BELIEF-CENTRIC REINFORCEMENT LEARNING FOR LLM AGENTS IN POMDP TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Decision making under uncertainty is a central challenge for large language model (LLM) agents (e.g., robotics). Partially Observable Markov Decision Processes (POMDPs) provide a principled framework for modeling decision-making from incomplete observations. In POMDP tasks, agent's performance depends on two mutually coupled capabilities: how to estimate the state-relevant belief information, and how to disentangle the estimation of failure beliefs and assign the accurate credit to actions. To address this challenge, this paper proposes **BeliefRL**, a belief-centric RL framework that jointly adapts belief-state estimation and policy learning. BeliefRL first maintains a structured belief state over established, ruled-out, and unresolved information, continually adapts the belief-state model using simulator-grounded supervision. These state-reference discrepancies are further measured as a reliability signal to modulate policy updates. BeliefRL is validated on four partially observable tasks in VirtualHome and Overcooked. Compared with POAD, the strongest overall baseline, BeliefRL improves averaged return by 44.1% and normalized training AUC by 34.8%. Ablations support the complementary benefits of training-time belief adaptation and reliability-aware policy optimization.

1 INTRODUCTION

Large language models (LLMs), advanced by reinforcement learning (RL), have demonstrated remarkable decision-making capabilities in diverse interactive language tasks (Carta et al., 2023; Wang et al., 2025). Many existing tasks (e.g., math, coding and computer use) can be modeled as a Markov Decision Process (MDP), where decision-relevant information is fully observable to the agent through the current observation or historical context. In contrast, many real-world sequential decision problems are inherently partially observable (Kaelbling et al., 1998; Lauri et al., 2022; Sun et al., 2024). For example, in a household search task, an agent may currently observe only the kitchen while the target object is unobservable, located in another room or hidden inside a container. Partially Observable Markov Decision Processes (POMDPs) provide a flexible framework for modeling such problems, in which effective decisions depend not only on the current observation, but also on the relevant information about the task accumulated from the interaction history (Wang et al., 2026a). Directly using current observation can lead to *perceptual aliasing*, where different underlying states are indistinguishable from the policy (Tan et al., 2024; Wen et al., 2024), while using the entire interaction history as context provides the relevant information only implicitly and can make long-horizon policy learning more difficult (Wang et al., 2023a). As an exploratory illustration in the VirtualHome Food Preparation task, Figure 1 shows that a full state RL agent out-

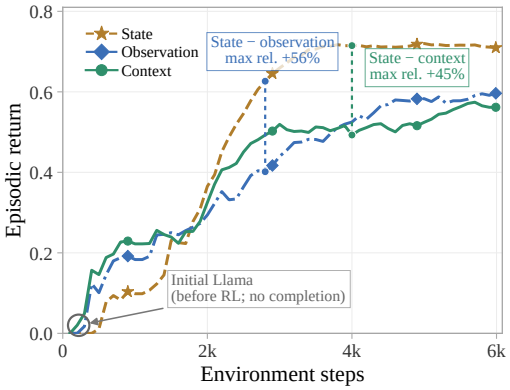


Figure 1: In the VirtualHome Food Preparation task, RL returns for current observations only (blue), historical context (green) and full state (orange). State plays a large role in performance.

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

performs agents with current observation and historical context, a maximum return improvement of approximately 50%.

An agent can mitigate partial observability by learning memory functions that condense past sequences of interactions into a memory state. Popular memory functions include recurrent neural networks (RNNs) (Hausknecht & Stone, 2015), transformers (Ni et al., 2021) and prompts (Lidayan et al., 2025). This implicit memory function cannot maintain accurate representations of the underlying state, and the uncertainty obscures the causal impact of agent actions (Sun et al., 2024; Liao et al., 2026; Bagaria et al., 2026). Recently, Zou et al. (Zou et al., 2026b;a) and Tang et al. (Tang et al., 2026a) propose to track the deviation of belief and decouple the failures of the trajectory from the actions. However, some existing belief-tracking methods rely on model-internal estimates of task progress (e.g., changes inferred between consecutive steps), which may be unreliable when they are not explicitly grounded in environment states. In summary, in POMDP tasks, the LLM agent's performance depends on two mutually coupled capabilities: *belief state estimation*, which determines what task-relevant information to track and how to update it, and *belief-based policy optimization*, which accounts for belief-estimation errors when assigning credit to actions.

To address these challenges, we propose **BeliefRL** (Belief-Centric RL), a framework that couples structured belief-state estimation with reliability-aware policy optimization. The first contribution of BeliefRL is an LLM-based Belief State Model (BSM) that recursively maintains task-relevant logical predicates and assigns each predicate one of three values—*True*, *False*, or *Unknown*, to distinguish established, ruled-out, and unresolved information. During RL, our second contribution is to co-train the BSM and policy of agent, where the BSM is supervised by simulator-grounded reference states. We measure the discrepancies between the BSM prediction and reference grounded states and use it as a reliability signal to modulate actor advantages, attributing the accurate credit to policy updates. Finally, we evaluate BeliefRL on four partially observable tasks: Food Preparation and Entertainment in VirtualHome, and Tomato Salad and Tomato Lettuce Salad in Overcooked. Compared with POAD, the strongest overall baseline among GLAM, TWOSOME, ARCHER, NTPO, and POAD, BeliefRL improves task-averaged final-window environment return by 44.1% and normalized training AUC by 34.8%.

2 PRELIMINARIES

We formalize the task as a POMDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}, \gamma \rangle$, where $\mathcal{S}$, $\mathcal{A}$, and Ω denote the latent state, action, and observation spaces, respectively; $\mathcal{T}$ is the transition model, $\mathcal{R}$ the reward function, $\mathcal{O}$ the observation model, and $\gamma \in [0, 1]$ the discount factor. A language goal g specifies the task. At time t, the agent observes $o_t \sim \mathcal{O}(\cdot \mid s_t)$, selects an action a_t based on its interaction history $\mathcal{H}_t = (o_0, a_0, \ldots, a_{t-1}, o_t)$ and goal g, receives $r_t^{env} = \mathcal{R}(s_t, a_t)$, and transitions to $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$. The objective is to maximize the expected discounted environment return over T decision steps:

$$J_{env}(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t r_t^{env} \right]. \tag{1}$$

A belief state $b_t = P(s_t \mid \mathcal{H}_t, g)$ summarizes the agent's uncertainty about the latent state given its interaction history and task goal. Under the POMDP assumptions, it is a sufficient statistic of the history for control, given the goal and the remaining horizon (Kaelbling et al., 1998; Wijesundara et al., 2025). Exact belief inference is generally intractable in the embodied settings considered here, motivating learned representations that retain task-relevant information from the history. Their estimation errors can affect both action selection and value estimation, making the quality of the learned state representation relevant to policy optimization.

3 METHODOLOGY

BeliefRL jointly learns a recursive belief state model and a policy for partially observable language tasks. Figure 2(a) illustrates how the BSM updates structured beliefs to guide action selection. During training, simulator-grounded reference beliefs support BSM supervision and reliability-weighted policy optimization (Figure 2(b)). Policy updates alternate with training-time BSM adaptation; ref-

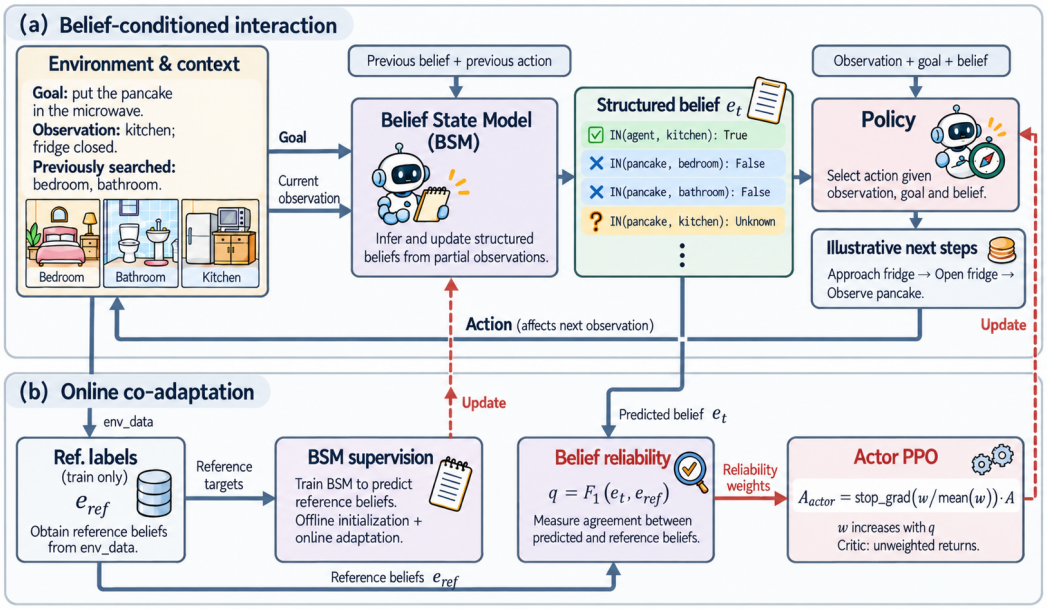


Figure 2: BeliefRL framework. (a) The BSM recursively updates a structured belief from the previous belief, previous action, current observation, and task goal. The policy conditions on the belief, observation, and goal to select actions. (b) Reference beliefs supervise the BSM and provide an agreement-based reliability signal for weighting actor advantages during training.

erence beliefs are not required at deployment. Appendix A provides the training pseudocode and module interactions (Figure 7).

