail distribution of the refined LLM prior. Specifically, the refined LLM has distilled domain-specific decision-making knowledge and is capable of generating high-quality actions when $K = 5$. However, when K is increased, less relevant or noisy action candidates are more likely to be sampled from the refined LLM prior's low-probability tail. This inclusion of noisy actions degrades the final policy performance. Overall, these results indicate that K=5 balances sufficient exploration coverage with high-quality action candidates.

Case study The refined LLM has distilled domain-specific decision-making knowledge and is capable of generating high-quality actions when K is small (e.g., $K = 5$).

Example: In the state "You are carrying: a tissuebox 1..." with the task "put some tissuebox on coffeetable," the $K = 5$ action candidates (`'put tissuebox 1 in/on coffeetable 1'`, `'inventory'`, `'examine coffeetable 1'`) successfully include the optimal action (`put tissuebox 1 in/on coffeetable 1`).

E.3 Additional baselines on agentic tasks

To further demonstrate the sample efficiency of our memory-driven self-improvement framework, we compare with a memory-based LLM-agent approach Memorybank (Zhong et al., 2024), for which previous state-action pairs are stored in memory, and the LLM makes decisions based on retrieved state-relevant memory.

Although both our Mem-EM$_{\text{w/tune}}$ and MemoryBank involve memory for previous experiences, our Mem-EM$_{\text{w/tune}}$ outperforms MemoryBank on Overcooked(Tomato). The key difference lies in how memory is utilized: our method uses memory for value estimation and LLM prior refinement combined with a value-guided action policy, whereas MemoryBank simply uses retrieved memory as additional decision context for the LLM. On the more complex ALFWorld environment, we retrieved K=20 similar (s,a) pairs as input context to enhance the decision-making ability of the original LLMs (the same setting as our Mem-EM method). As shown in Figure 6, the Memorybank baseline, which uses retrieval memory as an enhanced prompt, performs significantly worse than our Mem-EM approach, which utilizes memory for value estimation and efficient search guidance. Furthermore, the Memorybank with Qwen2.5-3B performs similarly to the original base model, only showing better results when scaled up to Qwen2.5-7B. This clearly indicates that the effectiveness of the Memorybank method is highly limited by the foundation model size.

	ALFWorld (Pick)	ALFWorld (Examine)	Overcooked (Tomato)	Overcooked (Salad)	Game of 24	MiniF2F
Maximum horizon	60	60	15	30	3	20
Memory capacity	15,000	15,000	100	1000	4,000	4,000

Table 9: Maximum episode horizon and memory capacity across environments. Memory capacity denotes the maximum number of stored state–action entries.

Table 10: **Effect of the fine-tuning objective on Game of 24.** All methods use Qwen3-8B Q-only and ρ-corrected fine-tuning use the same memory table and same LLM prior model initialization. Positive-only SFT instead trains a fresh adapter on the 204 positive-value entries.

Training objective	Test-time decision	Train win rate	Test win rate
Positive-only SFT	Policy greedy	14.86%	11.79%
Q-only weighting	Policy greedy	37.14%	**18.78%**
ρ-corrected Q weighting	Policy greedy	36.57%	17.90%
Positive-only SFT	Ten proposals + Q value	31.43%	17.03%
Q-only weighting	Ten proposals + Q value	36.57%	**19.65%**
ρ-corrected Q weighting	Ten proposals + Q value	37.14%	18.78%

Table 11: Results on the generalization ability of ALFWorld(Pick). All trainable models are trained with $K = 5$ action candidates, and their performance is evaluated using different values of K.

Baseline	Unseen Tasks				Seen Tasks			
	K=5	K=10	K=15	K=20	K=5	K=10	K=15	K=20
LLM	0.19				0.1			
LLM + $Q_{\text{DQN-Prior}}$	0.16	0.19	0.14	**0.22**	0.6	**0.70**	**0.70**	0.65
LLM + $Q_{\text{Mem-EM}_{\text{w/tune}}}$	0.22	0.27	**0.35**	0.22	0.65	0.80	**0.85**	0.65
$\text{LLM}_{\text{Mem-EM}_{\text{w/tune}}}$	0.59				0.85			
$\text{LLM}_{\text{Mem-EM}_{\text{w/tune}}}$ +$Q_{\text{DQN-Prior}}$	**0.59**	0.54	0.46	0.46	**0.90**	0.85	**0.90**	0.75
$\text{LLM}_{\text{Mem-EM}_{\text{w/tune}}}$ +$Q_{\text{Mem-EM}_{\text{w/tune}}}$	**0.81**	0.65	0.62	0.68	0.95	0.95	**1.00**	0.95

