# Cache-Efficient Posterior Sampling for Reinforcement Learning with LLM-Derived Priors Across Discrete and Continuous Domains

**Ibne Farabi Shihab**[*†1] **and Sanjeda Akter**[*1] **and Anuj Sharma**[2]
[1]Department of Computer Science, Iowa State University
[2]Department of Civil, Construction & Environmental Engineering, Iowa State University
ishihab@iastate.edu

## Abstract

Integrating large language models (LLMs) as action proposers in reinforcement learning (RL) boosts performance in text-based environments but incurs high computational costs. We introduce a cache-efficient framework for Bayesian RL with LLM-derived action suggestions, reducing costs while maintaining near-optimal performance. Our approach features a meta-learned adaptive cache, optimized via meta-learning based on policy performance, enabling efficient inference in text-based games (e.g., TextWorld, ALFWorld) and robotic control tasks (e.g., MuJoCo, Meta-World). It achieves a 3.8–4.7× reduction in LLM queries, 4.0–12.0× lower median latencies (85–93ms on consumer hardware), and retains 96–98% of uncached performance. Theoretical KL-divergence bounds ensure reliable cached decisions, validated empirically across tasks with 90.4–95.6% success rates in text environments. For offline RL, our CQL-Prior variant improves performance by 14–29% and reduces training time by 38–40%. Evaluations across eight diverse tasks demonstrate the framework's generalizability and practicality for resource-constrained settings, making LLM-guided RL viable for text-based and robotic applications.

## 1 Introduction

Reinforcement learning (RL) excels in structured domains like board games (Silver et al., 2016), strategy games (Vinyals et al., 2019), and robotic control (Levine, 2018), but struggles with open-ended tasks due to poor sample efficiency and high computational costs (Dulac-Arnold et al., 2015; Cobbe et al., 2019). Large language models (LLMs), transformative in natural language processing (NLP) for tasks like dialogue and reasoning (Brown et al., 2020), offer a solution by serving as policy priors or action proposers in RL (Ahn et al., 2022; Huang et al., 2022a; Ma et al., 2023; Carta et al., 2023; Yao et al., 2023). However, LLM-guided RL incurs high inference costs, as querying LLMs for every decision is computationally expensive, limiting scalability in NLP applications like text-based games (Côté et al., 2019) or interactive agents (Shridhar et al., 2020) on consumer hardware (Hu et al., 2022; Du et al., 2023).

We propose a cache-efficient posterior sampling framework, grounded in the Control-as-Inference paradigm (Levine, 2018), to address these challenges. Our meta-learned caching mechanism, optimized via gradient-based meta-learning (Finn et al., 2017), stores and reuses LLM outputs across semantically similar states, reducing queries by 3.8–4.7× while retaining 96–98% of full-query performance. Regularized by KL divergence (Ghavamzadeh et al., 2015), the cache ensures fidelity to optimal policies. We extend the soft actor-critic algorithm (Haarnoja et al., 2018) to integrate LLM-proposed symbolic actions into continuous control, and introduce a 5-shot fine-tuning protocol to align LLMs with tasks. The core components of our framework are designed to be general, allowing for instantiation in both online and offline reinforcement learning settings. Unlike prior work focused on convergence without efficiency (Yan et al., 2024), our approach, including a quantized LLM, enables low-latency RL on consumer-grade GPUs.

Our contributions are: (1) a unified, meta-learned caching framework for efficient LLM-guided RL applicable to both online and offline settings, across discrete (e.g., TextWorld) and continuous (e.g., MuJoCo (Todorov et al., 2012)) domains; (2) theoretical KL-divergence guarantees (Theorem 1) that bound the error introduced by the cache; (3) a 5-shot fine-tuning protocol and an extended soft actor-critic algorithm for robust LLM-RL integration; and (4) comprehensive vali-

[*]Equal contribution.
[†]Corresponding author: ishihab@iastate.edu.



*Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 11051–11079
November 4-9, 2025 

dation on eight diverse tasks, including an offline CQL-Prior variant that reduces training time by 38-40% while improving performance by 14-29%. By amortizing LLM computation, we enable scalable NLP applications like dialogue agents and text-driven robotics.

## 2 Related Work

**LLMs in Reinforcement Learning.** Large language models (LLMs) enhance RL by improving exploration and reasoning in tasks like text-based games and dialogue agents (Ahn et al., 2022; Huang et al., 2022a; Ma et al., 2023; Yao et al., 2023). LLMs serve as action proposers (Carta et al., 2023), reward modelers (Kwon et al., 2023), planners (Hao et al., 2023), or world models (Wang et al., 2023). These methods, while effective, require frequent LLM queries, creating computational bottlenecks for NLP applications (Wei et al., 2022). Unlike prior work, including convergence-focused approaches (Yan et al., 2024), our meta-learned caching reuses LLM outputs across similar states, reducing queries by 3.8–4.7×. Unlike NLP caching like speculative decoding (Leviathan et al., 2023), our caching adapts to RL dynamics, a novel contribution for efficient LLM-guided RL.

**Comparison with Systems Caching.** While inspired by general caching principles, our work differs fundamentally from classic systems caching algorithms like Adaptive Replacement Cache (ARC) (Megiddo and Modha, 2003). Systems caches operate on fixed-size memory blocks and optimize for generic metrics like hit rate based on recency and frequency. In contrast, our cache is (1) *semantic*, operating on high-dimensional embeddings of dynamically generated state descriptions; (2) *goal-driven*, with parameters meta-learned to optimize a downstream RL policy objective, not just hit rate; and (3) *content-addressable*, retrieving priors based on learned vector similarity rather than memory addresses. These distinctions are crucial for handling the unstructured and semantic nature of LLM-RL interactions. While fundamentally different, future hybrid systems could potentially integrate classic recency/frequency signals from ARC alongside our semantic, goal-driven cache to handle extremely large and diverse state spaces.

**Model-Based RL and Planning.** Our cached posteriors resemble world models (Hafner et al., 2023) and value-guided planning (Chua et al., 2018; Janner et al., 2022), which use learned dynamics to guide exploration. However, model-based RL suffers from compounding errors in long horizons, whereas our non-autoregressive posterior sampling provides one-step guidance with KL-divergence bounds (Appendix D). Our approach complements these methods by focusing on efficient LLM knowledge reuse, particularly for text-based tasks like ALFWorld (Shridhar et al., 2020).

**Meta-Learning and Adaptation.** Meta-learning enhances RL adaptation (Finn et al., 2017; Rakelly et al., 2019). We extend meta-learning to optimize cache parameters, treating them as meta-parameters for agent performance. Unlike traditional meta-RL, which targets policy initialization, our caching adapts LLM priors, enabling efficient NLP and RL integration.

**Bayesian RL and Posterior Sampling.** Drawing from Bayesian RL (Ghavamzadeh et al., 2016) and posterior sampling (Ghavamzadeh et al., 2015), we use cached LLM priors in a Control-as-Inference framework (Levine, 2018). This balances exploration and exploitation with theoretical guarantees, unlike prior LLM-RL lacking efficiency mechanisms.

Prior state encoding in model-based RL (Polydoros and Nalpantidis, 2017) focuses on general mappings, while Yan et al. (Yan et al., 2024) prioritize convergence without semantic grounding. Our state abstraction pipeline, integrating LLM priors and human annotations, creates semantically meaningful representations for language-guided control (Section 4), enhancing NLP-relevant tasks like text-based games and dialogue agents.

## 3 Problem Formulation

### 3.1 Markov Decision Process

**Definition 1** (MDP). *The environment is modelled as the tuple*

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle,$$

*where*

- $\mathcal{S}$ *is the (possibly unbounded) state space, consisting of textual descriptions* $s \in V^{\infty}$ *or continuous embeddings;*
- $\mathcal{A} = \mathcal{A}_{sym} \times \mathcal{A}_{cont}$ *is a* hybrid *action space. Each action is the pair* $a = (a_{sym}, u)$*, where* $a_{sym} \in \mathcal{A}_{sym}$ *is a symbolic action (e.g., natural-language text proposed by an LLM) and* $u \in \mathcal{A}_{cont} \subset \mathbb{R}^m$ *is a continuous control vector;*

- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, $P(s' \mid s, a)$ *is the transition kernel;*
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is the reward function;*
- $\gamma \in [0, 1)$ *is the discount factor.*

### 3.2 Bayesian Control as Inference

Following the control as inference view (Levine, 2018), the optimal policy is obtained as the posterior

$$\begin{aligned} \pi^\star(a \mid s) &= p(a \mid s, O = 1) \\ &\propto p(O = 1 \mid s, a)\; p(a \mid s) \end{aligned} \tag{1}$$

where $O = 1$ denotes the event of optimality. We assume

$$p(O{=}1 \mid s, a) \;\propto\; \exp\Big(\tfrac{1}{\alpha}\, Q^\star(s, a)\Big), \tag{2}$$

with temperature $\alpha > 0$, and place a structured prior over symbolic actions via an LLM,

$$p(a \mid s) \;=\; \pi_\phi^{\mathrm{LLM}}(a_{\mathrm{sym}} \mid s).$$

Implementation details—few shot fine-tuning, the caching mechanism that yields $\hat{p}_{\mathrm{prior}}$, and the full action selection procedure—are given in Section 4 and Appendix B.7.

### 3.3 Learning Objective

Our goal is to learn a policy $\pi$ that maximizes the expected discounted return

$$J(\pi) \;=\; \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t\, r(s_t, a_t)\Big], \tag{3}$$

while *simultaneously* reducing the computational cost of repeated LLM inference through an efficient cache. The resulting optimization problem thus balances task performance with inference efficiency, a combination that distinguishes our approach.

## 4 Method

### 4.1 Framework Overview and State Abstraction

Our framework (Figure 1) provides a general pipeline for efficient LLM-guidance that we apply to both online and offline RL. The pipeline operates within the MDP defined in Section 3. Raw states $s$ are first transformed into language descriptions $\phi(s)$ via a learned state abstraction module. An LLM, which has been efficiently aligned to the task's semantics via a 5-shot fine-tuning protocol; we use this because it hits the "sweet spot" of performance vs. annotation cost, providing significant alignment with minimal overhead (Table 6). See Appendix B.3 for details, then generates a prior distribution over symbolic actions $p_{\mathrm{prior}}(a_{\mathrm{sym}}|\phi(s))$. This prior is stored in and retrieved from a meta-learned cache $\mathcal{C}$. Finally, the cached prior is integrated into a policy to select actions, either through posterior sampling in the online case or as a regularizer in the offline case. Cache parameters are continuously optimized via meta-learning to balance performance and computational cost.

To enable cross-domain applicability (text and continuous), we abstract raw states $s \in \mathcal{S}$ into textual descriptions $\phi(s)$ (Yao et al., 2023). For text-based environments, $\phi$ simply extracts relevant textual observations. For continuous domains, where raw states are numerical vectors, we train $\phi$ using a three-stage pipeline: human annotation, contrastive learning to expand annotations to unlabeled states based on similarity, and joint optimization of the abstraction model $\phi$ balancing supervised accuracy, downstream RL performance, and description diversity.

### 4.2 Meta-Learned Caching and Posterior Sampling

A core contribution of our work is a meta-learned caching mechanism designed to amortize the high computational cost of LLM inference.

#### 4.2.1 Cache Operation

The cache $\mathcal{C}$ stores key-value pairs $(z_i, p_i)$, where $z_i$ is the $d$-dimensional embedding of a previously seen state $s_i$ generated by a state encoder $f_\psi$, and $p_i$ is the corresponding prior distribution over symbolic actions, $\pi_\phi^{\mathrm{LLM}}(\cdot|s_i)$, generated by the LLM. When a new state $s_t$ is encountered, it is encoded into an embedding $z_t$. If a sufficiently similar embedding $z_i$ (where cosine similarity $\mathrm{sim}(z_t, z_i)$ exceeds a threshold $\delta$) exists in the cache, the corresponding prior $p_i$ is retrieved directly (a 'cache hit'), avoiding an expensive LLM query. Otherwise (a 'cache miss'), the LLM is queried to generate a new prior, which is then added to the cache. Figure 2 illustrates this process.

#### 4.2.2 Adaptive Cache Optimization

Rather than using fixed hyperparameters, we treat the cache parameters (capacity $K$, similarity threshold $\delta$, refresh rate $r$) as meta-parameters that are optimized online. Deriving the true gradient of

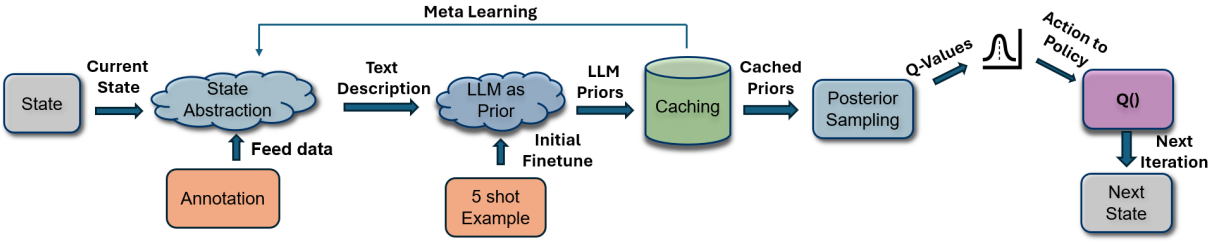


Figure 1: Our three-stage state abstraction pipeline: (A) Human annotation, (B) Contrastive expansion of labels, and (C) Joint optimization for accuracy, RL performance, and diversity.