3.1 Belief-State Modeling

Predicate-Based Belief Representation. We represent task-relevant latent information with a predicate set $\mathcal{P} = \{p_1, \ldots, p_N\}$. Each predicate p_i represents a symbolic relation among task-relevant entities, locations, containers, or interaction states.

At decision step t, the structured belief state is

$$e_t = \{(p_i, v_{t,i})\}_{i=1}^{N}, \qquad v_{t,i} \in \{\texttt{True}, \texttt{False}, \texttt{Unknown}\}, \tag{2}$$

where `True`, `False`, and `Unknown` indicate established, ruled-out, and unresolved information, respectively; `Unknown` is assigned when observations support neither polarity. This representation retains task-relevant information after it becomes unobservable (Zhao et al., 2025).

Recursive Belief Estimation. We parameterize the belief estimator with an LLM-based Belief State Model (BSM) with parameters ψ. At each step, the BSM updates the previous belief using the previous action, current observation, and task goal. It does not reconstruct the state from the full interaction history. Specifically,

$$x_t = \text{Prompt}(e_{t-1}, a_{t-1}, o_t, g), \qquad P_\psi(e_t \mid x_t) = \prod_{j=1}^{|e_t|} P_\psi(e_{t,j} \mid x_t, e_{t,<j}). \tag{3}$$

The prompt specifies the allowed predicate schema. At the initial step, the BSM sets unresolved predicates to `Unknown`. At each later step, it uses its previous prediction and updates only predicates supported by the new observation.

Offline BSM Initialization. We initialize the BSM by supervised fine-tuning of a pretrained LLM on offline belief-transition data $\mathcal{D}$, establishing basic belief-update competence before interactive

agent training. Using simulator-grounded reference beliefs $\bar{e}_t$ as targets, we minimize

$$\mathcal{L}_{\text{offline}}(\psi) = -\mathbb{E}_{d\sim\mathcal{D}}\left[\frac{1}{|\bar{e}_t|}\sum_{j=1}^{|\bar{e}_t|}\log P_\psi(\bar{e}_{t,j} \mid x_t, \bar{e}_{t,<j})\right]. \tag{4}$$

The loss applies only to reference-belief target tokens; x_t uses the reference or predicted previous belief according to scheduled sampling (Bengio et al., 2015). Appendices D and E.2 detail data construction and training settings, respectively.

3.2 Belief-Based Policy Optimization

After offline initialization, BeliefRL improves the belief estimator and policy using trajectories from the current policy. The actor and critic condition on

$$X_t = (e_t, o_t, g), \tag{5}$$

where e_t is the belief state predicted by the BSM. Each rollout provides environment rewards for policy optimization. It also provides simulator-grounded reference states for BSM supervision during agent training.

BSM Adaptation During Agent Training. During interactive agent training in the simulator, we further adapt the BSM using trajectories collected by the current policy. Simulator-grounded reference beliefs provide supervision exclusively during training. Each rollout prompt x_t, which contains the previous BSM prediction e_{t-1}, is paired with the reference belief $\bar{e}_t$. For n_t valid target tokens, we optimize

$$\ell_t^{\text{BSM}} = -\frac{1}{n_t}\sum_{j=1}^{n_t}\log P_\psi(\bar{e}_{t,j} \mid x_t, \bar{e}_{t,<j}), \qquad \mathcal{L}_{\text{BSM}} = \frac{1}{|\mathcal{B}_{\text{BSM}}|}\sum_{t\in\mathcal{B}_{\text{BSM}}}\ell_t^{\text{BSM}}. \tag{6}$$

We update the BSM adapter by minimizing this loss over target tokens.

Belief Reliability Estimation. We parse the predicted state e_t and reference state $\bar{e}_t$ into tuple sets E_t and T_t. Each tuple contains a predicate name, its arguments, and a truth value. Let

$$M_t = |E_t \cap T_t|. \tag{7}$$

We define predicate-set agreement as

$$q_t = \begin{cases} \dfrac{2M_t}{|E_t| + |T_t|}, & |T_t| > 0, \\ 0, & |T_t| = 0. \end{cases} \tag{8}$$

The score does not depend on tuple order. It matches complete predicate tuples, including their `False` and `Unknown` values. The score is zero when the reference is empty or when the prediction is invalid and the reference is nonempty. We use q_t as a reliability signal during training. The acting policy cannot access it.

Reliability-Weighted Policy Optimization. Let θ and ϕ denote the actor and critic parameters, respectively. For an action $a_t = (\tau_{t,1}, \ldots, \tau_{t,L_t})$, the critic estimates the pre-token value

$$V_t^i = V_\phi(X_t, \tau_{t,<i}). \tag{9}$$

Token-level GAE (Schulman et al., 2015; Wen et al., 2024) computes advantages A_t^i from rollout values and environment rewards r_t^{env}. Appendix F.1 gives the token recursion and action-scoring details. We convert q_t into an action-level weight to account for belief reliability during actor learning.

$$w_t = \text{clip}(w_{\min} + \lambda_q q_t, w_{\min}, w_{\max}), \qquad \bar{w}_t = \frac{w_t}{\max\left(\frac{1}{|\mathcal{B}|}\sum_{s\in\mathcal{B}} w_s, 10^{-6}\right)}, \qquad \widetilde{A}_t^i = \text{sg}(\bar{w}_t)\widehat{A}_t^i, \tag{10}$$

where $\widehat{A}_t^i$ is the standardized token advantage in PPO minibatch $\mathcal{B}$ and $\text{sg}(\cdot)$ denotes stop-gradient. The same positive weight is applied to every token of an action. Following POAD (Wen et al., 2024), we use its action-decomposed, clipped token-level PPO surrogate for the actor and the corresponding token-level clipped value objective for the critic. Appendix F.2 gives their definitions.

Alternating BSM–Policy Updates. At each iteration, we collect trajectories for policy optimization and BSM supervision. We write the actor, critic, and BSM objectives as the following joint loss:

$$\mathcal{L}_{\text{BeliefRL}}(\theta, \phi, \psi) = -\frac{1}{K}\sum_{k=1}^{K} J_{\text{actor}}^{\mathcal{C}_k} + \mathcal{L}_V + \mathcal{L}_{\text{BSM}}, \tag{11}$$

where K is the number of actor accumulation chunks. At each iteration, we first update the actor and critic with PPO. We then update the BSM using stored prompt–reference pairs while keeping the actor and critic fixed. Rollout beliefs are fixed during PPO, so BSM gradients do not pass through belief generation. Algorithm 1 in Appendix A describes BSM initialization and adaptation. Algorithm 2 gives the complete alternating training procedure. At deployment, the BSM and policy parameters remain fixed. The BSM recursively updates its predicted belief from the previous belief, previous action, current observation, and task goal, without access to simulator states or reference beliefs.

3.3 Theoretical Analysis

We analyze belief errors in credit assignment and conditionally bound their contribution to actor-gradient error.

Belief Deviation and Credit Assignment. For the analysis, we associate each textual BSM state with a latent belief b. For a fixed target state $s^\star$ with $b(s^\star) > 0$, we define $\Psi(b) = -\log b(s^\star)$. Let $I(b)$ denote the ideal expected one-step decrease in Ψ, and let $c(b)$ denote the additional expected potential from the imperfect update $\mathcal{B}_\psi$. If ideal progress is bounded by $I(b) \le \bar{I}$ and update error satisfies $c(b) \ge m\Psi(b) - c_0$ for $\Psi(b) \ge U_0$, with $m > 0$ and $c_0, U_0 \ge 0$, Proposition 1 implies

$$\Psi(b) \ge \max\{U_0, (\bar{I} + c_0)/m\} \quad \Longrightarrow \quad \mathbb{E}_{a,o}[\Psi(\mathcal{B}_\psi(b, a, o))] \ge \Psi(b). \tag{12}$$

Thus, above this threshold, the expected belief deviation does not decrease after one update.