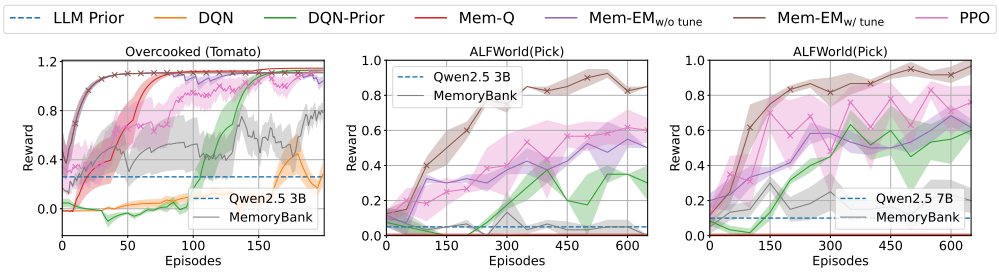


Figure 6: Additional Results of comparison with MemoryBank. We plot the mean and standard error of the cumulative reward. The dashed line represents directly prompting the LLM prior to generating actions given state information, with the corresponding LLM version specified. The '$\times$' markers for $\text{Mem-EM}_{\text{w/ tune}}$ indicate the time steps when the LLM prior is fine-tuned. The baselines in each subfigure are run on the identical foundation LLM, which is explicitly noted in the legend.

E.4 Testing across Different Foundation LLMs

To validate the effectiveness of our method across different foundation LLMs, we fine-tune three models: Qwen2.5-3B, Qwen2.5-7B, and LLaMA3-8B. We use the memory table obtained by performing $\text{Mem-EM}_{\text{w/tune}}$ with Qwen2.5-7B on ALFWorld(Pick), collecting approximately 6,000 (s, a, Q) tuples. The three LLMs are then fine-tuned using our proposed LLM prior refinement objective function (Eq. 7) with this memory table. As shown in Table 12, our LLM fine-tuning objective is adaptable to different LLMs, which demonstrates that the memory gathered using Qwen2.5-7B can be easily distilled into both the smaller model Qwen2.5-3B and a model with a different architecture, LLaMA3-8B.

Table 12: Results of fine-tuning different LLMs on ALFWorld(Pick).

	Without Finetune	Finetuned
Qwen2.5-3B	0.05	0.85
Qwen2.5-7B	0.10	0.88
LLaMA3-8B	0.0	0.83

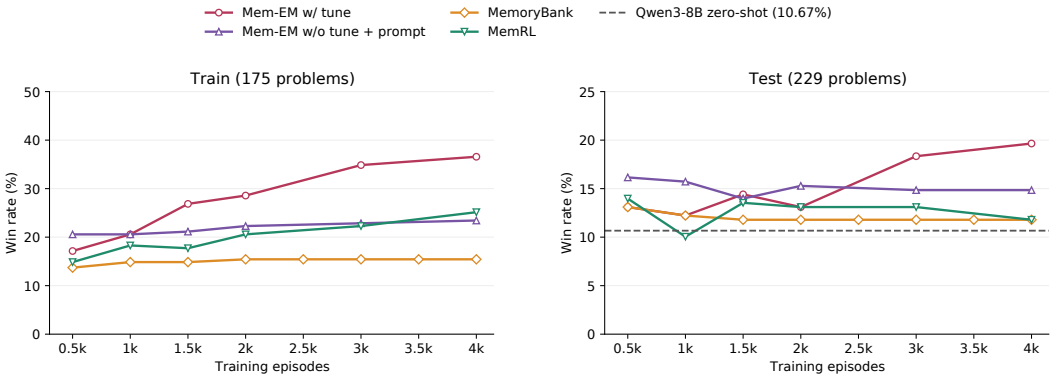


Figure 7: **Different mechanisms for utilizing experience on Game of 24.** Training- and test-set win rates across interaction checkpoints. Mem-EM$_{\text{w/tune}}$ distill memory into the LLM policy, whereas prompt-conditioned Mem-EM$_{\text{w/o tune}}$, MemoryBank, and MemRL reuse experience through context at inference.