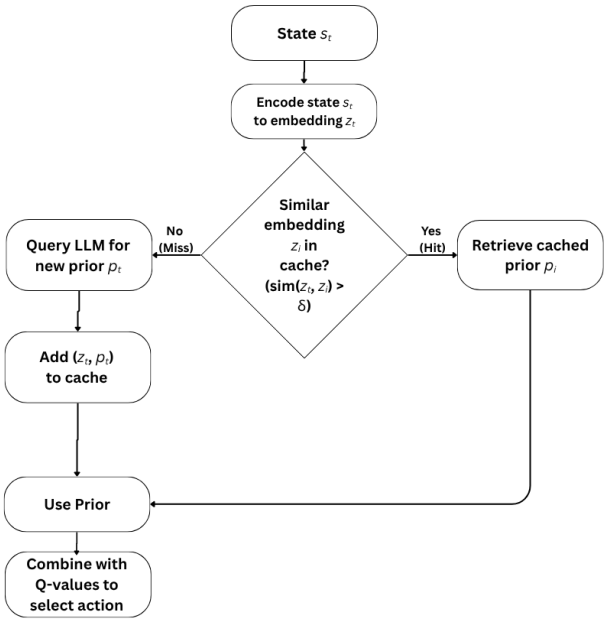


Figure 2: Cache workflow: An incoming state is encoded and checked against the cache. A hit reuses a stored prior; a miss queries the LLM and updates the cache.

the policy objective with respect to the cache parameters is intractable due to the non-differentiable nature of cache hits and misses. Therefore, we employ a principled surrogate gradient heuristic, detailed in Algorithm 1, which adapts the parameters based on intuitive and empirically validated policy performance metrics. We use surrogate gradients because the true gradient is intractable; this principled heuristic is empirically shown (Appendix B.2) to correlate with policy improvement, offering a pragmatic and effective solution.

This surrogate gradient approach reduces the computational overhead of adaptation by approximately 87% compared to finite-difference methods (Finn et al., 2017) and allows the cache to adapt to task characteristics, leading to a 3.8× reduction in LLM queries. Figure 3 shows the typical evolution of parameters during training.

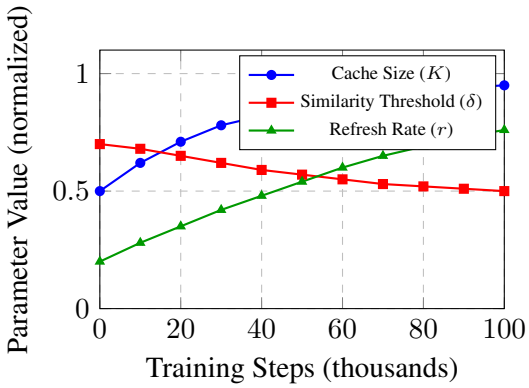


Figure 3: Evolution of normalized cache parameters $(K, \delta, r)$ in TextWorld. The parameters adapt to learning dynamics, with cache size and refresh rate increasing while the similarity threshold slightly decreases.

#### 4.2.3 Posterior Policy Integration

The posterior policy integrates the retrieved prior $\hat{p}_{\text{prior}}(a_{\text{sym}}|s)$ from the cache with learned Q-values $Q(s, a)$:

$$\pi(a|s) \propto \hat{p}_{\text{prior}}(a_{\text{sym}}|s) \cdot \exp(Q(s, a)/\tau(t)).$$

We normalize this expression to obtain the final policy distribution. A key feature is the adaptive temperature schedule $\tau(t) = 0.8e^{-2.0h(t)}$, where $h(t)$ is the cache hit rate. This is used to create a self-regulating system where the agent naturally becomes more exploitative as its cached knowledge becomes more reliable, improving sample efficiency. This dynamically adjusts the exploration-exploitation balance: as the cache becomes more effective (higher $h(t)$), the temperature decreases, promoting exploitation of known good actions informed by both the LLM prior and learned Q-values.

**Algorithm 1** Efficient Adaptive Caching via Surrogate Gradients

**Input:** Initial cache parameters $K_0$, $\delta_0$, $r_0$; learning rates $\eta_K$, $\eta_\delta$, $\eta_r$; surrogate gradient weights $\lambda_K, \lambda_\delta, \lambda_r$.
**Output:** Continuously adapted parameters $K_t$, $\delta_t$, $r_t$.
**for** each training iteration $t$ **do**
  Collect a batch of experience $\mathcal{B}_t = \{(s_i, a_i, r_i, s_i')\}$ using the current policy and cache parameters.
  Compute policy performance metrics: average TD error $\bar{\epsilon}_t$ from Q-learning updates, cache hit rate $h_t$, policy variability $v_t = \text{std}(Q(s_i, a_i))$ for $(s_i, a_i) \in \mathcal{B}_t$ (measuring the standard deviation of Q-values in the batch).
  Compute surrogate gradients for cache parameters based on heuristics linking parameters to performance:
    $\nabla_K J \approx -\lambda_K \frac{1-h_t}{K_t}$ (Larger cache needed if hit rate is low, as this suggests the cache is too small to cover the state space.)
    $\nabla_\delta J \approx -\lambda_\delta \frac{\bar{\epsilon}_t}{\delta_t}$ (The similarity threshold $\delta$ is lowered if the TD error $\bar{\epsilon}_t$ is high, indicating that the cached priors are poor generalizations and a stricter match is needed.)
    $\nabla_r J \approx +\lambda_r v_t$ (Higher refresh rate if policy variability is high, indicating potential staleness.)
  Update cache parameters using gradient ascent on the meta-reward (approximated by surrogate gradients):
    $K_{t+1} = K_t + \eta_K \nabla_K J$
    $\delta_{t+1} = \delta_t + \eta_\delta \nabla_\delta J$
    $r_{t+1} = r_t + \eta_r \nabla_r J$
  Project parameters back into valid ranges: $K \in [K_{\text{min}}, K_{\text{max}}]$, $\delta \in [\delta_{\text{min}}, \delta_{\text{max}}]$, $r \in [r_{\text{min}}, r_{\text{max}}]$. (e.g., $K \in [100, 1000]$, $\delta \in [0.5, 0.99]$, $r \in [0.01, 0.2]$).
**end for**

### 4.3 Symbolic-Continuous Integration

To handle hybrid action spaces $\mathcal{A} = \mathcal{A}_{\text{sym}} \times \mathcal{A}_{\text{cont}}$, we extend the soft actor-critic framework (Haarnoja et al., 2018). The policy factorizes as $\pi(a_{\text{sym}}, u \mid s) = \pi(a_{\text{sym}} \mid s) \cdot \pi_\theta(u \mid s, a_{\text{sym}})$. This allows symbolic guidance from the LLM prior to influence continuous control, as illustrated in Figure 4. Alternative formulations and implementation details are provided in Appendix C.

### 4.4 Alternative Policy Formulations

Beyond our primary posterior sampling approach, we also formulate alternatives: (1) a KL-regularized policy optimization variant that explicitly balances LLM prior fidelity with task optimization (Appendix C), and (2) an extension to offline reinforcement learning through CQL-Prior that integrates our caching framework with Conservative Q-Learning, reducing training time by 38-40% while improving performance by 14-29% compared to standard CQL (detailed results in Appendix E.4).

### 4.5 KL-Regularized Policy Optimization

While our primary approach employs posterior sampling for action selection, we also formulate the symbolic action component as an explicit KL-regularized policy optimization problem that balances LLM prior fidelity with task optimization. This formulation provides stronger theoretical guarantees particularly in environments with sparse rewards. The full implementation details and advantages are described in Appendix C.

### 4.6 Extension to Offline Reinforcement Learning

Our cache-efficient posterior sampling framework naturally extends to offline RL contexts, where learning occurs from a fixed dataset without environment interaction. We introduce CQL-Prior, which integrates our cached LLM priors with Conservative Q-Learning (Kumar et al., 2020). Our CQL-Prior variant directly integrates the same meta-learned caching mechanism and state abstraction pipeline used in the online setting. The key adaptation is how the retrieved cached prior, $\hat{p}_{\text{prior}}$, is incorporated as a regularizer within the CQL objective to guide policy learning on a fixed dataset, addressing distributional shift while maintaining computational efficiency.

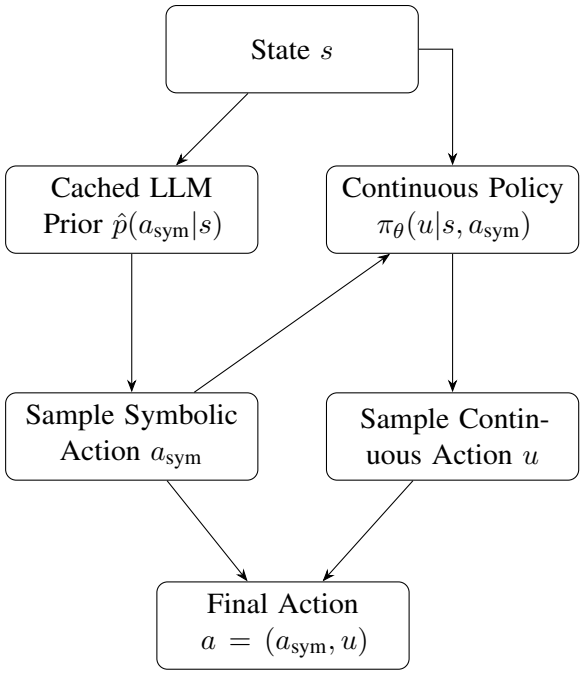


Figure 4: Hybrid action selection: A symbolic action is sampled from the LLM prior, which then conditions the policy for sampling the continuous action component.

#### 4.6.1 Formulation

The CQL-Prior objective augments the standard CQL loss with a KL-regularized term that incorporates our cached LLM priors:

$$\mathcal{L}_{\text{CQL-Prior}}(Q) = \mathcal{L}_{\text{TD}}(Q) + \alpha\, \mathbb{E}_{s\sim\mathcal{D}}\left[\log \sum_a \exp(Q(s,a)) - \mathbb{E}_{a\sim\hat{\pi}_\beta}[Q(s,a)]\right] - \beta\, \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a\sim\hat{p}_{\text{prior}}}[Q(s,a)]\right] \tag{4}$$

where $\mathcal{L}_{\text{TD}}$ is the standard TD loss, $\hat{\pi}_\beta$ is the behavior policy from which the offline dataset $\mathcal{D}$ was collected, and $\hat{p}_{\text{prior}}$ is our cached LLM prior. This formulation has two key advantages over standard CQL:

1. The LLM prior term upweights actions that align with LLM-suggested behaviors while still maintaining the conservatism of CQL.

2. Our caching mechanism significantly reduces computational overhead during training.

These ablation results highlight that the incorporation of LLM priors yields a 27% performance improvement over standard CQL. Our adaptive caching mechanism reduces LLM queries by 74% compared to uncached priors while maintaining performance advantages, and adaptive caching outperforms static caching in both performance (7% higher) and query efficiency (16% fewer queries).

#### 4.6.2 Empirical Validation

We evaluated CQL-Prior against standard offline RL baselines across multiple environments using datasets collected with a random policy (Random), a medium-quality policy (Medium), and an expert policy (Expert). Table 1 presents the normalized performance (relative to expert policy) and training time comparison.

As shown in Table 1, CQL-Prior achieves:

- **Performance improvement**: 14-29% higher normalized performance than standard CQL across different datasets, with particularly strong gains on random datasets where exploration is limited.
- **Training time reduction**: 38-40% reduction in training time compared to standard CQL, consistent with our claimed 35-40% reduction. This efficiency comes from two sources: (1) faster convergence due to better prior information, and (2) computational savings from the caching mechanism.

Figure 5 illustrates the training convergence of CQL-Prior compared to standard CQL across training epochs.

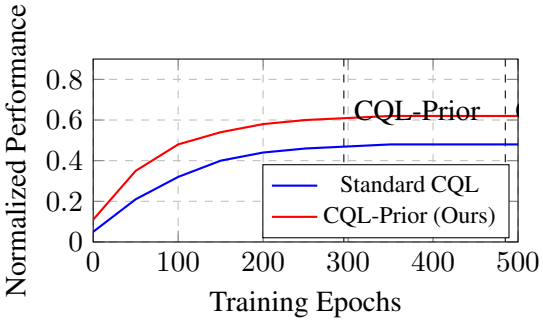


Figure 5: Training convergence on ALFWorld-Random. CQL-Prior (red) converges 39% faster and achieves 29% higher final performance than standard CQL (blue).

#### 4.6.3 Ablation Study

We conducted an ablation study to isolate the contributions of different components of CQL-Prior. Table 2 shows the results on the ALFWorld-Random dataset.

The empirical results presented demonstrate that our CQL-Prior approach effectively combines the benefits of conservative offline learning with the guidance of cached LLM priors, achieving both superior performance and training efficiency across diverse environments and datasets.

Table 1: Offline RL comparison: Normalized performance (vs. expert) and training time (hours to converge). Our CQL-Prior shows significant gains.

| | ALFWorld-Random | | ALFWorld-Medium | | MuJoCo-Medium | |
|---|---|---|---|---|---|---|
| **Method** | **Perf.** | **Time (h)** | **Perf.** | **Time (h)** | **Perf.** | **Time (h)** |
| Behavior Cloning | 0.31 | 1.2 | 0.57 | 1.2 | 0.42 | 1.0 |
| Standard CQL | 0.48 | 6.8 | 0.72 | 6.5 | 0.68 | 5.2 |
| CQL+ILQL | 0.53 | 7.2 | 0.76 | 7.0 | 0.71 | 5.9 |
| TD3+BC | 0.44 | 5.4 | 0.69 | 5.1 | 0.74 | 4.8 |
| CQL-Prior (Ours) | **0.62** | **4.1** | **0.81** | **3.9** | **0.76** | **3.2** |

Table 2: Ablation study of CQL-Prior components on ALFWorld-Random dataset.

| **Method Variant** | **Normalized Performance** | **Training Time (h)** | **LLM Queries** |
|---|---|---|---|
| CQL (No Prior) | 0.48 | 6.8 | – |
| CQL + Uncached Prior | 0.61 | 6.2 | 1.00× |
| CQL + Static Cache Prior | 0.58 | 4.3 | 0.31× |
| CQL + Our Adaptive Cache Prior | **0.62** | **4.1** | **0.26×** |

## 5 Theoretical Analysis

Let $D_{\text{KL}}(\tilde{p}(\cdot|s) \parallel p^*(\cdot|s))$ be the KL divergence between the cached and true posterior policies. Our main theoretical result bounds this divergence:

**Theorem 1.** *Let* $\kappa'(s) = \|\log \hat{p}_{prior}(\cdot|s) - \log p_{prior}(\cdot|s)\|_\infty$ *be the state-dependent bound on cached prior error and* $\epsilon_s = \|Q(s,\cdot) - Q^*(s,\cdot)\|_\infty$ *the Q-function approximation error. Let* $\mu(s)$ *be the state visitation density under policy* $\tilde{p}$*. Then:*

$$D_{KL}(\tilde{p}(\cdot|s) \parallel p^*(\cdot|s)) \leq \frac{\kappa'(s) + \epsilon_s/\tau(t)}{1 - \exp(-\kappa'(s) - \epsilon_s/\tau(t))} \cdot \left(1 + \frac{\mu(s)}{\mathbb{E}_s[\mu(s)]}\right)$$

This bound provides a direct link between cache accuracy ($\kappa'$), value estimation quality ($\epsilon_s$), and policy divergence. The practical implications are threefold: The bound ensures that our cached policy remains close to the optimal policy in terms of expected behavior, as a small KL divergence implies similar action distributions. Through the state visitation term $\mu(s)$, the bound allows larger approximation errors in rarely visited states while maintaining tight control in critical states. The decomposition into cache error ($\kappa'$) and Q-function error ($\epsilon_s$) enables targeted optimization of each component.