For credit assignment, consider environment-step GAE before standardization (Schulman et al., 2015), $\widehat{A}_t^{\text{env}} = \sum_{j=0}^{T-t-1} \alpha^j \delta_{t+j}$, where $\alpha = \gamma\lambda_{\text{GAE}} \in [0, 1]$ and $\delta_k = r_k^{\text{env}} + \gamma(1 - d_k)V_{k+1} - V_k$. Fix a split point t_S. Before t_S, assume that $\mathbb{E}[\delta_k] \le u_k$. After t_S, assume that $\mathbb{E}[\delta_k] \le -r$, where $r > 0$. Then Proposition 2 yields

$$\mathbb{E}[\widehat{A}_t^{\text{env}}] \le \underbrace{\sum_{j=0}^{t_S-t-1} \alpha^j u_{t+j}}_{S_{\text{pre}}(t)} - r \underbrace{\sum_{j=t_S-t}^{T-t-1} \alpha^j}_{S_{\text{tail}}(t)}, \qquad t < t_S. \tag{13}$$

If $rS_{\text{tail}}(t) > S_{\text{pre}}(t)$, the expected advantage of the informative prefix is negative (Zou et al., 2026b). We assume this negative-tail condition separately from the preceding drift condition.

Reliability-Weighted Gradient Error. We fix the rollout distribution, policy parameters, and valid-token sampling rule. Write $\zeta = \nabla_\theta \log P_\theta(\tau_i \mid X, \tau_{<i})$ and $\widehat{A} = A^\star + \varepsilon_A$, where q is the decision's agreement score, $A^\star$ is a same-scale reference sharing non-belief estimation effects with $\widehat{A}$, and ε_A is the additional belief-induced error. Let $w(q)$ be the positive, nondecreasing raw weight defined in Equation (10). We define

$$g_w^{\text{obs}} = \frac{\mathbb{E}[w(q)\zeta\widehat{A}]}{\mathbb{E}[w(q)]}, \qquad g_w^{\text{ref}} = \frac{\mathbb{E}[w(q)\zeta A^\star]}{\mathbb{E}[w(q)]}. \tag{14}$$

Theorem 1 (Reliability-weighted belief-induced gradient-error bound) *Suppose $\zeta A^\star$ and $\zeta\varepsilon_A$ are integrable, $0 < w_{\min} \le w(q) \le w_{\max} < \infty$, and w is nondecreasing. If $\mathbb{E}[\|\zeta\varepsilon_A\| \mid q] \le C(1 - q)$ almost surely for finite $C \ge 0$, then*

$$\|g_w^{\text{obs}} - g_w^{\text{ref}}\| \le C\frac{\mathbb{E}[w(q)(1 - q)]}{\mathbb{E}[w(q)]} \le C\mathbb{E}[1 - q]. \tag{15}$$

Calibration of this component is an additional assumption, not implied by predicate agreement. Shared value-estimation, finite-horizon, and normalization effects, and token-surrogate discrepancies, remain outside the bound and may persist at $q = 1$. Appendix G gives the full assumptions and proofs.

4 EXPERIMENTS

We evaluate BeliefRL's belief-state prediction and policy performance, then examine the contributions of joint BSM–policy adaptation and reliability weighting. We further assess robustness to environment shifts, retention of language ability, and performance on the more challenging Office Preparation task.

4.1 EXPERIMENTAL SETUP

Environments. We evaluate BeliefRL in VirtualHome and Overcooked, which have distinct observation spaces, action spaces, and reward structures. The four main tasks are Food Preparation and Entertainment in VirtualHome, and Tomato Salad and Tomato Lettuce Salad in Overcooked. Due to space limitations, detailed descriptions of the environments, task configurations, and experimental settings are provided in Appendix C.

Model Configurations. BeliefRL uses Meta-Llama-3.1-8B-Instruct with separate LoRA adapters (Hu et al., 2021) for the policy and BSM. The BSM is initialized through offline supervised fine-tuning (SFT) and further adapted through state supervision during agent training, while the policy is optimized using token-level PPO. Training is performed on NVIDIA A100 GPUs.

Baselines. We evaluate BeliefRL against GLAM, TWOSOME, ARCHER, NTPO, and POAD on the same benchmark tasks in VirtualHome and Overcooked. The experimental question is whether training-time adaptation of an explicit belief interface, together with reliability-aware actor optimization, improves language-agent RL. GLAM provides a short-term history-memory baseline, while BeliefRL w/o BSM serves as a direct reference for assessing the contribution of the explicit BSM interface. Component ablations further isolate the effects of training-time BSM adaptation and reliability weighting, enabling us to disentangle the contributions of joint BSM–policy adaptation and reliability-aware optimization.

Evaluation Metrics. Unless otherwise stated, we compute all reported metrics from environment-only episodic returns r_t^{env}. These metrics include learning curves, final-window returns, AUC values, and bootstrap comparisons. For the four main tasks, we report five runs per method using episodic-return logs collected during training.

Appendix B provides supplementary results and analyses for the experiments in this section.

4.2 MAIN RESULTS

Belief-State Estimation Accuracy. We assess belief-state prediction accuracy after offline SFT and subsequent adaptation during agent training using 20 episodes per task, with actions sampled uniformly from the available action set. At each step, the BSM conditions on its own previous prediction, thereby evaluating state estimation under recursive belief updates. We report exact match, defined as the fraction of steps for which the canonicalized predicted belief agrees with the simulator-grounded reference state. Figure 3 compares BSM prediction accuracy after offline SFT and subsequent agent training. Following agent training, exact match increases by approximately 2.0, 21.5, and 1.9 percentage points on Food Preparation, Tomato Salad, and Tomato Lettuce Salad, respectively, while remaining at 100.00% on Entertainment. These results support the effectiveness of BeliefRL's belief-state estimation and suggest that adaptation during agent training can further improve prediction accuracy.

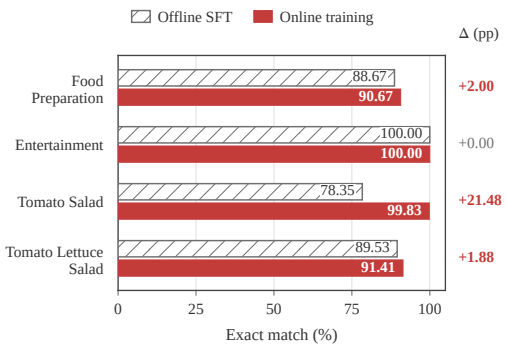
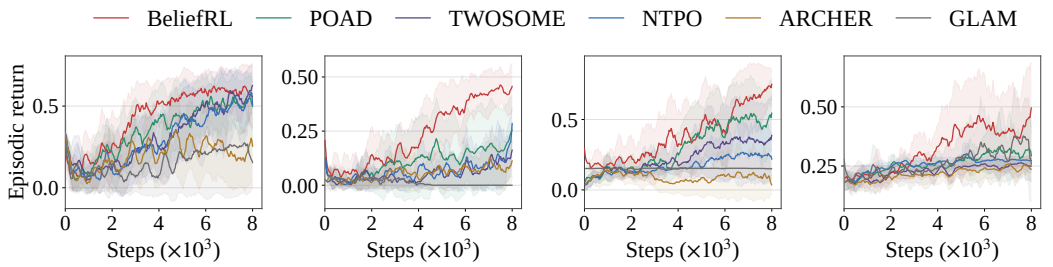


Figure 3: Comparison of prediction accuracy of BSM after offline SFT and subsequent agent training.



(a) Food Preparation (b) Entertainment (c) Tomato Salad (d) Tomato Lettuce Salad

Figure 4: Learning curves on the four main benchmark tasks. Curves report smoothed environment-only episodic return averaged over five seeds and the shaded band denotes seed standard deviation.

Policy Learning Performance. Figure 4 compares the training dynamics of BeliefRL and the baseline methods across the four benchmark tasks. BeliefRL achieves the highest final-window environment return on all four tasks relative to the equal-weighted mean of the five baselines. As shown in Table 5, it also obtains the highest normalized training AUC on every task, indicating stronger aggregate learning progress throughout training.

Table 1: Final-window environment-only returns over five runs, using each run's last 10% of records. Δ is BeliefRL minus this mean with a pointwise 95% bootstrap CI.

Task	BeliefRL	All baselines	Δ [95% CI]
Entertainment	0.447	0.092	+0.355 [0.287, 0.423]
Food Preparation	0.631	0.422	+0.209 [0.137, 0.282]
Tomato Salad	0.703	0.271	+0.432 [0.310, 0.538]
Tomato Lettuce Salad	0.402	0.282	+0.120 [-0.033, 0.280]

We compute the aggregate baseline as the equal-weighted mean of the five baseline methods. Appendix B.1 describes the bootstrap procedure and effect sizes. Appendix D describes the offline data-construction pipeline. Appendix E reports the model and optimization configurations.