E.5 Ablation on Experience Reuse

Figure 7 contrasts policy fine-tuning with inference-time memory reuse. Prompt-conditioned Mem-EM benefits early from retrieved experience, but its test performance subsequently plateaus while training performance continues to improve slightly, suggesting limited transfer of additional experience. In contrast, fine-tuning improves both training and test performance at later checkpoints. These trends highlight the importance of the finetuning-based M-step. Storing and retrieving more experience alone may be insufficient, whereas consolidating experience into the policy can improve generalization to unseen problems.

The prompt-based variants also differ in how they represent and utilize experience. MemoryBank retrieves examples based on state similarity without explicit value estimation, whereas MemRL assigns outcome-based utility to episodic (trajectory-level) memories to guide retrieval. Its Q-value estimation is indirect: the utility reflects the downstream return obtained when a memory is used as an in-context prompt, and thus depends on both memory-query relevance and the LLM's ability to exploit the retrieved experience. In contrast, Mem-EM stores state-action-level experiences with environment-grounded values and directly estimates $Q(s, a)$ for candidate action selection, while the same memories can also be used for prompting. This establishes a shorter credit-assignment path from environmental feedback to action selection.

E.6 Computational Resource Requirements

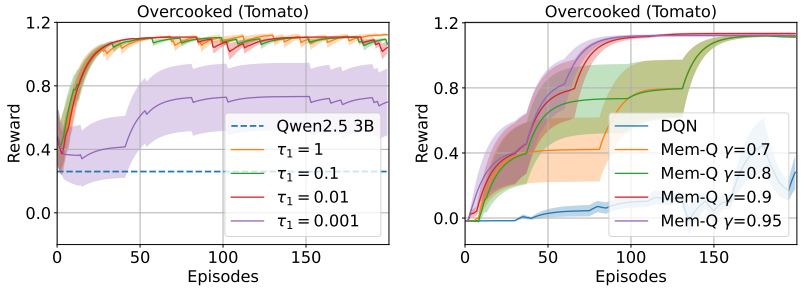
Table 13 summarizes the computational costs for the baselines on the ALFWorld (Pick) task using Qwen2.5-3B. Our method achieves significant performance gains over PPO and DQN-Prior while maintaining training time and GPU overhead comparable to PPO. Specifically, with only 1.15$\times$ the GPU usage and nearly identical training time, our approach delivers a 1.5$\times$ performance improvement over the best-performing PPO baseline.

E.7 Additional results on the hyperparameter tuning

Ablation on exploration trade-off hyperparameter We have conducted an experiment on the exploration trade-off hyperparameter τ_1. Specifically, we tuned this hyperparameter of our Mem-EM$_{\text{w/o tune}}$ on Overcooked(Tomato) with Qwen2.5-3B. As shown in Figure 8(a), the hyperpa-

Table 13: Computational Resource on ALFWorld(Pick) with Qwen2.5-3B. We record the inference time by querying baselines for 1000 times.

	PPO	DQN-Prior	Mem-EM$_{\text{w/ tune}}$	Mem-EM$_{\text{w/o tune}}$
Performance	0.62	0.38	**0.93**	0.55
Training Time	11.05h	12.07h	11.20h	13.57h
GPU usage	13.34G	8.78G	15.45G	7.83GM
Memory table size	/	/	7100	7554
Memory CPU usage	/	/	57.42M	61.12M
Inference Time(min)	3.40	3.59	5.28	8.16



(a) Exploration parameter τ_1 (b) Discount factor γ

Figure 8: Hyperparameter ablations on Overcooked (Tomato): (a) τ_1 in Mem-EM$_{\text{w/o tune}}$ with Qwen2.5-3B; (b) γ in Mem-Q.

rameter τ_1 is not as sensitive as we imagined, and the algorithm fails to learn only under the setting of $\tau_1 = 0.001$, which is extremely greedy and poor for exploration.

Ablation on discount factor γ We conduct an ablation study on the discount factor γ using Overcooked(Salad) with dense rewards. As shown in Figure 8(b), our Mem-Q demonstrates robustness to different values of γ.