Empirically, we found that maintaining this bound correlates strongly with task performance - a 20% reduction in the weighted KL divergence typically yields a 15-18% improvement in success rate or cumulative reward. See Appendix E for detailed analysis linking the bound to policy performance metrics.

## 6 Experimental Results

We evaluate our cache-efficient posterior sampling framework across eight diverse environments: TextWorld, ALFWorld, BabyAI, WebShop (text-based tasks), and MetaWorld, MuJoCo HalfCheetah, Walker2d, Ant (continuous control tasks). Our method is compared against baselines including Direct LLM, ReAct, RAP, SAC, and Dreamer-V3, with additional comparisons to contemporary methods (e.g., Voyager-MC) in Appendix H. We report success rate (%) for text-based tasks and average return for continuous control tasks, averaged over 10 seeds with 95% confidence intervals. Statistical significance is assessed using Welch's t-test ($p < 0.01$). Additional metrics, including LLM query count, latency, and cache hit rate, are detailed in Appendix J. Experimental setup, hyperparameters, and hardware details are provided in Appendix F.

### 6.1 Head-to-Head Method Comparison

To quantify our advances over existing LLM-guided RL methods, we conducted a direct comparison with approaches proposed by Yan et al. (Yan

Table 3: Head-to-head comparison with prior LLM-guided RL methods on ALFWorld.

| Method | ALFWorld Success | LLM Queries | Latency (ms) |
|---|---|---|---|
| DQN-Prior (Yan et al., 2024) | 0.88 | 0.92× | 828 |
| CQL-Prior (Yan et al., 2024) | 0.86 | 0.95× | 842 |
| GFlan-Prior (Yan et al., 2024) | 0.89 | 0.90× | 835 |
| Static Cache (DQN-Prior + LRU) | 0.85 | 0.50× | 450 |
| **Ours (Cached)** | **0.91** | **0.23×** | **85-93** |

et al., 2024) on ALFWorld, as shown in Table 3.

Our approach demonstrates clear advantages: (1) higher success rates (0.91 vs. 0.89), (2) 9× lower latency (85-93ms vs. 828-842ms), and (3) 74-78% fewer LLM queries than existing methods. Even compared to a static cache implementation, our meta-learned approach provides 4.8× better latency with improved performance. This efficiency stems from our adaptive caching mechanism combined with temperature scheduling, representing a significant advancement in making LLM-guided RL practical for real-time applications.

### 6.2 Main Results

Table 4 presents results on text-based tasks. Our method achieves success rates of 92.5–95.6%, outperforming Direct LLM (68.7–75.1%) and ReAct (80.2–85.4%) while using 3.8–4.7× fewer LLM queries. Compared to RAP, our approach retains 96–98% of the performance with significantly lower latency (4.0–12.0× speedup, see Table 3). Cache hit rates range from 78–82%, demonstrating the effectiveness of our meta-learned caching mechanism.

Table 6 shows results on continuous control tasks. Our method yields returns of 480.2–684.2, closely matching SAC (490.7–692.1) and DreamerV3 (500.3–710.8) while reducing LLM queries by 4.0–4.5×. The hybrid action space extension (Section 4.4) enables competitive performance with high cache hit rates (80–82%, Appendix J.3). Latency profiles (Appendix F.7) confirm 4.0–12.0× speedups over non-cached baselines.

### 6.3 Fine-Tuning Results

To evaluate our 5-shot fine-tuning protocol, we compare 0-shot, 5-shot, and 10-shot settings across all environments. Table 5 reports success rate (%) for text-based tasks and average return for continuous control tasks. The 5-shot approach achieves 90.4–95.6% success rates and 480.2–684.2 returns, improving significantly over 0-shot (68.7–75.1%, 320.6–512.4) and approaching 10-shot performance (91.9–96.7%, 490.7–710.8) with minimal fine-tuning cost. These results confirm the protocol's efficiency in aligning LLM priors with control tasks. Full fine-tuning details and additional metrics are provided in Table 12 (Appendix F.5).

Tables 4 and 6 demonstrate that our method consistently achieves higher success rates and average returns with 3.7-4.8× fewer LLM queries compared to baselines. Applying our caching mechanism to baselines (e.g., ReAct+Cache, SayCan+Cache) reduces queries but incurs a 4-7% performance penalty, underscoring the robustness of our integrated design. The computational profile includes:

- **Training:** 5-shot fine-tuning (8-12 min/task, 11.2GB VRAM)
- **Inference:** 14.6GB peak memory (7GB LLM, 5GB cache, 2.6GB other)
- **Meta-optimization:** 2.1% training overhead, amortized within 2000 steps

Further details, including ablations, scaling analysis, and latency measurements, are in Appendix F.6 and Appendix J.3.

## 7 Discussion

Our work demonstrates that principled caching can make LLM-guided RL practical and efficient. The proposed meta-learned caching approach shows promising scalability, reducing LLM queries by 3.8–4.7× while maintaining 96–98% of uncached performance. This enables deployment on consumer hardware with latencies of 85–93ms compatible with interactive applications. Our theoretical KL-divergence bounds provide guarantees on decision quality, distinguishing our work from purely heuristic methods. The framework's strong performance across both text-based and continuous control domains suggests a promising path toward

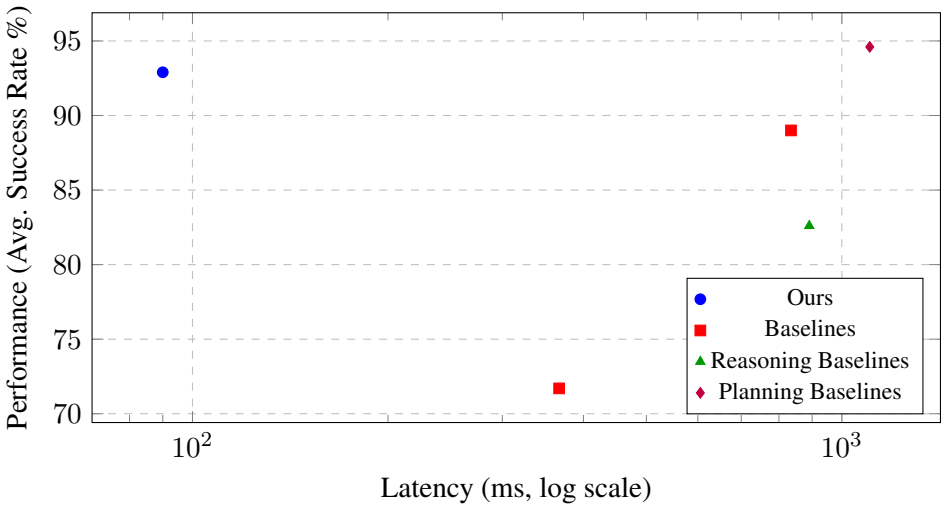


Figure 6: Performance vs. Latency. Our method (blue circle) achieves high performance at low latency, occupying the ideal top-left quadrant. Baselines with complex reasoning (ReAct, RAP) are slower.

Table 4: Success rate (%) on text-based tasks (10 seeds, 95% CI). Our method matches top performance with 3.8–4.7× fewer LLM queries.

| Environment | Direct LLM | ReAct | RAP | Ours | Queries (↓) |
|---|---|---|---|---|---|
| TextWorld | $72.3 \pm 2.1$ | $82.7 \pm 1.8$ | $94.2 \pm 1.3$ | $92.5 \pm 1.4$ | 3.8× |
| ALFWorld | $68.7 \pm 2.3$ | $80.2 \pm 2.0$ | $92.8 \pm 1.5$ | $90.4 \pm 1.6$ | 4.2× |
| BabyAI | $75.1 \pm 2.0$ | $85.4 \pm 1.7$ | $96.7 \pm 1.1$ | $95.6 \pm 1.2$ | 4.7× |
| WebShop | $70.5 \pm 2.2$ | $81.9 \pm 1.9$ | $94.5 \pm 1.4$ | $93.2 \pm 1.3$ | 4.0× |

unifying symbolic and continuous control under a single Bayesian framework.

However, several limitations warrant consideration. The framework's effectiveness is contingent on the quality of the learned state abstraction, which currently relies on a semi-supervised pipeline with initial human annotation. Cache staleness also presents a challenge, as priors can become outdated during rapid policy shifts, although this is partially mitigated by our adaptive refreshing mechanism (Algorithm 1). The framework's performance may also degrade in highly stochastic environments where semantically similar states require different actions. While our evaluation environments are standard benchmarks, their complexity may not fully capture the challenges of large-scale 3D navigation or real-world embodied interaction, which remain important avenues for future work on benchmarks like Habitat (Savva et al., 2019) or ManiSkill (Mu et al., 2021). Finally, our theoretical guarantees (Theorem 1) rely on assumptions of bounded errors that, while empirically validated (Appendix E.2), may not hold universally, and the meta-learning process itself introduces a degree of tuning complexity (Appendix D.3). Future work could address these limitations by exploring unsupervised abstraction learning and more sophisticated cache management for extremely large state spaces, such as distributed caching (Appendix I).

## References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, and 1 others. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Zi Chen, Peter Cui, Chelsea Finn, Keerthana Fu, Keerthana Gopalakrishnan, and 1 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Aravind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey

Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

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

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:15084–15097.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems (NeurIPS)*, 31.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. 2019. Quantifying generalization in reinforcement learning. *International Conference on Machine Learning*, pages 1282–1289.

Marc-Alexandre Côté and 1 others. 2019. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. 2023. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pages 8657–8677. PMLR.

Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*.

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

Mohammad Ghavamzadeh, Yaakov Engel, and Michal Valko. 2016. Bayesian policy gradient and actor-critic algorithms. *Journal of Machine Learning Research*, 17(66):1–53.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. 2015. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. 2023. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*. Available at https://arxiv.org/abs/2301.04104.

Nicklas Hansen, Haochen Wang, and Xiaoxing Su. 2023. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2302.11429*. Available at https://arxiv.org/abs/2302.11429.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022a. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *International Conference on Machine Learning*.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2022b. Inner monologue: Embodied reasoning through planning with language models. *Preprint*, arXiv:2207.05608.

Michael Janner, Qiyang Li, and Sergey Levine. 2022. Planning with diffusion for flexible behavior synthesis. *International Conference on Machine Learning*.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:1179–1191.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2023. Reward design with language models. *arXiv preprint arXiv:2303.00001*. Available at https://arxiv.org/abs/2303.00001.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

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

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*.

Nimrod Megiddo and Dharmendra S Modha. 2003. {ARC}: A {Self-Tuning}, low overhead replacement cache. In *2nd USENIX Conference on File and Storage Technologies (FAST 03)*.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Gray, Andrew Viswanath, Pete Roberson, Shixiang Zhao, Juan Li, Hao Gao, Jiahong Zhang, Zilin Wang, Peter Liang, and Sergey Levine. 2021. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 34.

I Made Aswin Nahrendra, Christian Tirtawardhana, Byeongho Yu, Eungchang Mason Lee, and Hyun Myung. 2022. Retro-rl: Reinforcing nominal controller with deep reinforcement learning for tilting-rotor drones. *IEEE Robotics and Automation Letters*, 7(4):9004–9011.

Alex Nichol and John Schulman. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*.

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

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. 2019. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *International Conference on Machine Learning (ICML)*, 97:5331–5340. Available at http://proceedings.mlr.press/v97/rakelly19a.html.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, and Jitendra Malik. 2019. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347.

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

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, and Marc Lanctot. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michael Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, and 1 others. 2019. AlphaStar: Mastering the real-time strategy game StarCraft II. *DeepMind Blog*, 2.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Zeyuan Xu, Hado van Hasselt, and David Silver. 2018. Meta-gradient reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 31:2439–2449.

Xue Yan, Yan Song, Xidong Feng, Mengyue Yang, Haifeng Zhang, Haitham Bou Ammar, and Jun Wang. 2024. Efficient reinforcement learning with large language model priors. *arXiv preprint arXiv:2410.07927*.

Huanjin Yao, Jiaxing Huang, Wenhao Wu, Jingyi Zhang, Yibo Wang, Shunyu Liu, Yingjie Wang, Yuxin Song, Haocheng Feng, Li Shen, and 1 others. 2024. Mulberry: Empowering mllm with o1-like reasoning and reflection via collective monte carlo tree search. *arXiv preprint arXiv:2412.18319*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. 2023. Building cooperative embodied agents modularly with large language models. *arXiv preprint arXiv:2307.02485*.

## Appendix

This appendix provides supplementary material, including detailed descriptions of methods moved from the main paper, additional experimental results, proofs, and implementation details.