Frozen-Policy Deployment Performance. To assess whether the learned policy retains its advantage under a fixed deployment policy, we additionally evaluate the trained checkpoints on Tomato Lettuce Salad with both BSM and policy parameters fixed. All methods use greedy action selection and are evaluated on the same 50 randomly selected test seeds, which are distinct from the training seeds. BeliefRL achieves a mean environment-only return of 0.562, compared with 0.282 for the strongest baseline, GLAM, corresponding to an absolute difference of 0.280. These results provide additional evidence for the effectiveness of the BeliefRL training framework under frozen-policy deployment in this setting.

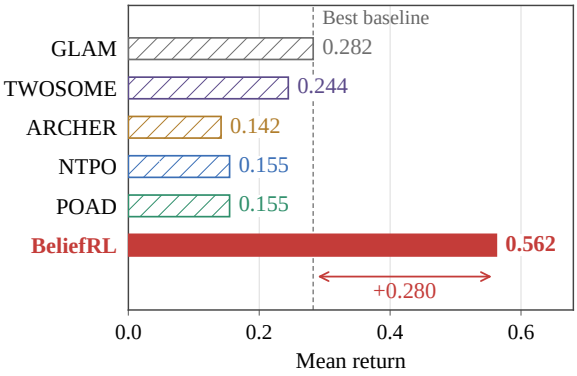
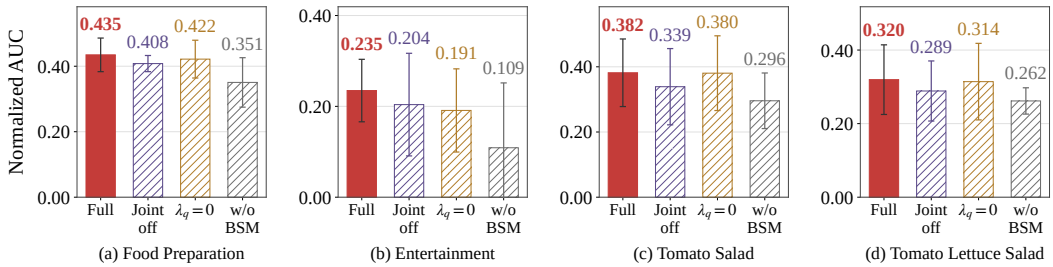


Figure 5: Frozen-policy mean returns on Tomato Lettuce Salad, evaluated greedily on 50 shared random test seeds.

4.3 Ablation: Joint BSM–Policy Coupling and Reliability Weighting

Figure 6 compares the full method with ablations of joint BSM–policy adaptation and reliability weighting. The full configuration achieves the highest normalized training AUC on all four tasks. Removing either component lowers the normalized training AUC. These results support combining training-time BSM adaptation with reliability-weighted policy updates.



(a) Food Preparation (b) Entertainment (c) Tomato Salad (d) Tomato Lettuce Salad

Figure 6: Component ablations: normalized training AUC (mean ± sample SD, $n = 5$). Full denotes BeliefRL; Joint-RL-off, $\lambda_q = 0$, and w/o BSM remove joint BSM–policy adaptation, reliability weighting, and the BSM interface, respectively.

4.4 Robustness and Generalization to Unseen Scenarios

Table 2: Component ablation under aligned OOD shifts on Food Preparation.

Task	BeliefRL w/o BSM	BeliefRL	Δ OOD
Fridge Shift	0.400 ± 0.000	0.537 ± 0.013	+0.137
Full Spatial Shift	0.287 ± 0.041	0.437 ± 0.024	+0.150

We evaluate two OOD shifts on Food Preparation. *Fridge Shift* randomizes the fridge room. *Full Spatial Shift* randomizes the spawn, fridge, microwave, and pancake locations while preserving the task semantics and actions. Each method uses three seeds and 100 episodes per seed. We compare BeliefRL with BeliefRL w/o BSM under the same shifts to isolate the contribution of the BSM interface. Table 2 shows that BeliefRL exceeds BeliefRL w/o BSM by 0.137 on Fridge Shift and by 0.150 on Full Spatial Shift. These results suggest that the explicit BSM interface improves robustness under the tested spatial shifts.

4.5 Impact on Original Language Ability

Table 3: Zero-shot language ability after RL fine-tuning on Food Preparation.

Method	ARC-C	HSwag	PIQA	MMLU	Avg
LLaMA-3.1-8B	0.559	0.795	0.803	0.684	0.710
GLAM	0.544	0.791	0.799	0.676	0.703
TWOSOME	0.559	0.797	0.804	0.686	0.711
ARCHER	0.557	0.796	0.801	0.687	0.710
NTPO	0.554	0.798	0.801	0.689	0.711
POAD	0.550	0.797	0.803	0.685	0.709
BeliefRL	0.567	0.799	0.801	0.688	0.714

We test whether RL fine-tuning on Food Preparation affects the model's general language ability. Following POAD (Wen et al., 2024), we evaluate the Food Preparation checkpoints on ARC-Challenge, HellaSwag, PIQA, and MMLU. We use Language Model Evaluation Harness for evaluation (Gao et al., 2021). All RL methods use the same backbone and differ only in their policy adapters. We use the untuned LLaMA-3.1-8B as the reference. Higher scores are better. Table 3 shows that BeliefRL achieves the highest average score among the evaluated RL methods (0.714). This score is slightly higher than that of the untuned LLaMA-3.1-8B (0.710). Scores remain close to the untuned model's scores on all four benchmarks. These results suggest that RL fine-tuning preserves the model's general language ability. Figure 8 shows score changes on each benchmark. Table 9 reports scores for individual MMLU subjects.

4.6 Additional Experiment: Office Preparation

Table 4: Environment-only episodic return on Office Preparation.

Metric	GLAM	ARCHER	TWOSOME
Last return	0.000 ± 0.000	0.000 ± 0.000	0.164 ± 0.126
Metric	**NTPO**	**POAD**	**BeliefRL**
Last return	0.190 ± 0.268	0.221 ± 0.231	**0.390 ± 0.278**

Office Preparation uses a separate apartment layout and requires the agent to search among more candidate containers before completing a longer sequence of actions. We include it to assess BeliefRL on a harder VirtualHome task beyond the four main benchmarks. Table 4 reports the results. BeliefRL achieves the highest mean last episodic return among the displayed methods (0.390 ± 0.278), followed by POAD (0.221 ± 0.231).

5 Related Works

Language Agents for POMDP. Language models support embodied planning and interaction (Ahn et al., 2022; Huang et al., 2022; Driess et al., 2023; Liang et al., 2023; Zeng et al., 2022; Yao et al., 2022; Wang et al., 2023b; Shridhar et al., 2020); LLaRP further trains an embodied language policy directly through environment interaction (Szot et al., 2023). Capabilities are assessed by interactive benchmarks (Liu et al., 2024; Zhou et al., 2024a; Yang et al., 2025), while MIKASA targets memory-intensive RL tasks, including partially observable robotic manipulation (Cherepanov et al., 2026). LLM-POP studies state abstraction and information gathering in task-dependent POMDPs (Sun et al., 2024). Online, hierarchical, and token-level RL further optimize language agents (Carta et al., 2023; Tan et al., 2024; Zhou et al., 2024b; Wen et al., 2024), alongside embodied post-training (Fei et al., 2025; Zhai et al., 2024; Wang et al., 2025; Chen et al., 2025) and search-guided learning (Feng et al., 2023). Recent methods improve long-horizon credit assignment through state-conditioned objectives, milestone-level advantages, and transition-aware updates (Feng et al., 2026; Wang et al., 2026b; Liu et al., 2026). Other agents incorporate explicit reasoning or state abstractions (Lidayan et al., 2025; Jiang et al., 2026; Wang et al., 2026a; Liao et al., 2026; Bagaria et al., 2026); world-model prediction residuals also trigger online adaptation in continuous-control robotics (Domberg & Schildbach, 2026). Recent work further decouples belief estimation from action selection with explicit verbalized uncertainty and joint RL (Singh et al., 2026). BeliefRL focuses on adapting an explicit state estimator alongside the policy and using its reliability to modulate actor updates.

Belief State Modeling for POMDP. Classical belief-space planning typically requires transition and observation models (Åström, 1965; Kaelbling et al., 1998; Pineau et al., 2003; Lauri et al., 2022), while model-free POMDP learning studies scalable policy optimization from partial observations (Cai et al., 2022). Learned latent and predictive representations further improve scalability (Guo et al., 2018; Wang et al., 2023a; Allen et al., 2024; Kuo et al., 2026). Recent explicit representation and interface precedents include probabilistic programs in POMDP Coder (Curtis et al., 2025), progress-aware history in PABU (Jiang et al., 2026), probabilistic memory in Belief Memory (Liao et al., 2026), and recursive belief-conditioned control (Bagaria et al., 2026). Recent LLM-based planners also induce POMDP world models from interaction data (Pinductor), construct hypotheses for Bayesian belief-space planning (Tru-POMDP), or maintain an external Bayesian state filter (Belief-State Engine) (Six et al., 2026; Tang et al., 2026b; Chattopadhayay & Halder, 2026). Other work learns uncertainty-aware belief states for partially observable cooperative RL, separating estimation from policy learning (Pritz & Leung, 2025). Addressing belief-dependent learning failures, T^3 truncates uninformative trajectory tails (Zou et al., 2026b), while AREW reweights advantages using directional critiques to mitigate information self-locking (Zou et al., 2026a). GUI diagnostics additionally reveal conflicts between visual evidence and structural state information and examine consistency gating (Zhang et al., 2026). BeliefRL does not introduce a new belief representation; it studies co-adaptation of an explicit BSM and the policy during agent training, with actor updates weighted by predicate-set agreement.

6 Conclusion

In this work, we introduced BeliefRL, a reliability-aware belief co-adaptation framework for partially observable environments that improves language-agent decision making. BeliefRL couples recursive symbolic state estimation with supervised BSM adaptation during agent training and uses agreement between model states and reference states to modulate policy updates. Across four tasks in VirtualHome and Overcooked, BeliefRL consistently achieves higher mean final-window returns than the equal-weighted average of all baselines. The ablation results further show that training-time BSM adaptation and reliability weighting are complementary. In addition, evaluations of OOD generalization, language retention, and Office Preparation further indicate that BeliefRL's state-aware learning improves robustness and transfers across task configurations. We hope this work provides useful insights into reliable state estimation and policy learning and supports the development of next-generation intelligent agents.

AI USE STATEMENT

AI tools assisted with language editing, manuscript organization, LaTeX preparation, and local scripts for analysis and figure assembly. The authors are responsible for the scientific content, experimental design, reported measurements, and interpretation of the results.

REFERENCES

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

Cameron Allen, Aaron Kirtland, Ruo Yu Tao, Sam Lobel, Daniel Scott, Nicholas Petrocelli, Omer Gottesman, Ronald Parr, Michael Littman, and George Konidaris. Mitigating partial observability in sequential decision processes via the lambda discrepancy. *Advances in Neural Information Processing Systems*, 37:62988–63028, 2024.

Karl Johan Åström. Optimal control of markov processes with incomplete state information i. *Journal of mathematical analysis and applications*, 10:174–205, 1965.

Vaidehi Bagaria, Bijo Sebastian, and Nirav Kumar Patel. Recursive belief vision language action models. *arXiv preprint arXiv:2602.20659*, 2026.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

Qi Cai, Zhuoran Yang, and Zhaoran Wang. Reinforcement learning from partial observation: Linear function approximation with provable sample efficiency. In *International Conference on Machine Learning*, pp. 2485–2522. PMLR, 2022.

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

Arnab Chattopadhayay and Debdipta Halder. Belief-state engine: Augmenting llms for principled planning under partial observability. *arXiv preprint arXiv:2609.10036*, 2026.

Hanyang Chen, Mark Zhao, Rui Yang, Qinwei Ma, Ke Yang, Jiarui Yao, Kangrui Wang, Hao Bai, Zhenhailong Wang, Rui Pan, et al. Era: Transforming vlms into embodied agents via embodied prior learning and online reinforcement learning. *arXiv preprint arXiv:2510.12693*, 2025.

Egor Cherepanov, Nikita Kachaev, Alexey Kovalev, and Aleksandr Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 9759–9815, 2026.

Aidan Curtis, Hao Tang, Thiago Veloso, Kevin Ellis, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Llm-guided probabilistic program induction for pomdp model estimation. In *Conference on Robot Learning*, pp. 3137–3184. PMLR, 2025.

Fabian Domberg and Georg Schildbach. Self-adapting robotic agents through online continual reinforcement learning with world model feedback. *arXiv preprint arXiv:2603.04029*, 2026.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Zhaoye Fei, Li Ji, Siyin Wang, Junhao Shi, Jingjing Gong, and Xipeng Qiu. Unleashing embodied task planning ability in llms via reinforcement learning. *arXiv preprint arXiv:2506.23127*, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*, 2023.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo A Pires, and Rémi Munos. Neural predictive belief representations. *arXiv preprint arXiv:1811.06407*, 2018.

Matthew J Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI fall symposia*, volume 45, pp. 141, 2015.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

Haitao Jiang, Lin Ge, Hengrui Cai, and Rui Song. Pabu: Progress-aware belief update for efficient llm agents. *arXiv preprint arXiv:2602.09138*, 2026.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Po-Chen Kuo, Han Hou, Will Dabney, and Edgar Walker. Predictive coding enhances meta-rl to achieve interpretable bayes-optimal belief representation under partial observability. *Advances in Neural Information Processing Systems*, 38:28024–28055, 2026.

Mikko Lauri, David Hsu, and Joni Pajarinen. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40, 2022.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Junfeng Liao, Qizhou Wang, Jianing Zhu, Bo Du, Rui Yan, and Xiuying Chen. Belief memory: Agent memory under partial observability. *arXiv preprint arXiv:2605.05583*, 2026.

Aly Lidayan, Jakob Bjorner, Satvik Golechha, Kartik Goyal, and Alane Suhr. Abbel: Learning natural-language belief states for memory-efficient interaction. *arXiv preprint arXiv:2512.20111*, 2025.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Yuze Liu, Chaochao Lu, and Chao Yang. Shadow: Dynamic-aware credit assignment against long-horizon tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23935–23944, 2026.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. *arXiv preprint arXiv:2110.05038*, 2021.

Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *Ijcai*, volume 3, pp. 1025–1032, 2003.

Paul J Pritz and Kin K Leung. Belief states for cooperative multi-agent reinforcement learning under partial observability. *arXiv preprint arXiv:2504.08417*, 2025.

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

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

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

Joykirat Singh, Zaid Khan, Archiki Prasad, Justin Chih-Yao Chen, Akshay Nambi, Hyunji Lee, Elias Stengel-Eskin, and Mohit Bansal. Agent-brace: Decoupling beliefs from actions in long-horizon tasks via verbalized state uncertainty. *arXiv preprint arXiv:2605.11436*, 2026.

Valentin Six, Frederik Panse, Mathis Fajeau, Lancelot Da Costa, Mridul Sharma, Alfonso Amayuelas, Tim Z Xiao, David Hyland, Philipp Hennig, and Bernhard Schölkopf. Learning pomdp world models from observations with language-model priors. *arXiv preprint arXiv:2605.13740*, 2026.

Lingfeng Sun, Devesh K Jha, Chiori Hori, Siddarth Jain, Radu Corcodel, Xinghao Zhu, Masayoshi Tomizuka, and Diego Romeres. Interactive planning using large language models for partially observable robotic tasks. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14054–14061. IEEE, 2024.

Andrew Szot, Max Schwarzer, Harsh Agrawal, Bogdan Mazoure, Rin Metcalf, Walter Talbott, Natalie Mackraz, R Devon Hjelm, and Alexander T Toshev. Large language models as generalizable policies for embodied tasks. In *The Twelfth International Conference on Learning Representations*, 2023.

Weihao Tan, Wentao Zhang, Shanqi Liu, Longtao Zheng, Xinrun Wang, and Bo An. True knowledge comes from practice: Aligning large language models with embodied environments via reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 54281–54328, 2024.

Wenjie Tang, Minne Li, Sijie Huang, Liquan Xiao, and Yuan Zhou. Rewarding beliefs, not actions: Consistency-guided credit assignment for long-horizon agents. *arXiv preprint arXiv:2605.20061*, 2026a.

Wenjing Tang, Xinyu He, Yongxi Huang, Yunxiao Xiao, Cewu Lu, and Panpan Cai. Tru-pomdp: Task planning under uncertainty via tree of hypotheses and open-ended pomdps. *Advances in Neural Information Processing Systems*, 38:55367–55396, 2026b.

Andrew Wang, Andrew C Li, Toryn Q Klassen, Rodrigo Toro Icarte, and Sheila A McIlraith. Learning belief representations for partially observable deep rl. In *International Conference on Machine Learning*, pp. 35970–35988. PMLR, 2023a.

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

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Yiping Lu, Zhengyuan Yang, Lijuan Wang, et al. Vagen: Reinforcing world model reasoning for multi-turn vlm agents. *Advances in Neural Information Processing Systems*, 38:172871–172933, 2026a.