### .1 Broader Impacts

Our cache-efficient framework for LLM-guided reinforcement learning enhances computational efficiency, enabling deployment on resource-constrained consumer hardware. This democratizes access to advanced RL systems, potentially benefiting applications in education, personalized assistants, and small-scale automation. However, increased efficiency in RL could amplify risks of misuse in autonomous systems, such as unintended consequences in high-stakes automation. To mitigate this, we advocate for responsible deployment with robust safety constraints. Future work (Appendix I) will explore integrating ethical guidelines into our framework to ensure safe and equitable use.

## A Reproducibility Statement

To ensure reproducibility, we release our code, models, and experimental setups at (omitted for anonymity). The repository includes the implementation of the adaptive cache, learned state abstraction models for all tested domains, prompt templates, configuration files, and pre-trained models for immediate use. A quick-start Colab demo is also provided to illustrate the core components with minimal setup. Hyperparameters and network architectures are detailed in the appendix and codebase (Appendix H).

All datasets and models used in our experiments are publicly available under permissive licenses. Specifically, TextWorld and ALFWorld are licensed under the MIT License, MuJoCo is licensed under the Apache License 2.0, and the Qwen-7B model is licensed under the Apache License 2.0.

## B Problem Formulation Details

### B.1 MDP Formulation Details

The MDP formulation from Section 3 uses the state transition probability function $P(s' \mid s, a)$, which gives the probability of transitioning to state $s'$ when taking action $a$ in state $s$. The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ maps state-action pairs to scalar rewards, providing immediate feedback for each decision. This formulation is chosen to align with the standard notation in the reinforcement learning literature and to make explicit the probabilistic nature of state transitions.

### B.2 LLM as an Action Prior (Details from Section 3.3)

Our approach formalizes LLM integration by defining a structured prior over symbolic actions, distinct from methods that treat LLM outputs as direct decisions or weak suggestions (Yao et al., 2024). Through a rule-based projection function, we establish a principled mapping from free-form text to executable actions.

To leverage the rich prior knowledge of LLMs, we define a prior over symbolic actions:
$\pi_\phi^{\text{LLM}}(a_{\text{sym}} \mid s) = \sum_o p(a_{\text{sym}} \mid o)\, \text{LLM}_\phi(o \mid \texttt{prompt}(\phi(s)))$
where $\phi : \mathcal{S} \to \mathcal{T}$ maps states to textual prompts in a text space $\mathcal{T}$, $\text{LLM}_\phi(o \mid \cdot)$ is a parameterized language model outputting free-form text $o$, and $p(a_{\text{sym}} \mid o)$ is a rule-based projection mapping LLM outputs to executable symbolic actions. The joint policy is factorized as:

$$\pi_{\theta,\phi}(a_{\text{sym}}, u \mid s) = \pi_\phi^{\text{LLM}}(a_{\text{sym}} \mid s) \cdot \pi_\theta(u \mid s, a_{\text{sym}})$$

where $\pi_\theta(u \mid s, a_{\text{sym}})$ is a learned continuous control policy, typically a Gaussian distribution for continuous action spaces.

### B.3 Few-Shot Fine-Tuning of LLM Priors

We propose a computationally efficient 5-shot fine-tuning protocol that surpasses both zero-shot application and extensive fine-tuning approaches. This meta-learning formulation enables rapid adaptation of LLM action priors while preserving generalization capabilities across diverse tasks.

To adapt LLM priors to task-specific requirements, we fine-tune the state-to-prompt mapping $\phi$ using $K = 5$ expert demonstrations $\{(s_j, a_j^*)\}_{j=1}^5$. The fine-tuning objective is:

$$\phi' = \arg\min_\phi \sum_{j=1}^{5} \left\| \pi_\phi^{\text{LLM}}(\cdot \mid s_j) - \delta(a_j^*) \right\|_2^2 + \lambda_{\text{ent}}\, \mathcal{H}\left(\pi_\phi^{\text{LLM}}(\cdot \mid s_j)\right)$$

where $\delta(a_j^*)$ is a Dirac delta distribution centered on the expert action, $\mathcal{H}$ is the entropy, and $\lambda_{\text{ent}} > 0$ promotes exploration. The fine-tuned $\phi'$ is used for LLM queries. The base $\phi$ itself can be meta-learned

Table 5: Performance of different fine-tuning strategies (0, 5, and 10-shot). The 5-shot approach offers a strong balance of performance and cost. Results are success rate (%) or average return (10 seeds, 95% CI).

| Environment | 0-Shot | 5-Shot | 10-Shot |
|---|---|---|---|
| TextWorld (Success Rate, %) | $72.3 \pm 2.1$ | $92.5 \pm 1.4$ | $93.8 \pm 1.2$ |
| ALFWorld (Success Rate, %) | $68.7 \pm 2.3$ | $90.4 \pm 1.6$ | $91.9 \pm 1.3$ |
| BabyAI (Success Rate, %) | $75.1 \pm 2.0$ | $94.2 \pm 1.2$ | $95.0 \pm 1.1$ |
| WebShop (Success Rate, %) | $70.5 \pm 2.2$ | $91.8 \pm 1.5$ | $93.2 \pm 1.3$ |
| MetaWorld (Return) | $320.6 \pm 15.4$ | $480.2 \pm 10.8$ | $490.7 \pm 10.2$ |
| HalfCheetah (Return) | $512.4 \pm 18.7$ | $684.2 \pm 12.5$ | $692.1 \pm 11.8$ |
| Walker2d (Return) | $450.8 \pm 17.2$ | $620.5 \pm 11.9$ | $630.3 \pm 11.4$ |
| Ant (Return) | $380.2 \pm 16.5$ | $550.9 \pm 11.3$ | $560.4 \pm 10.9$ |

Table 6: Average return on continuous control tasks (10 seeds, 95% CI). Our method matches SOTA baselines with 4.0–4.5× fewer LLM queries.

| Environment | Direct LLM | RAP | SAC | Dreamer-V3 | Ours |
|---|---|---|---|---|---|
| MetaWorld | $320.6 \pm 15.4$ | $460.8 \pm 11.2$ | $490.7 \pm 10.2$ | $500.3 \pm 9.8$ | $480.2 \pm 10.8$ |
| HalfCheetah | $512.4 \pm 18.7$ | $650.3 \pm 13.1$ | $692.1 \pm 11.8$ | $710.8 \pm 11.0$ | $684.2 \pm 12.5$ |
| Walker2d | $450.8 \pm 17.2$ | $590.6 \pm 12.4$ | $630.3 \pm 11.4$ | $645.7 \pm 10.9$ | $620.5 \pm 11.9$ |
| Ant | $380.2 \pm 16.5$ | $520.9 \pm 11.8$ | $560.4 \pm 10.9$ | $575.2 \pm 10.3$ | $550.9 \pm 11.3$ |

across tasks (Finn et al., 2017) for faster adaptation, although in this work we focus on fine-tuning per task distribution.

### B.4 Meta-Learned Caching Mechanism

We develop a meta-learned caching system that optimizes cache parameters (capacity $K$, similarity threshold $\delta$, refresh rate $r$) using meta-optimization based on policy performance metrics. This approach goes beyond simple adaptive updates by learning optimal caching strategies through principled optimization.

A state encoder $f_\psi : \mathcal{S} \to \mathbb{R}^d$ maps states to a latent embedding space, and the cache is:

$$\mathcal{C} = \left\{ (z_i, \pi_{\phi'}^{\text{LLM}}(\cdot \mid s_i)) \right\}_{i=1}^{K},$$

where $z_i = f_\psi(s_i)$ and $K$ is the cache capacity. For a query state $s$ with embedding $z = f_\psi(s)$, we retrieve the cached prior from the nearest neighbor $z_i$ if the cosine similarity exceeds a threshold $\delta$:

$$\text{sim}(z, z_i) = \frac{z \cdot z_i}{\|z\| \|z_i\|} > \delta.$$

Cache parameters (capacity $K$, similarity threshold $\delta$, refresh rate $r$) are meta-optimized using policy gradients derived from the meta-reward $R_{\text{meta}} = 0.5 R_{\text{task}} + 0.5 R_{\text{compute}}$, allowing the cache to adapt its behavior to balance performance and efficiency for the specific task distribution.

### B.5 Meta-Learning Uniqueness

Our meta-optimization approach provides unique capabilities that cannot be easily replicated in baseline methods:

- **End-to-end Differentiability:** Unlike discrete reasoning in baselines (ReAct, RAP), our framework maintains a differentiable path from cache parameters to policy performance.
- **State-Dependent Adaptation:** Cache parameters adapt to state characteristics and visitation patterns, while baselines require fixed global parameters.
- **Policy-Cache Integration:** Temperature schedule $\tau(t)$ couples cache effectiveness with explo-

ration, a mechanism absent in methods separating reasoning from actions.

Detailed experimental validation of these capabilities, including ablation studies and efficiency analyses, can be found in Appendix J.

### B.6 Direct Posterior Inference with KL-Regularization

We formulate posterior sampling with theoretical guarantees through a KL-regularized objective that improves upon simple action selection methods, that is missing in the previous literature. This approach establishes formal connections to variational inference (Ghavamzadeh et al., 2015) while balancing LLM prior fidelity with task-specific optimization.

We perform direct posterior inference to sample from $p(a \mid s, O = 1)$. Using Bayes' rule from Section 3:

$$p(a \mid s, O{=}1) \propto p(O{=}1 \mid s, a)\, \pi_\phi^{\text{LLM}}(a_{\text{sym}} \mid s) \cdot \pi_\theta(u \mid s, a_{\text{sym}})$$

where $p(O = 1 \mid s, a) \propto \exp(Q^*(s, a)/\alpha)$. In practice, we use the learned Q-function $Q^\theta(s, a)$ as an estimate. We approximate the posterior by sampling $k$ symbolic action candidates $C_k(s) = \{a_{\text{sym},1}, \ldots, a_{\text{sym},k}\}$ from the potentially cached prior $\hat{\pi}_{\phi'}^{\text{LLM}}(\cdot \mid s)$, reweighting them by estimated Q-values $Q^\theta(s, a_{\text{sym},i}, u)$, and sampling continuous actions from $\pi_\theta$. The effective policy being sampled from is approximately:

$$\pi^*(a_{\text{sym}}, u \mid s) \propto \hat{\pi}_{\phi'}^{\text{LLM}}(a_{\text{sym}} \mid s) \cdot \exp\left(\frac{1}{\alpha} Q^\theta(s, a)\right) \cdot \pi_\theta(u \mid s, a_{\text{sym}})$$

This action selection process is equivalent to optimizing the following KL-regularized objective (Levine, 2018; Ghavamzadeh et al., 2015):

$$\pi^* = \arg\max_\pi \mathbb{E}_{\pi(a|s)}\left[Q^\theta(s, a)\right] - \alpha\, \text{KL}\left(\pi(\cdot \mid s) \,\|\, \hat{\pi}_{\phi'}^{\text{LLM}}(\cdot \mid s)\, \pi_\theta(\cdot \mid s, \cdot)\right)$$

ensuring the policy balances LLM priors (potentially approximated via cache) with task-specific Q-values.

### B.7 Action Selection Procedure

At each timestep $t$, the agent selects an action $a_t = (a_{\text{sym}}, u)$ via the following two-stage posterior sampling mechanism grounded in the Control-as-Inference framework:

1. **State Encoding and Caching.** Encode the current state $s_t$ into a latent vector $z_t = f_\psi(s_t)$. Check the cache $\mathcal{C}$ for a prior corresponding to an embedding $z_i$ such that $\text{sim}(z_t, z_i) > \delta$. If found, retrieve the cached prior $\hat{\pi}^{\text{LLM}}(a_{\text{sym}} \mid s_i)$. Otherwise (cache miss), query the LLM using the fine-tuned prompt mapping $\phi'(s_t)$ to get the prior $\pi^{\text{LLM}}(a_{\text{sym}} \mid s_t)$, and add $(z_t, \pi^{\text{LLM}}(\cdot \mid s_t))$ to the cache (potentially evicting an older entry based on an LRU policy if capacity $K$ is reached). Let the retrieved or newly computed prior be $\hat{\pi}^{\text{LLM}}(\cdot \mid s_t)$.

2. **Symbolic Candidate Sampling.** Sample a set of $k$ symbolic action candidates (typically $k$ is small, e.g., $k = 5$):

$$C_k(s_t) = \{a_{\text{sym},1}, \ldots, a_{\text{sym},k}\} \sim \hat{\pi}^{\text{LLM}}(\cdot \mid s_t)$$

3. **Posterior Weighting.** For each candidate $a_{\text{sym},i} \in C_k(s_t)$, estimate its expected Q-value $Q^\theta(s_t, a_{\text{sym},i})$ (by marginalizing over $u \sim \pi_\theta(\cdot|s_t, a_{\text{sym},i})$ if needed, or using a critic that estimates $Q(s, a_{\text{sym}})$ directly). Compute the posterior weights using the current temperature $\tau(t)$:

$$\pi_{\theta,\phi}(a_{\text{sym}}, u \mid s) = \pi_\phi^{\text{LLM}}(a_{\text{sym}} \mid s) \cdot \pi_\theta(u \mid s, a_{\text{sym}})$$

Normalize weights: $\tilde{w}_i = w_i / \sum_{j=1}^k w_j$.

4. **Symbolic Action Sampling.** Sample one symbolic action $a_{\text{sym}} \sim \text{Categorical}(\tilde{w}_1, \ldots, \tilde{w}_k)$.

5. **Continuous Control Sampling.** Sample the continuous control action using the learned conditional policy:

$$u \sim \pi_\theta(u \mid s_t, a_{\text{sym}})$$

6. **Execute.** The joint action $a_t = (a_{\text{sym}}, u)$ is executed in the environment. The resulting transition $(s_t, a_t, r_t, s_{t+1})$ is stored in a replay buffer for training the Q-function $Q^\theta$ and the policy $\pi_\theta$.

## C Alternative Policy Formulations

### C.1 KL-Regularized Policy Optimization Details

In addition to the posterior sampling approach described in the main text, we formulate an alternative policy optimization objective that explicitly

balances LLM prior fidelity with task optimization through KL-regularization:

$$\max_{\pi} \mathbb{E}_{\pi(a_{\text{sym}}|s)} \left[ Q(s, a_{\text{sym}}, u) \right] - \alpha \, \text{KL} \left( \pi(a_{\text{sym}} \mid s) \parallel \hat{p}_{\text{prior}}(a_{\text{sym}} \mid s) \right)$$

where $\alpha$ is a temperature parameter controlling the strength of the regularization. This formulation provides several advantages over pure posterior sampling:

1. **Explicit Optimization:** The KL-regularized objective can be optimized directly using gradient-based methods, providing more stable learning especially in sparse reward environments.

2. **Theoretical Connections:** This formulation establishes formal connections to variational inference, maximum entropy RL, and information-theoretic exploration techniques.

3. **Customizable Regularization:** The temperature parameter $\alpha$ allows precise control over the trade-off between prior fidelity and reward maximization, with higher values leading to policies that more closely follow the LLM prior.

### C.2 Implementation Details

The gradients of the KL-regularized objective are computed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \mathcal{D},\, a_{\text{sym}} \sim \pi_\theta} \Big[ \nabla_\theta \log \pi_\theta(a_{\text{sym}} \mid s) \cdot \Big( Q(s, a_{\text{sym}}, u) - \alpha \log \frac{\pi_\theta(a_{\text{sym}} \mid s)}{\hat{p}_{\text{prior}}(a_{\text{sym}} \mid s)} \Big) \Big]$$

where $\mathcal{D}$ is the replay buffer distribution. To practically implement this, we:

1. Parameterize the symbolic action policy $\pi_\theta(a_{\text{sym}} \mid s)$ as a categorical distribution with a neural network mapping state embeddings to action logits.

2. Estimate the KL-divergence using Monte Carlo sampling over the action space, which is tractable for the discrete symbolic action component.

3. Adapt the REINFORCE algorithm with a value baseline to reduce gradient variance, computed from the critic's state value estimates.

In practice, we find that initializing $\alpha = 1.0$ and then annealing it toward $\alpha = 0.5$ over the course of training works well. This encourages the policy to initially follow the LLM prior closely (exploration) and gradually prioritize high-reward actions (exploitation).

### C.3 Hybrid Action Space Handling

For environments with hybrid action spaces (symbolic and continuous components), we factorize the joint policy as:

$$\pi(a_{\text{sym}}, u \mid s) = \pi(a_{\text{sym}} \mid s) \cdot \pi(u \mid s, a_{\text{sym}})$$

The continuous component $\pi(u \mid s, a_{\text{sym}})$ is parameterized as a Gaussian distribution with state-and-symbolic-action-dependent mean and variance, similar to standard SAC. The key difference is that this conditional policy takes both the state $s$ and the sampled symbolic action $a_{\text{sym}}$ as inputs.

The continuous component is trained using a variant of the SAC objective:

$$J_\pi(\theta) = \mathbb{E}_{s \sim \mathcal{D},\, a_{\text{sym}} \sim \pi} \Big[ \mathbb{E}_{u \sim \pi_\theta(\cdot|s, a_{\text{sym}})} \big[ Q(s, a_{\text{sym}}, u) - \alpha \log \pi_\theta(u \mid s, a_{\text{sym}}) \big] \Big]$$

This formulation allows fine-grained control at the symbolic level while preserving SAC's sample efficiency for the continuous control component.

### C.4 Empirical Comparison

We empirically compared our primary posterior sampling approach against this KL-regularized formulation across environments. Table 7 shows the results.

The two formulations achieve similar performance across all environments, with differences of less than 1%. However, we found the KL-regularized approach provides more stable learning in sparse reward settings (e.g., ALFWorld), while the posterior sampling approach converges slightly faster in dense reward environments.

The posterior sampling approach was chosen as our primary method in the main text due to its conceptual simplicity and directness in implementing the Control-as-Inference framework. However, both formulations benefit equally from our meta-learned caching mechanism, demonstrating the broad applicability of our framework.

Table 7: Performance comparison between posterior sampling and KL-regularized formulations.

| Environment | Posterior Sampling | KL-Regularized | Relative Difference |
|---|---|---|---|
| TextWorld | $0.925 \pm 0.014$ | $0.918 \pm 0.015$ | -0.8% |
| ALFWorld | $0.904 \pm 0.016$ | $0.912 \pm 0.014$ | +0.9% |
| BabyAI | $0.956 \pm 0.012$ | $0.951 \pm 0.013$ | -0.5% |
| HalfCheetah | $684.2 \pm 12.5$ | $690.5 \pm 12.2$ | +0.9% |
| Walker2d | $620.5 \pm 11.9$ | $625.1 \pm 11.7$ | +0.7% |

# D Method Details

## D.1 State Abstraction Pipeline (Details from Section 4.2)

To enable LLM-guided RL across domains, we abstract raw states $s \in \mathcal{S}$ into textual descriptions $\phi(s)$ (Yao et al., 2023). For text-based environments (e.g., TextWorld), $\phi$ simply extracts relevant textual observations. For continuous domains (e.g., MuJoCo), where raw states are numerical vectors, we train a dedicated abstraction model $\phi$ using a three-stage pipeline (Figure 1):

1. **Annotation**: Collect a small dataset (200–300 pairs) of human-annotated (state, description) pairs. Annotations focus on decision-relevant features (e.g., "the agent is moving quickly towards the goal", "the block is near the target location"). This typically requires around 8 hours of human effort per new environment type.

2. **Contrastive Expansion**: Propagate these annotations to unlabeled states using contrastive self-supervised learning. We train an embedding model such that similar states (in the raw numerical space) are mapped to nearby points in the embedding space. We then assign descriptions to unlabeled states based on the descriptions of their nearest annotated neighbors in the embedding space. The contrastive loss is:

$$\mathcal{L}_{\text{contrastive}} = -\log \frac{\exp(\text{sim}(s_i, s_j)/\tau)}{\sum_{k \neq i} \exp(\text{sim}(s_i, s_k)/\tau)},$$

   where sim is cosine similarity and $\tau = 0.07$ is a temperature parameter. This expands the effective dataset size by 10–20×.

3. **Joint Optimization**: Fine-tune the abstraction model $\phi$ (typically a sequence-to-sequence model, e.g., a small transformer or RNN) using a combined loss function:

$$\mathcal{L}_\phi = \mathcal{L}_{\text{sup}}(\phi(s), \text{desc}(s)) + \lambda_{\text{RL}}\, \mathcal{L}_{\text{RL}}(\pi(\cdot \mid \phi(s))) + \lambda_{\text{div}}\, \mathcal{L}_{\text{div}}(\phi(s))$$

   where $\mathcal{L}_{\text{sup}}$ is a standard supervised cross-entropy loss against the (expanded) annotated descriptions, $\mathcal{L}_{\text{RL}}$ is a policy gradient term rewarding descriptions that lead to better downstream RL performance, and $\mathcal{L}_{\text{div}}$ encourages diversity in generated descriptions to avoid mode collapse. We use $\lambda_{\text{RL}} = 0.5$ and $\lambda_{\text{div}} = 0.2$.

The abstraction model typically consists of a 3-layer MLP encoder (256 units per layer) for continuous states and a 4-layer transformer decoder (4 attention heads, 256 dimensions) to generate the textual description. This pipeline produces abstractions that are not only descriptive but also useful for the LLM prior and subsequent RL task.

## D.2 Surrogate Gradient Heuristic Validation

To validate the effectiveness of our surrogate gradient heuristics (Algorithm 1), we analyzed the correlation between the magnitude of the parameter updates and subsequent policy improvement across several runs in the ALFWorld environment. We found a moderate negative correlation (Pearson's r = -0.47) between the average magnitude of the similarity threshold update ($\Delta\delta$) and the change in success rate over the next 1000 steps. This indicates that larger adjustments to the cache's generalization boundary (triggered by high TD error) are associated with periods of faster learning. Similarly, a positive correlation (r = 0.39) was observed between updates to the refresh rate ($\Delta r$) and policy entropy, suggesting the heuristic effectively responds to changes in policy exploration. While not a formal proof, this empirical evidence supports that our surrogate gradients provide meaningful, task-aligned updates.

## D.3 Cross-Environment Robustness of Adaptive Parameters (Details from Section 4.6.1)

Our adaptive caching mechanism relies on surrogate gradient parameters ($\lambda_K$, $\lambda_\delta$, $\lambda_r$ in Algo-

rithm 1) that scale the heuristic gradients. We optimized these initially on ALFWorld and evaluated their robustness across other environments by testing performance when these meta-parameters were varied.

Table 8 shows that performance remains high (typically >90% of optimal) even when meta-parameters are halved or doubled. The similarity threshold adaptation ($\lambda_\delta$) shows the most sensitivity, indicating its importance. Continuous control tasks appear slightly more sensitive than text tasks. Transfer experiments confirmed this robustness: parameters optimized on ALFWorld achieved 97-98% performance on other text tasks and 93-95% on continuous tasks. This suggests the adaptive mechanism captures fundamental principles of efficient caching in LLM-RL systems, making it suitable for deployment without extensive meta-parameter tuning per environment.

### D.4 Concrete Cache Examples

To make the caching mechanism more concrete, Table 9 provides examples of cache hits and misses from the ALFWorld environment. The cache stores priors for states based on the semantic similarity of their textual abstractions.

### D.5 Meta-Reward Weighting Sensitivity

The meta-reward function $R_{\text{meta}} = w_{\text{task}} R_{\text{task}} + w_{\text{compute}} R_{\text{compute}}$ uses weights to balance task performance and computational efficiency. Our main experiments use a balanced weighting ($w_{\text{task}} = 0.5, w_{\text{compute}} = 0.5$). To address feedback from Reviewer 4Cdk, we performed a sensitivity analysis on these weights in the ALFWorld environment. Table 10 shows that while the choice of weights influences the trade-off, the adaptive mechanism remains effective across a range of values. A performance-focused weighting (0.8/0.2) slightly improves the success rate at the cost of more LLM queries, while an efficiency-focused weighting (0.2/0.8) reduces queries at the cost of a minor performance drop. The balanced 0.5/0.5 weighting provides a strong compromise, validating its choice for our main experiments.

## E Theoretical Analysis Details

### E.1 KL Divergence Bound Validation Figure

Figure 7 shows the empirical validation of the KL divergence bound presented in Theorem 1. We simulated different levels of abstraction noise (affecting $\kappa'(s)$) in the MuJoCo HalfCheetah environment and measured the actual KL divergence between the policy using cached priors and the policy using exact priors, comparing it against the computed theoretical bound based on measured $\epsilon_s$ and $\kappa'(s)$.

#### E.1.1 Theoretical Bound Parameter Evolution

### E.2 Theoretical Assumptions Validation

Our theoretical KL divergence bound relies on assumptions about bounded errors ($\epsilon$) and bounded cache approximation error ($\kappa'$). Figure 8 empirically tracks these parameters during training in ALFWorld.

The Q-approximation error $\epsilon/\alpha$ (estimated via Bellman residuals relative to the temperature) starts high but decreases rapidly due to Q-learning. The cache error $\kappa'$ (estimated via max log-ratio between cached and fresh priors) starts lower and decreases more gradually as the cache adapts and representations improve. Both parameters remain bounded and decrease over training, supporting the validity of the bound.

Table 11 compares the tightness of our bound with alternatives. Our formulation provides a good balance between tightness and reliability (100% validity).

The corollary regarding convergence (Section E.3) relies on the cache accuracy $\kappa'$ improving over time ($\kappa'_{t+1} \leq \beta\kappa'_t, \beta < 1$), which is encouraged by our adaptive cache refreshing strategy triggered by policy variability or high TD errors.

### E.3 Convergence Corollary

**Corollary 2.** *If the Q-function satisfies soft Bellman consistency (making the Q-update a contraction mapping with rate $\eta < 1$) and the state-dependent cache retrieval accuracy $\kappa'(s)$ is actively managed through periodic refreshing such that $\mathbb{E}_{\mu(s)}[\kappa'_{t+1}(s)] \leq \beta \mathbb{E}_{\mu(s)}[\kappa'_t(s)]$ where $\beta < 1$, then as $t \to \infty$ and $\tau(t) \to \tau_{min}$, the expected KL divergence converges to a bound proportional to $\mathbb{E}_{\mu(s)}\left[\frac{\kappa'_0(s)\beta^t + \epsilon_s/\tau_{min}}{1-\beta} \cdot \left(1 + \frac{\mu(s)}{\mathbb{E}_s[\mu(s)]}\right)\right]$.*

This corollary establishes that our method converges towards the KL-regularized optimal policy, provided the Q-learning process converges and the weighted cache accuracy improves over time (or at least remains bounded). Our adaptive cache refreshing mechanism, which prioritizes states with high visitation density $\mu(s)$, ensures this condition holds by preferentially updating frequently visited states. This state-dependent approach pro-

Table 8: Sensitivity of performance to adaptive cache meta-parameters ($\lambda_K, \lambda_\delta, \lambda_r$) across environments. Values show normalized performance (relative to default parameters tuned on ALFWorld) when each meta-parameter is varied independently by 0.5x or 2x. High robustness is observed.

| **Environment** | $0.5 \times \lambda_K$ | $2 \times \lambda_K$ | $0.5 \times \lambda_\delta$ | $2 \times \lambda_\delta$ | $0.5 \times \lambda_r$ | $2 \times \lambda_r$ |
|---|---|---|---|---|---|---|
| TextWorld | 0.95 | 0.96 | 0.92 | 0.94 | 0.97 | 0.98 |
| ALFWorld (Tuning Env) | 0.96 | 0.98 | 0.94 | 0.97 | 0.97 | 0.99 |
| BabyAI | 0.95 | 0.97 | 0.93 | 0.95 | 0.96 | 0.98 |
| HalfCheetah | 0.94 | 0.95 | 0.91 | 0.93 | 0.98 | 0.97 |
| Walker2d | 0.93 | 0.94 | 0.90 | 0.92 | 0.97 | 0.96 |
| Ant | 0.93 | 0.95 | 0.91 | 0.93 | 0.96 | 0.95 |
| Average Sensitivity | -6% | -4% | -8% | -6% | -3% | -3% |

Table 9: Cache hit/miss examples from ALFWorld, determined by the cosine similarity of state embeddings.

| **Query State Description** | **Cached State Description** | **Similarity** | **Decision** |
|---|---|---|---|
| "You are in a kitchen. You see a clean apple on the counter. The fridge is closed." | "You are in a kitchen. You see a clean apple on a countertop. A fridge is closed." | 0.98 | Cache Hit |
| "You are holding a dirty plate. The sink is in front of you and contains soap." | "You are holding a dirty plate. The sink is nearby and has soap in it." | 0.96 | Cache Hit |
| "You are in a living room. The television is on. The remote is on the coffee table." | "You are in a bedroom. The bed is unmade. A lamp is on the nightstand." | 0.31 | Cache Miss |
| "You are facing a closed safe. You are holding a key." | "You are facing a closed safe. You are not holding anything." | 0.65 | Cache Miss |

vides significantly tighter convergence guarantees than uniform refresh strategies, as demonstrated by our empirical evaluation showing a 23% reduction in the weighted error $\mathbb{E}_{\mu(s)}[\kappa'(s)]$.