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

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Milestone-guided policy learning for long-horizon language agents. *arXiv preprint arXiv:2605.06078*, 2026b.

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

Chulabhaya Wijesundara, Andrea Baisero, Gregory David Castanon, Alan S Carlin, Robert Platt, and Christopher Amato. Leveraging fully-observable solutions for improved partially-observable offline reinforcement learning. *Transactions on Machine Learning Research*, 2025.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025.

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

Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

Yuexiang Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Shengbang Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, et al. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *Advances in neural information processing systems*, 37:110935–110971, 2024.

Guijia Zhang, Yuxun Chen, Yuheng Qi, and Harry Yang. Do gui agents believe their eyes? diagnosing state-belief reliance on pixels versus structure. *arXiv preprint arXiv:2607.04334*, 2026.

Linfeng Zhao, Willie McClinton, Aidan Curtis, Nishanth Kumar, Tom Silver, Leslie Pack Kaelbling, and Lawson LS Wong. Seeing is believing: Belief-space planning with foundation models as uncertainty estimators. *arXiv preprint arXiv:2504.03245*, 2025.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024a.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*, 2024b.

Deyu Zou, Yongqiang Chen, Fan Feng, Mufei Li, Pan Li, Yu Gong, and James Cheng. On information self-locking in reinforcement learning for active reasoning of llm agents. *arXiv preprint arXiv:2603.12109*, 2026a.

Deyu Zou, Yongqiang Chen, Jianxiang Wang, Garry Yang, Mufei Li, Qing Da, James Cheng, Pan Li, and Yu Gong. Reducing belief deviation in reinforcement learning for active reasoning of llm agents. In *International Conference on Learning Representations*, volume 2026, pp. 21511–21545, 2026b.

A ALGORITHMS

Figure 7 provides a condensed overview: its abbreviated pseudocode highlights the main training phases, while the module diagram summarizes information flow. Algorithms 1 and 2 specify the complete update procedures.

```
# Stage 1: Collect parallel rollouts
# BSM initialized by offline supervision
for each rollout step:
    x_t = prompt(e_prev, a_prev, o_t, goal)
    e_t = bsm.decode(x_t)
    e_ref = ref_update(e_ref_prev,
            a_prev, o_t, env_state)
    q_t = predicate_f1(e_t, e_ref)
    context = (e_t, o_t, goal)
    action = actor.select(context, valid_actions)
    next_obs, reward, done = env.step(action)
    ppo_buffer.store(context, action, q_t,
            log_probs, values, reward, done)
    bsm_buffer.store(x_t, e_ref)
    e_prev, a_prev, o_t = e_t, action, next_obs
    e_ref_prev = e_ref
    if done: reset_episode()
# Stage 2: Prepare learning signals
returns, A = token_gae(ppo_buffer)
w = clip(w_min + lambda_q * q, w_min, w_max)
w = stop_gradient(w / mean(w))
A_actor = w * standardize(A)
```

```
# Stage 3: Policy and state adaptation
for each PPO minibatch:
    critic.update(returns)  # unweighted targets
    actor.ppo_update(A_actor)

# After all PPO updates
bsm.teacher_forcing_update(bsm_buffer)
bsm_buffer.clear()
```

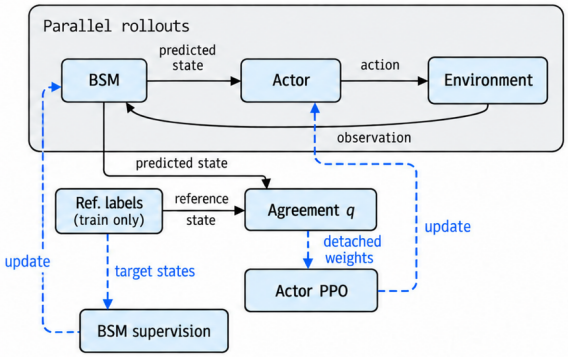


Figure 7: Condensed overview of BeliefRL training. The pseudocode (left and upper right) sketches the main phases, and the module diagram (lower right) shows information flow.

Algorithm 1 BeliefRL BSM Initialization and Training-Time Adaptation

Input: ψ, offline corpus $\mathcal{D}$ / training rollout pairs $\{(x_t, \bar{e}_t)\}$; configured settings
Offline initialization (once)
for each configured offline update **do**
  Sample $(a_{t-1}, o_t, \bar{e}_{t-1}, \bar{e}_t, g)$ from $\mathcal{D}$
  Select e^{in}_{t-1} from $\bar{e}_{t-1}$ or the previous BSM prediction using scheduled sampling
  $x_t \leftarrow \mathrm{Prompt}(e^{in}_{t-1}, a_{t-1}, o_t, g)$
  $\psi \leftarrow \mathrm{SafeStep}(\psi, \nabla_\psi \,\mathrm{mean}_t\, \ell^{BSM}_t)$ [Eq. 6]
end for
Agent-training adaptation (after PPO; if joint adaptation is enabled):
for each configured epoch and valid minibatch $\mathcal{B}_{BSM}$ **do**
  $x_t \leftarrow x^{rollout}_t$; targets $\leftarrow \bar{e}_t$; mask prompt/padding tokens
  $G_\psi \leftarrow 0$; exclude empty targets; preserve targets when truncating x_t
  for each CE microbatch $\mathcal{C} \subseteq \mathcal{B}_{BSM}$ **do**
    $G_\psi \leftarrow G_\psi + \nabla_\psi \left[|\mathcal{B}_{BSM}|^{-1} \sum_{t \in \mathcal{C}} \ell^{BSM}_t \right]$ [Eq. 6]
  end for
  $G_\psi \leftarrow \mathrm{ClipGrad}(G_\psi)$
  if $\mathrm{finite}(G_\psi) \wedge \mathrm{finite}(\|G_\psi\|)$ **then**
    $\psi \leftarrow \mathrm{OptimizerStep}(\psi, G_\psi)$
  end if
end for
Clear training rollout pairs; θ, ϕ unchanged; **Output:** ψ

Algorithm 2 BeliefRL Reliability-Aware Alternating Training

Input: pretrained ψ, θ, ϕ, environment, reference-state mapping, g; configured rollout/PPO settings
Input: $\gamma, \lambda_{GAE}, \epsilon, c_{ent}, c_V, \lambda_q, w_{min}, w_{max}$; joint-adaptation flag
$(o_0, x_0, e_0, \bar{e}_0) \leftarrow \mathrm{EpisodeInit}(g)$
for each training iteration **do**
  $\theta_{old} \leftarrow \theta$; PPO/belief buffers $\leftarrow \emptyset$
  for each rollout decision t **do**
    $\mathcal{X}_t \leftarrow (e_t, o_t, g)$; $q_t \leftarrow \mathrm{Agreement}(e_t, \bar{e}_t)$ [Eq. 8]
    $a_t \sim \pi_\theta(\cdot \mid \mathcal{X}_t, \mathcal{A}_t)$ (word-normalized action scores)
    Record $\{\tau_{t,i}, \log P_{\theta_{old}}(\tau_{t,i} \mid \mathcal{X}_t, \tau_{t,<i}), V_t^i\}_{i=1}^{L_t}$
    $(o_{t+1}, r_t^{env}, d_t) \leftarrow \mathrm{EnvStep}(a_t)$
    $x_{t+1} \leftarrow \mathrm{Prompt}(e_t, a_t, o_{t+1}, g)$; $e_{t+1} \leftarrow \mathrm{Decode}(P_\psi, x_{t+1})$
    $\bar{e}_{t+1} \leftarrow \mathrm{ReferenceStateUpdate}(\bar{e}_t, a_t, o_{t+1}; s_{t+1})$
    Append $(\mathcal{X}_t, a_t, \tau_{t,i}, \log P_{\theta_{old}}, V_t^i, r_t^{env}, d_t, q_t)$ to PPO buffer
    Append $(x_t, \bar{e}_t)$ to belief buffer
    if $d_t = 1$ **then**
      $(o_0, x_0, e_0, \bar{e}_0) \leftarrow \mathrm{EpisodeInit}(g)$; continue rollout
    else
      $(o_t, x_t, e_t, \bar{e}_t) \leftarrow (o_{t+1}, x_{t+1}, e_{t+1}, \bar{e}_{t+1})$
    end if
  end for
  Bootstrap boundary values; $(A_t^i, R_t^i) \leftarrow \mathrm{TokenGAE}(r_t^{env}, V_t^i, d_t; \gamma, \lambda_{GAE})$ [Eq. 28]
  for each PPO epoch and minibatch $\mathcal{B}$ **do**
    $\widehat{A}_t^i \leftarrow \mathrm{Standardize}(A_t^i)$; $\widetilde{A}_t^i \leftarrow \mathrm{sg}(\bar{w}_t)\widehat{A}_t^i$ [Eq. 10]
    $\phi \leftarrow \mathrm{SafeStep}(\phi, \nabla_\phi \mathcal{L}_V)$ [Eq. 31]
    $\mathcal{J}_{actor} \leftarrow K^{-1} \sum_k \mathcal{J}_{actor}^{\mathcal{C}_k}$ [Eq. 29]
    $\widehat{D}_{KL} \leftarrow K^{-1} \sum_k \langle (\rho - 1) - \log \rho \rangle_{\mathcal{C}_k}$
    if $\widehat{D}_{KL} \leq 0.02$ **then**
      $\theta \leftarrow \mathrm{SafeStep}(\theta, -\nabla_\theta \mathcal{J}_{actor})$
    end if
  end for
  $\psi \leftarrow$ Algorithm 1, agent-training adaptation phase (ψ, belief buffer)
end for
Output: trained parameters (ψ, θ, ϕ)
Deployment: $x_t \to e_t \to \mathcal{X}_t \to a_t$; parameters fixed, no reference beliefs

B Additional Experimental Results

B.1 Final Performance, Effect Sizes, and Uncertainty

Table 5 summarizes the final-window environment-only return and AUC statistics corresponding to the main experiments. These statistics provide a compact numerical view of the trends reported in the main text.

For completeness, the following analysis reports descriptive effect sizes and resampling-based uncertainty for the mean-gap comparisons. We resample the five run-level final-window returns for BeliefRL and, independently, the five runs for each baseline method, then recompute the gap between the BeliefRL mean and the equal-weighted mean of the baseline means. The resulting intervals summarize variability across the available runs. This resampling operates on run-level statistics rather than additional environment interactions or training steps, and the intervals are pointwise across the four tasks.

For each task, let $x_1, \ldots, x_5$ be the BeliefRL final-window returns and let $y_i = M^{-1} \sum_{m=1}^{M} y_{i,m}$ be the equal-weighted mean across the $M = 5$ baseline methods for seed i. Table 6 reports the all-baseline mean gap and aggregate-comparator effect sizes. The statistics answer different questions:

$$\Delta = \bar{x} - \bar{y}, \qquad \delta = \frac{1}{25} \sum_{i=1}^{5} \sum_{j=1}^{5} \mathrm{sgn}(x_i - y_j), \qquad D = \frac{\Delta}{\sqrt{(v_x + v_y)/2}}, \tag{16}$$

Table 5: Final-window environment return and training AUC averaged over five runs per method. Std reports seed-level standard deviation.

Task	Algorithm	Final	Std	AUC	AUC Std
Entertainment	ARCHER	0.069	0.044	0.055	0.027
Entertainment	GLAM	0.000	0.000	0.012	0.004
Entertainment	NTPO	0.117	0.095	0.061	0.038
Entertainment	POAD	0.175	0.179	0.109	0.128
Entertainment	TWOSOME	0.102	0.070	0.051	0.032
Entertainment	BeliefRL	**0.447**	0.064	**0.235**	0.062
Food Preparation	ARCHER	0.228	0.138	0.195	0.049
Food Preparation	GLAM	0.268	0.293	0.155	0.153
Food Preparation	NTPO	0.513	0.147	0.292	0.077
Food Preparation	POAD	0.547	0.026	0.351	0.068
Food Preparation	TWOSOME	0.554	0.104	0.305	0.091
Food Preparation	BeliefRL	**0.631**	0.040	**0.435**	0.046
Tomato Salad	ARCHER	0.096	0.150	0.093	0.079
Tomato Salad	GLAM	0.152	0.004	0.152	0.003
Tomato Salad	NTPO	0.247	0.187	0.173	0.067
Tomato Salad	POAD	0.490	0.179	0.296	0.076
Tomato Salad	TWOSOME	0.371	0.267	0.229	0.119
Tomato Salad	BeliefRL	**0.703**	0.102	**0.382**	0.093
Tomato Lettuce Salad	ARCHER	0.241	0.031	0.216	0.016
Tomato Lettuce Salad	GLAM	0.339	0.126	0.264	0.059
Tomato Lettuce Salad	NTPO	0.271	0.004	0.257	0.006
Tomato Lettuce Salad	POAD	0.303	0.096	0.262	0.032
Tomato Lettuce Salad	TWOSOME	0.257	0.012	0.229	0.009
Tomato Lettuce Salad	BeliefRL	**0.402**	0.179	**0.320**	0.085

where $v_x = 5^{-1}\sum_i (x_i - \bar{x})^2$ and $v_y = 5^{-1}\sum_i (y_i - \bar{y})^2$. Cliff's δ compares BeliefRL run-level scores with the per-seed all-baseline aggregate scores through the 25 run-level pairings and is reported as a descriptive rank comparison. The standardized gap D rescales the same mean difference by the observed run dispersion. It uses population variances and complements, rather than replaces, the mean-gap intervals.

Table 6: Mean-gap CIs and descriptive effect sizes for the all-baseline comparisons in Table 1. The baseline comparator is the equal-weighted per-seed mean of the five baseline methods. D standardizes the mean gap using pooled population variance.

Task	Δ mean [95% CI]	Cliff's δ	Std. gap D
Entertainment	+0.355 [0.287, 0.423]	1.000	7.253
Food Preparation	+0.209 [0.137, 0.282]	1.000	3.443
Tomato Salad	+0.432 [0.310, 0.538]	0.888	2.427
Tomato Lettuce Salad	+0.120 [-0.033, 0.280]	0.040	0.933

The intervals are computed from the run-level resampling procedure described above and provide a pointwise summary of seed variation across the four tasks. All four mean-gap point estimates favor BeliefRL. Cliff's δ is 1.000 for Entertainment and Food Preparation, 0.888 for Tomato Salad, and 0.040 for Tomato Lettuce Salad, while the mean gaps are positive on every task. Together, these results provide a complementary run-level characterization of BeliefRL's observed advantage over the aggregate baseline comparator.

B.2 Cost Overhead

We measure deployment inference on Tomato Lettuce Salad using a retained jointly adapted checkpoint and a shared LLaMA-3.1-8B backbone on NVIDIA A100 GPUs. The actor-only control uses the same actor weights, goal, and observation but removes the BSM adapter and state prompt, serving as an inference-cost control rather than a separately trained baseline. With CUDA synchroniza-

tion, BSM timing includes prompt construction, decoding, and parsing, while policy timing includes prompt construction and action scoring; total timing excludes environment transitions, model loading, warmup actions, and logging. Each configuration contributes 320 measured decisions. Table 7 shows that BeliefRL takes 1.387 seconds per decision versus 0.055 seconds for the actor-only control, adds only 0.74 GiB (about 4.8%) peak memory. The table quantifies this latency–memory trade-off for the tested deployment setting; acceptable latency depends on the application.

Table 7: Deployment cost on Tomato Lettuce Salad with 10 episodes per configuration (seeds 101–110). Times are mean seconds per decision; P95 is the 95th percentile of total time; Peak is maximum PyTorch-allocated GPU memory, including the resident shared model and adapters but excluding CUDA-context and allocator-reserved overhead.

Configuration	BSM (s)	Policy (s)	Total (s)	P95 (s)	Peak (GiB)
Actor only	–	0.055	0.055	0.058	15.54
BeliefRL	1.308	0.079	1.387	1.429	16.28

B.3 Zero-Shot Object-Swap Generalization

We additionally test whether the Food Preparation policy generalizes when only the goal object changes while the native task interface is kept fixed. Starting from the original `pancake` task, we replace the target object with `milk`, `cereal`, `candybar`, `bananas`, `lime`, or `pear`, while preserving the same room graph, source-target structure, and action semantics. Evaluation is zero-shot, using one checkpoint per trained method and greedy action selection, without further training or few-shot adaptation. Each method is evaluated for 100 episodes on each swapped-object variant. We report mean episodic return as the primary metric.

Table 8 shows that BeliefRL obtains the highest mean episodic return among the evaluated checkpoints under this semantic perturbation. Its zero-shot OOD mean reaches 0.44, above TWOSOME's 0.43 and POAD's 0.25; POAD remains competitive in `cereal` and `lime`. These results indicate transfer to unseen goal objects, with the room graph, source-target structure, and action semantics held fixed.

Table 8: Zero-shot object-swap generalization on Food Preparation. Cells report mean episodic return over 100 episodes per method and variant; OOD Mean averages the six swapped-object tasks.