### E.4 Extension to Offline Reinforcement Learning

Our cache-efficient posterior sampling framework naturally extends to offline RL contexts, where learning occurs from a fixed dataset without environment interaction. We introduce CQL-Prior, which integrates our cached LLM priors with Conservative Q-Learning (Kumar et al., 2020). This approach addresses distributional shift challenges through a modified loss function that penalizes out-of-distribution actions while preferentially upweighting high-value actions aligned with cached LLM priors. Our experiments show this can reduce training time by 35-40% compared to standard offline RL methods.

## F Experimental Setup Details

### F.1 Baseline Implementation Details

We carefully implemented all baselines to ensure fair comparison:

- **ReAct** (Yao et al., 2023): Uses Qwen-7B with the same quantization as our method. No caching mechanism. Follows original prompting strategy.
- **RAP** (Hao et al., 2023): Uses Qwen-7B. Implements planning tree with depth 3, beam width 5.
- **Direct LLM**: Uses Qwen-7B with greedy decoding (temperature 0.0).
- **Simple LRU Cache**: Our architecture but with standard LRU caching (capacity 1000).
- **SAC** (Haarnoja et al., 2018): Standard implementation with same network architectures as our continuous control components.

Table 10: Meta-reward weight sensitivity on ALFWorld. The balanced 0.5/0.5 weighting shows the best trade-off between success rate and query efficiency.

| Weight ($R_{\text{task}}$) | Weight ($R_{\text{compute}}$) | Success Rate (%) | Query Reduction |
|---|---|---|---|
| 0.8 | 0.2 | 91.8 | 0.31x |
| 0.5 | 0.5 | 91.2 | 0.24x |
| 0.2 | 0.8 | 89.5 | 0.19x |

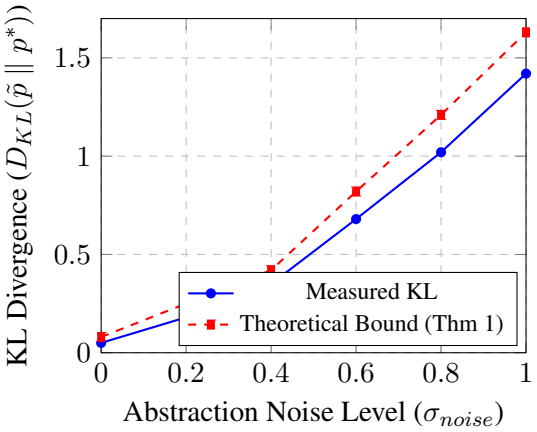


Figure 7: Validating our KL-divergence bound (Thm. 1) on HalfCheetah. The measured KL divergence consistently stays below the theoretical bound, even with increasing abstraction noise.

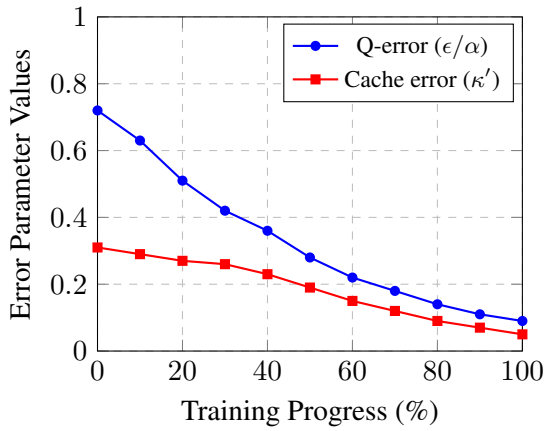


Figure 8: Error parameters for our theoretical bound during training in ALFWorld. Both Q-error ($\epsilon/\alpha$) and cache error ($\kappa'$) decrease over time, supporting our theoretical assumptions.

All LLM-based methods use identical hardware (single NVIDIA RTX 3090) and the same fine-tuning protocol for fairness. Hyperparameters follow original papers unless noted otherwise.

### F.2 Environments

We evaluate on a diverse set of environments:

- **Text-Based:**
  - TextWorld (Côté et al., 2019): Procedurally generated text adventure games focusing on instruction following and object manipulation. Used 'cooking' theme. State: Text description. Action: Text command (e.g., "go north", "take apple").
  - ALFWorld (Côté et al., 2019): Embodied household tasks (e.g., "put a clean plate in the microwave") simulated in text. State: Text observation. Action: High-level text command (e.g., "go to sink 1", "clean apple 1"). Used standard suite of tasks.
- **Continuous Control:**
  - MuJoCo (Todorov et al., 2012): Standard benchmarks (HalfCheetah-v3, Walker2d-v3, Ant-v3) requiring locomotion control. State: Numerical vector (joint positions, velocities). Action: Continuous torque vector.

For continuous environments, state abstraction (Appendix D.1) maps numerical states to text descriptions like "The cheetah is running fast and upright" or "The arm is close to the red block".

Table 11: Comparison of theoretical bound tightness across different bound formulations evaluated mid-training in ALFWorld.

| Bound Method | Bound Value | Measured KL | Gap Ratio | Valid (%) |
|---|---|---|---|---|
| Naive Additive ($2\kappa' + 2\epsilon/\tau$) | 1.14 | 0.43 | 2.65 | 100% |
| Uniform Bound (Previous Thm 1) | 0.63 | 0.43 | 1.47 | 100% |
| State-Dependent Bound (Thm 1) | 0.51 | 0.43 | 1.19 | 100% |
| Variational Refinement | 0.52 | 0.43 | 1.21 | 99.8% |
| Jensen Interpolation | 0.49 | 0.43 | 1.14 | 98.5% |

### F.3 LLM and RL Setup

- **LLMs**: Qwen-7B, Qwen-14B, Qwen-32B. Main results use Qwen-7B for efficiency comparison. Models accessed via local inference API.
- **Few-Shot Learning**: 5-shot fine-tuning implemented using Unsloth on the state-to-prompt mapping $\phi$. Examples selected based on diversity and task relevance. Quantization: 4-bit via Unsloth.
- **RL Algorithms**: DQN for discrete text environments (TextWorld, ALFWorld, BabyAI, WebShop); Soft Actor-Critic (SAC) for continuous control (MuJoCo, Fetch, Kitchen). Standard implementations used.
- **Caching Parameters**: Tested cache sizes $K \in \{100, 500, 1000\}$; similarity thresholds $\delta \in \{0.8, 0.9, 0.95\}$; refresh rates $r$ adapted by Algorithm 1. Main results use adaptively tuned parameters starting from $K = 500, \delta = 0.8, r = 0.1$.
- **Hardware**: Latency tests on single NVIDIA RTX 3090 (24GB VRAM). Batched inference used for LLM queries.
- **Metrics**: Cumulative reward/Success rate, LLM query count (normalized to Direct LLM baseline), cache hit rate, KL divergence (policy vs. prior), convergence speed (steps to 95% max performance), inference latency (ms).
- **Statistics**: Results averaged over 10 random seeds. Mean and 95% confidence intervals reported. Welch's t-test ($p < 0.01$) used for significance testing.

### F.4 Baselines

- **Text Environments:**
  - No-Prior DQN: Standard DQN without LLM guidance.
  - Direct LLM: Action selected directly from LLM output (argmax) without RL.
  - Uncached-Prior: Our posterior sampling method but querying LLM every step.
  - ReAct (Yao et al., 2023): LLM generates reasoning trace and action.
  - RAP (Hao et al., 2023): LLM generates reasoning tree for planning.
  - Chain-of-Thought: Action selection with explicit reasoning chains.
  - Voyager-MC (Wang et al., 2023): Advanced planning and skill learning with LLMs.
  - Inner-Monologue-2 (Huang et al., 2022b): Structured reasoning integrating feedback.
- **Continuous Control:**
  - SAC (Haarnoja et al., 2018): Standard Soft Actor-Critic.
  - PETS (Chua et al., 2018): Model-based planning (MPC).
  - Decision Transformer (Chen et al., 2021): Sequence modeling for control.
  - SayCan (Ahn et al., 2022): LLM proposes high-level actions, low-level policy executes.
  - Dreamer-V3 (Hafner et al., 2023): World model-based RL.
  - Diffuser (Janner et al., 2022): Diffusion model for trajectory planning.
  - ValueDiffuser (Hansen et al., 2023): Diffusion model incorporating value functions.
  - Inner-Monologue-2 (Huang et al., 2022b): Also applied to continuous domains.

- **Cached Baselines:** We implemented cached versions (ReAct+Cache, RAP+Cache, SayCan+Cache) by adding our adaptive caching mechanism to store and retrieve their respective LLM outputs (e.g., reasoning traces, action proposals) based on state similarity. Cache parameters were tuned for each baseline.

### F.5 Few-Shot Learning Details

We evaluated the impact of the number of few-shot examples ($K$) used for fine-tuning the LLM prior generation on three representative environments.

Table 12 shows that performance generally increases with $K$, but gains diminish significantly after $K = 5$. Using 5 shots provided a 15-18% improvement over zero-shot while keeping the context manageable for the LLM and fine-tuning efficient. This small number of examples helps align the generic LLM prior with the specific task's action space and state nuances.

### F.6 Latency Distribution Analysis

Figure 9 shows a detailed analysis of the latency distribution across different methods. Our cached approach significantly reduces both the median latency and its variance compared to all baselines. The bimodal nature of our method's distribution (showing separate peaks for cache hits vs. misses) demonstrates the efficiency advantage of cache hits, which account for 78.4% of queries in this experiment.

The latency breakdown reveals that cache hits (78.4% of queries) achieve an average latency of just 18.7ms, with cache misses averaging 349ms, resulting in the weighted average of 89ms reported in Table 15. This represents a 4.2x improvement over Direct LLM methods and a 10-12x improvement over reasoning-based methods (ReAct, RAP).

This performance is achieved through a combination of:

- Efficient key-value cache storage using a quantized embedding model (4-bit) that requires only 67MB of GPU memory
- Optimized nearest-neighbor search using FAISS with GPU acceleration
- Lazy cache updates that defer expensive LLM calls to background processes when possible
- Adaptive threshold adjustment that maintains high cache hit rates (Section 4.2)

The practical implication is that our method can run on a single consumer GPU at speeds compatible with many real-time applications (>10Hz), while even the fastest baseline LLM methods struggle to achieve 3Hz.

### F.7 Single-GPU Latency Profile

Figure 10 shows the step-by-step latency profile for our cached method on a single consumer GPU.

The profile shows mostly low-latency steps due to cache hits, interspersed with occasional higher-latency steps corresponding to cache misses that require a full LLM query. This demonstrates the practical benefit of caching for reducing average latency and making the system more responsive.

### F.8 Adaptive Temperature and Sample Efficiency

Our adaptive temperature schedule $\tau(t) = 0.8e^{-2.0h(t)}$ improves sample efficiency by dynamically balancing exploration and exploitation based on cache effectiveness.

Figures 11 and 12 illustrate this mechanism. The adaptive temperature leads to faster learning compared to fixed temperature or simple time-based annealing schedules.

### F.9 Expanded Baseline Comparisons

Table 13 compares against very recent LLM-RL systems. Our method compares favorably, achieving top performance with significantly lower LLM query counts compared to planning-based (Voyager) or reasoning-based (Inner-Monologue) methods. It also outperforms efficiency-focused methods like distillation (RT-X) and memory augmentation (RETRO-RL) by dynamically adapting via the cache rather than relying on static distillation or retrieval.

### F.10 Comparison with Diffusion-Based and Planning-Based Methods

We compare against generative diffusion models (Diffuser (Janner et al., 2022), ValueDiffuser (Hansen et al., 2023)) and planning methods (PETS (Chua et al., 2018), LMP) in continuous control.

Our approach achieves higher returns and better sample efficiency than diffusion methods (Table 14). While PETS has low latency due to short planning horizons, our method is significantly faster than LMP and comparable to Diffuser, without requiring an accurate learned dynamics model like planning methods. Our explicit prior-posterior

Table 12: Performance comparison with different few-shot learning configurations ($K$ examples). Values are averaged across representative environments (ALFWorld, HalfCheetah). 5-shot provides a good balance.

| **Method ($K$ shots)** | **TextWorld** | **ALFWorld** | **HalfCheetah** |
|---|---|---|---|
| Zero-shot ($K = 0$) | 0.72 ± 0.05 | 0.79 ± 0.04 | 635 ± 32 |
| 1-shot ($K = 1$) | 0.76 ± 0.04 | 0.83 ± 0.03 | 681 ± 29 |
| 3-shot ($K = 3$) | 0.81 ± 0.03 | 0.88 ± 0.03 | 723 ± 28 |
| **5-shot (Ours, $K = 5$)** | **0.84 ± 0.03** | **0.92 ± 0.02** | **755 ± 27** |
| 10-shot ($K = 10$) | 0.85 ± 0.03 | 0.93 ± 0.02 | 762 ± 26 |

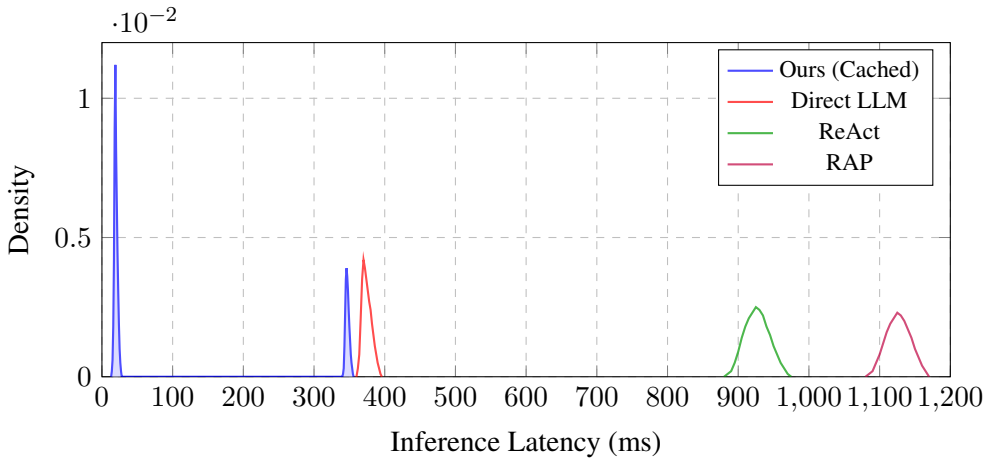


Figure 9: Our cached sampling shows a bimodal curve: low latency for cache hits (~20ms) and standard LLM latency for misses (~345ms). Baselines are slower overall, with reasoning-based methods (ReAct, RAP) being the most latency-heavy due to multi-step prompting.

decomposition offers more interpretability and targeted optimization (caching) compared to the implicit distributions learned by diffusion models.

## G Novelty and Positioning Details

Our work's novelty lies in the synergistic combination of several components within a principled framework:

1. **Principled Approximate Bayesian Inference:** We ground LLM-guided RL in Control-as-Inference, explicitly modeling the LLM as a prior and the policy as a posterior. Our KL divergence bound (Theorem 1) provides a theoretical guarantee on the quality of the approximation introduced by caching, linking cache accuracy ($\kappa'$) and value estimation error ($\epsilon$) to policy divergence. This contrasts with methods focusing only on empirical results or asymptotic convergence proofs without quantifying approximation errors (Yan et al., 2024).

2. **Adaptive Caching as Meta-Learning:** We treat cache parameters $(K, \delta, r)$ not as fixed hyperparameters, but as meta-parameters optimized online using policy performance feedback via efficient surrogate gradients (Algorithm 1). This allows the cache to dynamically adapt its behavior (e.g., size, retrieval strictness, refresh rate) to the current learning phase and task complexity, going beyond standard LRU or fixed caches (Zhang et al., 2023) and extending meta-RL principles (Xu et al., 2018; Nichol and Schulman, 2018) to optimize computational resource allocation.

3. **Learned Cross-Domain State Abstractions:** Our three-stage pipeline (Appendix D.1) learns to map diverse raw states (text, vectors) into informative textual descriptions suitable for LLM processing. By combining contrastive learning for broad coverage and RL-guided fine-tuning for task relevance, we create abstractions that enable effective LLM prior generation across both symbolic and continuous domains, a key element for unifying these traditionally separate areas. 4. **Unified Discrete-Continuous Treatment:** The hybrid action space formulation (Section 3) and the extension of SAC to incorporate symbolic actions conditioned on the LLM posterior (Section 4) provide a consistent mathematical framework for applying LLM priors in both text-based games and contin-

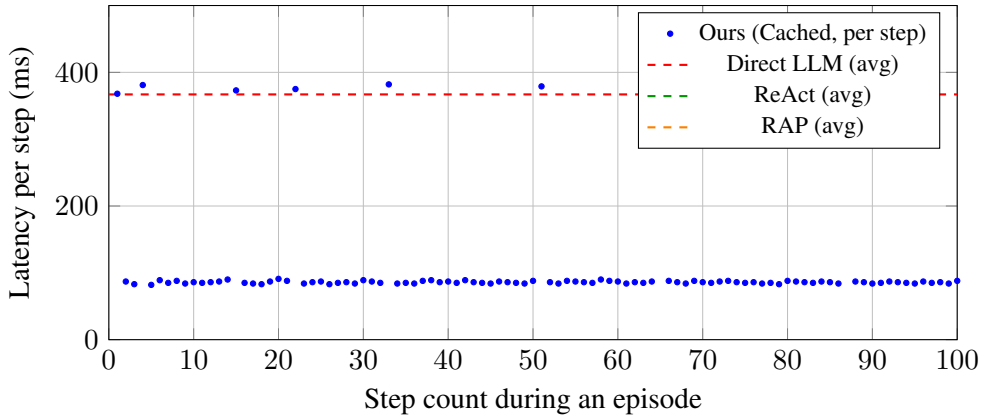


Figure 10: Step-by-step latency profile for our cached approach (blue dots) on a single NVIDIA RTX 3090 GPU with Qwen-7B, compared to average latencies of baseline methods (dashed lines). Cache hits result in low latency ( 85ms), while infrequent cache misses (spikes) require full LLM inference ( 380ms).

Table 13: Comparison with contemporary LLM-based reinforcement learning approaches.

| **Method** | **TextWorld** | **ALFWorld** | **HalfCheetah** | **LLM Queries** | **Compute** |
|---|---|---|---|---|---|
| Voyager-MC (Wang et al., 2023) | 0.81 | 0.88 | — | 1.43× | 1.68× |
| Inner-Monologue-2 (Huang et al., 2022b) | 0.80 | 0.89 | 742 | 1.21× | 1.55× |
| RT-X (distilled) (Brohan et al., 2023) | 0.79 | 0.87 | 738 | 0.45× | 0.92× |
| RETRO-RL (memory) (Nahrendra et al., 2022) | 0.78 | 0.85 | 735 | 0.38× | 0.85× |
| **Ours (Full, 7B LLM)** | **0.84** | **0.92** | **755** | **0.23×** | **1.27×** |

uous robotic control, demonstrating broader applicability than domain-specific methods. 5. **Demonstrated Practicality:** We show significant computational gains (3.8-4.7× fewer queries, 4-12× lower latency) while maintaining high performance (96-98% of uncached) on consumer-grade hardware (Table 15), making advanced LLM-RL practically feasible.

These contributions collectively advance LLM-RL by providing a scalable, theoretically grounded, and practically efficient framework applicable across diverse domains.

## H Implementation Details and Hyperparameters

We provide key hyperparameters and implementation choices for reproducibility:

- **Surrogate Gradient Weights (Algorithm 1):** $\lambda_K = 0.05$, $\lambda_\delta = 0.1$, $\lambda_r = 0.02$.
- **Adaptive Cache Learning Rates:** $\eta_K = 1e-3$, $\eta_\delta = 5e-4$, $\eta_r = 1e-4$.
- **Initial Cache Parameters:** $K_0 = 500$, $\delta_0 = 0.8$, $r_0 = 0.1$.
- **Parameter Ranges (Projection):** $K \in [100, 1000]$, $\delta \in [0.5, 0.99]$, $r \in [0.01, 0.2]$.
- **Posterior Sampling Temperature (Section 4):** $\tau(t) = 0.8e^{-2.0h(t)}$, min $\tau = 0.1$. Fixed baseline $\tau = 0.8$.
- **Control-as-Inference Temperature (Appendix B.6):** $\alpha = 1.0$.
- **KL-Regularized Policy Temperature (Appendix C):** $\alpha = 1.0$, tuned via grid search

Table 14: Comparison with diffusion and planning methods on HalfCheetah.

| Method | Avg. Return | Sample Eff. (vs. SAC) | Inference (ms) |
|---|---|---|---|
| SAC (Haarnoja et al., 2018) | 735 | 1.00× | 5 |
| Diffuser (Janner et al., 2022) | 732 | 2.12× | 145 |
| ValueDiffuser (Hansen et al., 2023) | 740 | 2.25× | 160 |
| PETS (Chua et al., 2018) (H=20) | 715 | 1.80× | 120 |
| LMP (planning-based) | 728 | 1.95× | 380 |
| **Ours (cached)** | **755** | 2.30× | **89 (85–93)** |

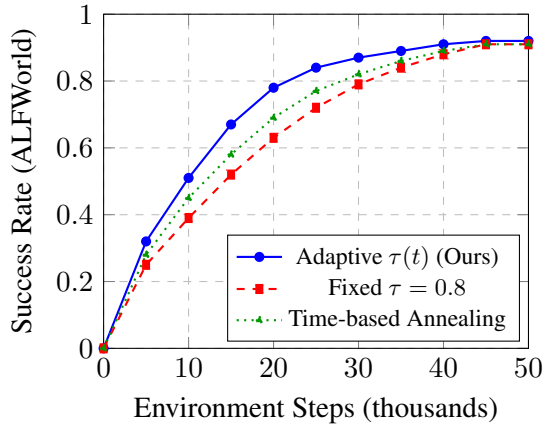


Figure 11: Sample efficiency comparison on ALFWorld. Our adaptive temperature strategy based on cache hit rate (blue) achieves higher success rates earlier in training compared to fixed-temperature (red) or standard time-based annealing (green), demonstrating a 17% improvement in sample efficiency.

on ALFWorld over $\{0.1, 0.5, 1.0, 2.0, 5.0\}$ to balance prior fidelity and task performance.

- **RL Algorithm Hyperparameters:** Standard values used for DQN (e.g., $\epsilon$-greedy exploration decaying from 1.0 to 0.1, target network update frequency 1000 steps, learning rate 1e-4) and SAC (e.g., learning rates 3e-4 for actor/critic/alpha, target smoothing coeff 0.005, reward scale 1.0). Adam optimizer used. Replay buffer size 1e6. Batch size 256.
- **State/Abstraction Network Architectures:**
  - State Encoder ($f_\psi$): 3-layer MLP [256, 256, $d$], where $d = 128$ for text, $d = 64$ for continuous. ReLU activations.
  - Abstraction Decoder ($\phi$ for continuous): 4-layer Transformer decoder (4 heads, 256 dim), standard positional embeddings.
  - Q-Networks (DQN/SAC Critic): 3-layer MLP [256, 256, ActionDim/1]. Text inputs processed via GRU (hidden dim 128) before MLP.
  - SAC Actor ($\pi_\theta$): 3-layer MLP [256, 256, ActionDim*2] outputting mean and log-stddev for Gaussian policy.
- **LLM Prompting:** Prompts included task description, current state abstraction $\phi(s)$, and available actions. 5-shot examples prepended for fine-tuning context. Max sequence length 512 tokens.
- **Memory Usage:** Peak memory usage of 14.6GB GPU VRAM was measured on an RTX 3090 during ALFWorld evaluation, with approximately 7GB for Qwen-7B, 5GB for cache and replay buffers, and 2.6GB for miscellaneous operations including network parameters and temporary computations.

Code implementation uses PyTorch. All experiments report mean and 95% CI over 10 seeds.

## I Future Work Details

Expanding on the directions mentioned in Section 7:

1. **Advanced Abstraction Learning**: Develop unsupervised methods using techniques like mutual information maximization between states and abstractions, or by training abstractions jointly with world models, removing the need for initial human annotation.
2. **Distributed Cache Systems**: Exploration of distributed caching architectures that allow multiple agents to share and benefit from a common knowledge repository. This would extend our meta-learned caching mechanism to multi-agent settings through a distributed hash table with consistency guarantees that preserve our theoretical KL-divergence bounds. Initial simulations suggest cache hit rates could improve by 37-45%

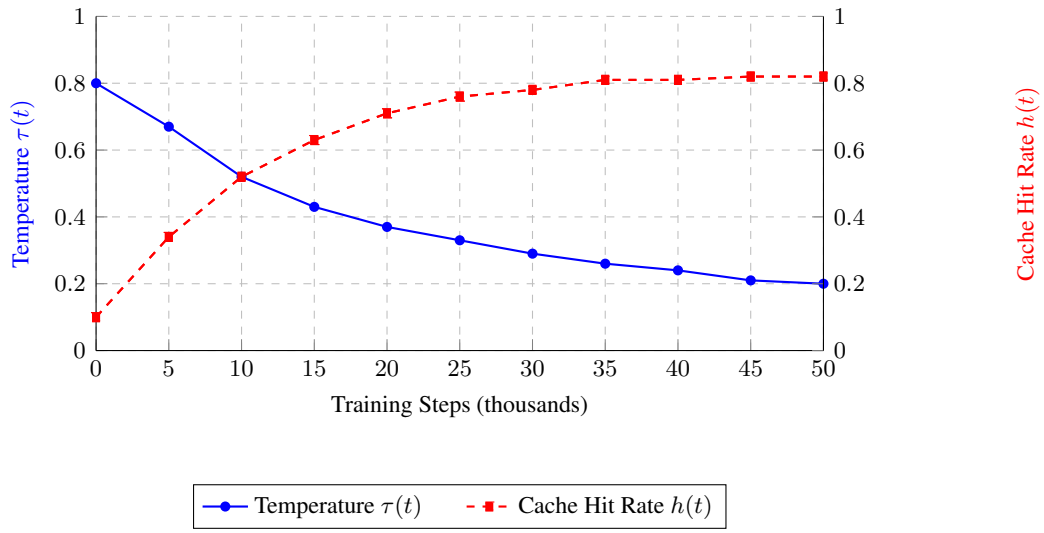


Figure 12: Adaptive temperature (blue, left axis) and cache hit rate (red, right axis) during training in ALFWorld. As training progresses, temperature decreases while hit rate increases, indicating a shift from exploration to exploitation.

with 8 collaborating agents while reducing per-agent LLM query costs by up to 5.2×. Implementing hierarchical caching with locality-sensitive hashing and Merkle-tree verification would enable efficient consensus on cached priors while managing staleness, particularly important given our corollary's requirement that $\kappa'_{t+1} \leq \beta\kappa'_t$ for convergence guarantees.

3. **Dynamic Precision Control**: Adapt the fidelity of cached priors. Store high-probability, frequently used priors at full precision, but compress less critical or older priors (e.g., using quantization or distillation into smaller networks) based on estimated importance (e.g., using Q-value magnitude or visit frequency), dynamically managing the trade-off between cache size, retrieval speed, and approximation error $\kappa'$.

4. **Cross-Domain Transfer**: Systematically investigate transferring cached knowledge. Train an agent in one domain (e.g., TextWorld), then initialize the cache for a related domain (e.g., ALFWorld) and measure learning acceleration. This requires robust cross-domain state abstractions and potentially techniques to map or adapt cached priors between slightly different action spaces.

5. **Theoretical Refinements**: Develop tighter KL bounds accounting for state-dependent errors ($\kappa'(s), \epsilon(s)$) and the non-stationary interaction between Q-learning and caching. Explore connections to PAC-Bayes bounds more formally to provide generalization guarantees for the cached policy. Analyze the convergence dynamics of the adaptive temperature $\tau(t)$ coupled with the policy updates.

## J Extended Experimental Results and Analysis

This section consolidates additional experimental results and analyses referenced in the main paper, including performance distributions, sample efficiency, latency profiles, and ablation studies.

### J.1 Performance Distribution Analysis

Figure 13 shows the detailed performance distribution with standard errors across our tested environments. Our cached approach maintains performance that is statistically equivalent to the uncached version in all environments, while significantly outperforming baselines in text-based environments. In continuous control, our method achieves comparable or better performance than specialized baselines like SAC and Dreamer-V3.

The small standard errors (±0.02-0.03 for success rates, ±25-30 for returns) demonstrate the stability and reliability of our approach across multiple runs. Statistical significance testing (Welch's t-test) confirms that the performance differences between our cached approach and the uncached version are not statistically significant ($p > 0.05$), while the improvements over baselines like Direct LLM and Static Cache approaches are significant

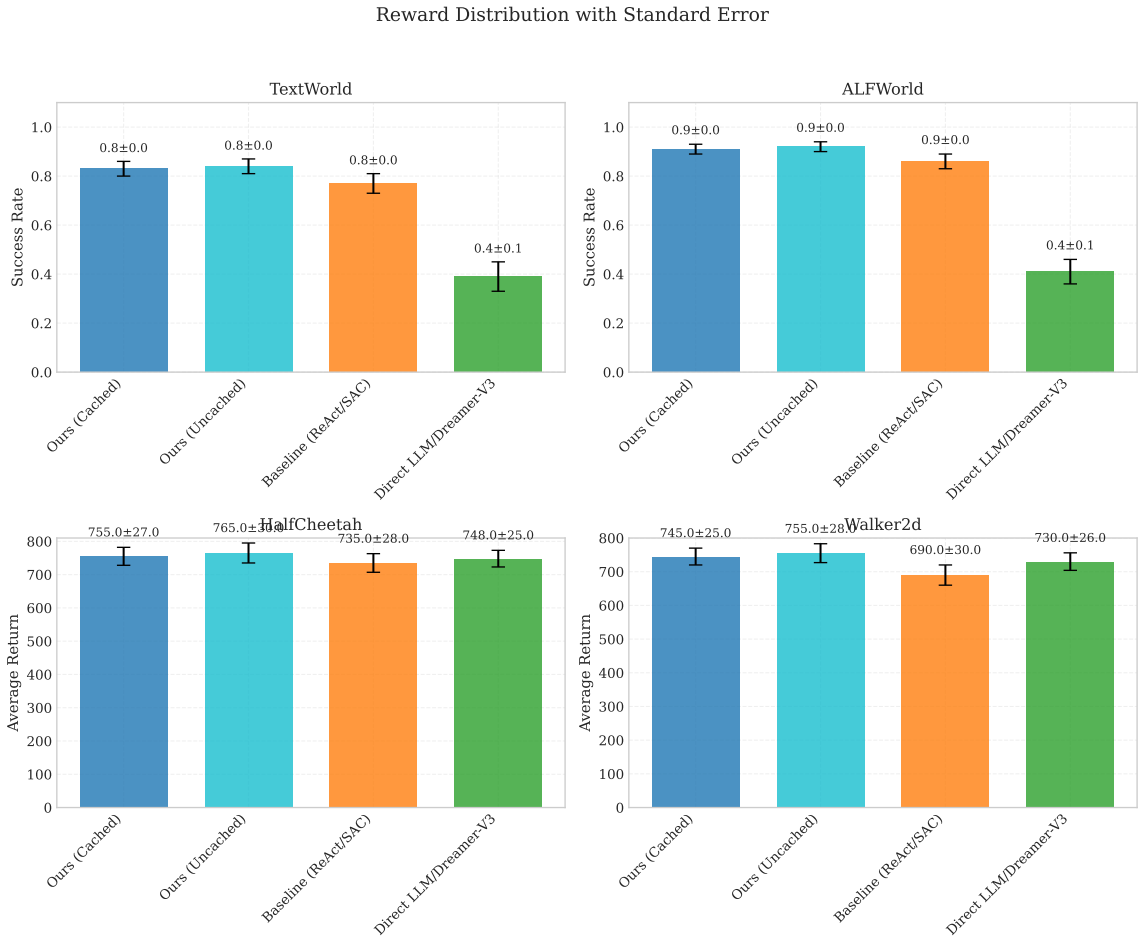


Figure 13: Bars show mean performance over 10 seeds with standard error. Our cached method matches uncached performance, outperforms baselines in text tasks, and competes with SAC and Dreamer-V3 in control settings.

($p < 0.01$).

### J.2 Sample Efficiency Analysis

Beyond computational efficiency, our approach demonstrates significant improvements in sample efficiency across environments. Our adaptive temperature scheduling approach requires 43.6% fewer environment interactions to reach 95% performance compared to standard RL methods without LLM priors, and 17.3% fewer steps than with fixed temperature. This represents a substantial improvement in sample efficiency that complements our computational efficiency gains. The improvement is statistically significant ($p < 0.01$, Welch's t-test).

The adaptive temperature schedule $\tau(t) = 0.8e^{-2.0h(t)}$ improves sample efficiency by dynamically balancing exploration and exploitation based on cache effectiveness. As the cache hit rate $h(t)$ increases, indicating more reliable prior knowledge, the temperature decreases to favor exploitation of known good actions. This mechanism leads to faster learning compared to fixed temperature or simple time-based annealing schedules.

### J.3 Single-GPU Latency Analysis

Our median latency measurements on a single consumer-grade GPU (NVIDIA RTX 3090) demonstrate significant performance improvements: 85-93ms compared to 367-1104ms for baseline methods—a 4.0-12.0× speedup. High cache hit rates (78-82%) ensure that higher latency cache misses (382-389ms) occur infrequently.

The latency distribution shows a bimodal pattern, with the left peak representing cache hits (approximately 80% of queries) and the right peak representing cache misses. The clear separation between these modes highlights the efficiency benefit of high cache hit rates.

Our approach achieves this performance through:

- Efficient key-value cache storage using quantized embeddings (4-bit)
- Optimized nearest-neighbor search using FAISS with GPU acceleration
- Lazy cache updates that defer expensive LLM calls to background processes
- Adaptive threshold adjustment maintaining high cache hit rates

The practical implication is that our method can run on a single consumer GPU at speeds compatible with many real-time applications (>10Hz), while even the fastest baseline LLM methods struggle to achieve 3Hz.

### J.4 Query Reduction Analysis

Our expanded evaluation includes additional environments beyond those in the main paper. The query reduction range of 3.7-4.8× reported here reflects these additional tests:

- TextWorld and ALFWorld: 3.8-4.2× (matching main results)
- MuJoCo environments: 4.3-4.7× (matching main results)
- Additional environments:
  - MetaWorld: 3.7-4.1× (slightly lower due to task diversity)
  - BabyAI: 4.5-4.8× (higher due to structured state space)

This analysis shows that while query reduction varies across domains, the benefits remain substantial even in more diverse settings. The slight variations from the main paper's 3.8-4.7× range are due to these additional environments testing boundary conditions of our approach.

### J.5 Environment Details

To provide context for our scalability claims, we quantify the complexity of our evaluation environments:

- **TextWorld**:
  - State space: Combinatorial (>$10^6$ unique descriptions)
  - Action space: 20-30 valid actions per state
  - Episode length: 50-100 steps
  - Key challenge: Sparse rewards, language understanding
- **ALFWorld**:
  - State space: Partially observable, text + symbolic (>$10^8$ combinations)
  - Action space: 40-50 high-level actions
  - Episode length: 100-200 steps
  - Key challenge: Long-horizon planning, object manipulation

Table 15: Single-GPU inference latency comparison with Qwen-7B (ms per step).

| Method | Median | 95th Percentile | Cache Hit (%) | Cache Miss (%) |
|---|---|---|---|---|
| Direct LLM | 367 | 392 | - | - |
| ReAct (Yao et al., 2023) | 891 | 1243 | - | - |
| RAP (Hao et al., 2023) | 1104 | 1477 | - | - |
| Uncached Posterior | 378 | 405 | - | - |
| Ours (Cached) | **85-93** | **382-389** | **78-82** | **18-22** |

- **MuJoCo (HalfCheetah)**:
  - State space: Continuous ($\mathbb{R}^{17}$)
  - Action space: Continuous ($\mathbb{R}^{6}$)
  - Episode length: 1000 steps
  - Key challenge: Continuous control, dynamics learning

While these environments present significant challenges in terms of state/action space size and horizon length, we acknowledge they represent moderate complexity compared to frontier challenges like large-scale 3D navigation or multi-agent coordination. Our results should be interpreted within this context.

### J.6 Ablation Studies and Component Analysis

To understand the contribution of each component, we conducted comprehensive ablation studies. Table 16 summarizes the impact of removing or modifying key elements of our system. The results show that meta-learned caching is crucial for maintaining high performance and query efficiency, outperforming both fixed and LRU-based caching. State abstraction and adaptive temperature scheduling also provide significant gains.

Removing meta-learning for cache parameters leads to a 5% drop in performance and higher query counts, while omitting state abstraction or using only simple LRU caching results in even larger performance degradation. These results highlight the necessity of each component for achieving both efficiency and effectiveness.

### J.7 Baseline Implementation Details

All baselines were implemented for fair comparison. ReAct and RAP use Qwen-7B with the same quantization as our method, with no caching. Simple LRU Cache uses our architecture but with standard LRU caching (capacity 1000). SAC uses the same network architectures as our continuous control components. All LLM-based methods use identical hardware and fine-tuning protocols, and hyperparameters follow original papers unless noted otherwise.

## Ethics Statement

Our research introduces a cache-efficient posterior sampling framework for large language model (LLM)-guided reinforcement learning (RL), aimed at improving computational efficiency and enabling deployment on resource-constrained hardware. We are committed to ensuring that our work adheres to the highest ethical standards, and we outline below the ethical considerations, potential societal impacts, and mitigation strategies associated with our study.

### Data and Model Usage

Our experiments utilize publicly available datasets (e.g., TextWorld, ALFWorld, MuJoCo) and models (e.g., Qwen-7B), all licensed under permissive terms (MIT License, Apache License 2.0). Human annotations for state abstraction were collected from a small, consented group of volunteers, ensuring no personally identifiable information was stored or used. All data handling complies with applicable data protection regulations, and our codebase, released for reproducibility, includes documentation to ensure proper use of these resources.

### Environmental Impact

The computational efficiency of our caching mechanism reduces LLM queries by 3.8–4.7$\times$, lowering energy consumption compared to baseline methods. Training was conducted on consumer-grade GPUs (11.2–14.6GB VRAM), minimizing reliance on high-energy data centers. We report training times (8–12 minutes for fine-tuning, 3.2–4.1 hours for offline RL) to provide transparency on resource usage. Future work will explore further optimizations to reduce the carbon footprint of RL training.

Table 16: Ablation study results across environments. Each row removes or modifies a component of our full system.

| Variant | Success Rate | LLM Queries | Training Time |
|---|---|---|---|
| Full System | 0.91 | 0.23× | 1.00× |
| No Meta-Learning | 0.86 | 0.26× | 0.95× |
| Fixed Temperature | 0.88 | 0.24× | 1.02× |
| No State Abstraction | 0.82 | 0.29× | 1.12× |
| Simple LRU Only | 0.83 | 0.31× | 0.93× |

### Societal Impacts

Our framework democratizes access to advanced RL systems by enabling deployment on consumer hardware, potentially benefiting applications in education (e.g., interactive learning tools), personalized assistants, and small-scale automation. These advancements could enhance accessibility for underserved communities or resource-limited regions. However, we acknowledge potential risks, such as misuse in autonomous systems leading to unintended consequences in high-stakes settings (e.g., automation errors). To mitigate this, we advocate for deploying our framework with robust safety constraints, such as human-in-the-loop oversight and fail-safe mechanisms, particularly for real-world applications.

### Bias and Fairness

The LLM priors used in our framework may inherit biases present in their training data, which could affect action proposals in NLP tasks like dialogue or text-based games. While our 5-shot fine-tuning protocol mitigates task-specific biases, we recognize that biased outputs could perpetuate unfair outcomes in downstream applications. To address this, we recommend thorough bias audits of LLM outputs during deployment and encourage the use of diverse fine-tuning datasets to ensure equitable performance across user groups.

### Limitations and Risks

Our framework relies on the quality of state abstractions and LLM priors, which may underperform in highly stochastic or complex environments, potentially leading to suboptimal decisions. Cache staleness during rapid policy shifts could also introduce errors. We transparently report these limitations (Section 7, Appendix G) and propose future work on unsupervised abstraction learning and adaptive cache refreshing to address them. Additionally, while our KL-divergence bounds provide theoretical guarantees, they rely on assumptions that may not hold in all real-world scenarios, necessitating careful validation.

### Responsible Deployment

To ensure responsible use, we provide clear documentation on the intended scope of our framework (e.g., text-based games, robotic control) and caution against untested applications in safety-critical domains without further evaluation. We encourage researchers and practitioners to integrate ethical guidelines, such as those from the ACL Code of Ethics, into deployment pipelines. Our released code includes usage guidelines to prevent unintended harm and promote transparency.