Method	Milk	Cereal	Candybar	Bananas	Lime	Pear	OOD Mean
LLaMA-3.1-8B	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NTPO	0.16	0.16	0.16	0.00	0.16	0.16	0.14
ARCHER	0.16	0.16	0.16	0.16	0.16	0.16	0.16
GLAM	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TWOSOME	0.44	0.66	0.30	0.43	0.30	0.43	0.43
POAD	0.16	0.44	0.16	0.16	0.42	0.16	0.25
BeliefRL	0.16	0.58	0.45	0.45	0.58	0.45	0.44

B.4 Evaluation on NLP Benchmarks

Figure 8 visualizes the benchmark-wise change relative to the untuned LLaMA-3.1-8B model while still annotating the absolute scores. The subject-level MMLU breakdown is then reported in Table 9. BeliefRL preserves broad language ability while improving task performance: its average score is 0.714, compared with 0.710 for the untuned model, and it obtains the highest average among the compared fine-tuned methods. Thus, the gains from RL fine-tuning do not require giving up the pretrained model's general language competence.

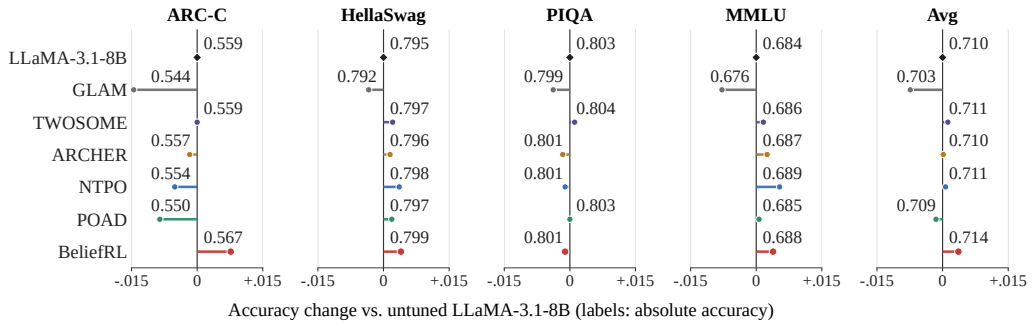


Figure 8: Benchmark-wise language retention after RL fine-tuning on Food Preparation. Each panel centers the untuned LLaMA-3.1-8B model at zero; marker labels show absolute accuracy.

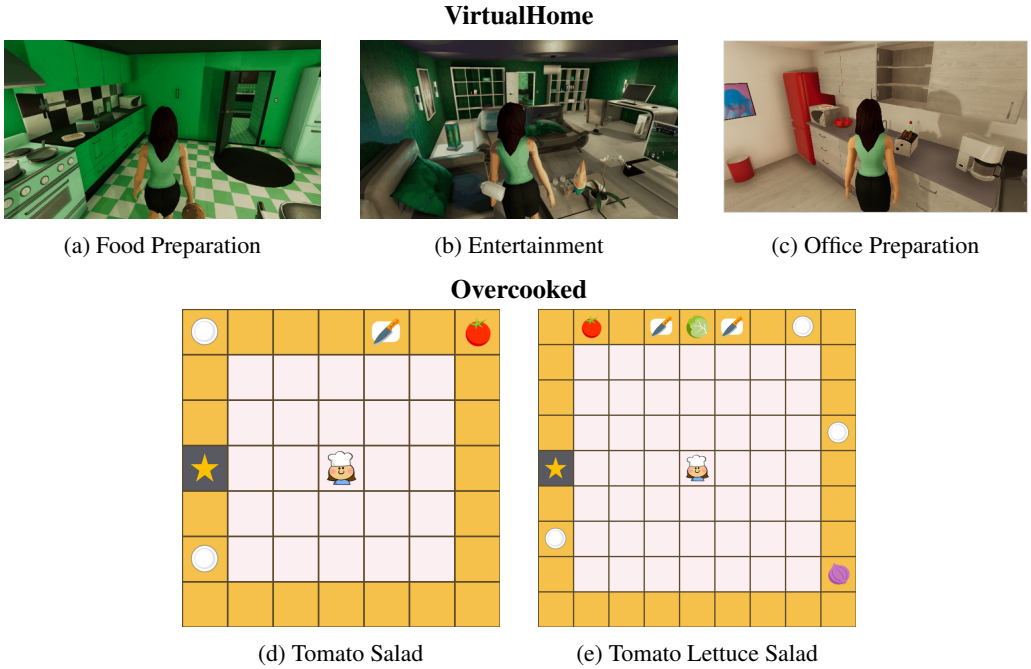
Table 9: Subject-level MMLU accuracy after RL fine-tuning on Food Preparation; higher is better.

Task	LLaMA-3.1-8B	GLAM	TWOSOME	ARCHER	NTPO	POAD	BeliefRL
Abstract Algebra	0.370	0.330	0.370	0.370	0.360	0.330	0.340
Anatomy	0.689	0.681	0.689	0.704	0.704	0.696	0.681
Astronomy	0.750	0.730	0.750	0.763	0.757	0.750	0.763
Business Ethics	0.690	0.730	0.690	0.700	0.700	0.690	0.700
Clinical Knowledge	0.777	0.781	0.770	0.781	0.781	0.777	0.774
College Biology	0.806	0.806	0.819	0.812	0.819	0.819	0.812
College Chemistry	0.490	0.450	0.500	0.460	0.510	0.450	0.490
College Computer Science	0.560	0.500	0.570	0.560	0.570	0.570	0.550
College Mathematics	0.350	0.310	0.330	0.320	0.370	0.350	0.350
College Medicine	0.688	0.688	0.682	0.688	0.676	0.688	0.694
College Physics	0.461	0.490	0.490	0.441	0.461	0.451	0.451
Computer Security	0.780	0.770	0.770	0.780	0.790	0.780	0.780
Conceptual Physics	0.596	0.596	0.596	0.587	0.596	0.579	0.579
Econometrics	0.509	0.535	0.518	0.535	0.535	0.518	0.526
Electrical Engineering	0.655	0.676	0.655	0.641	0.662	0.648	0.669
Elementary Mathematics	0.495	0.468	0.508	0.495	0.500	0.505	0.513
Formal Logic	0.500	0.484	0.500	0.516	0.508	0.492	0.524
Global Facts	0.370	0.390	0.350	0.370	0.360	0.400	0.370
High School Biology	0.810	0.810	0.813	0.803	0.800	0.813	0.816
High School Chemistry	0.635	0.611	0.640	0.626	0.650	0.616	0.635
High School Computer Science	0.750	0.730	0.760	0.760	0.760	0.760	0.750
High School European History	0.758	0.758	0.758	0.764	0.752	0.758	0.770
High School Geography	0.783	0.798	0.793	0.793	0.788	0.793	0.788
High School Government And Politics	0.891	0.860	0.891	0.891	0.886	0.881	0.881
High School Macroeconomics	0.695	0.674	0.705	0.697	0.708	0.703	0.695
High School Mathematics	0.415	0.407	0.415	0.415	0.426	0.407	0.415
High School Microeconomics	0.807	0.790	0.807	0.790	0.794	0.803	0.815
High School Physics	0.444	0.430	0.457	0.470	0.444	0.404	0.450
High School Psychology	0.873	0.868	0.873	0.870	0.875	0.870	0.873
High School Statistics	0.565	0.556	0.560	0.551	0.579	0.551	0.546
High School US History	0.848	0.838	0.833	0.838	0.843	0.838	0.843
High School World History	0.857	0.861	0.861	0.857	0.869	0.865	0.865
Human Aging	0.713	0.695	0.713	0.700	0.717	0.704	0.717
Human Sexuality	0.786	0.802	0.779	0.786	0.786	0.794	0.794
International Law	0.802	0.810	0.802	0.810	0.810	0.810	0.818
Jurisprudence	0.778	0.796	0.778	0.787	0.787	0.787	0.778
Logical Fallacies	0.810	0.798	0.791	0.798	0.798	0.810	0.822
Machine Learning	0.509	0.482	0.491	0.536	0.527	0.562	0.509
Management	0.825	0.816	0.825	0.816	0.825	0.825	0.835
Marketing	0.893	0.876	0.893	0.897	0.893	0.897	0.893
Medical Genetics	0.760	0.750	0.780	0.760	0.770	0.770	0.780
Miscellaneous	0.843	0.842	0.843	0.845	0.847	0.839	0.843
Moral Disputes	0.740	0.737	0.743	0.746	0.746	0.740	0.743

Task	LLaMA-3.1-8B	GLAM	TWOSOME	ARCHER	NTPO	POAD	BeliefRL
Moral Scenarios	0.598	0.552	0.601	0.628	0.629	0.617	0.629
Nutrition	0.768	0.758	0.771	0.768	0.765	0.771	0.771
Philosophy	0.723	0.740	0.733	0.730	0.733	0.723	0.730
Prehistory	0.756	0.765	0.747	0.756	0.756	0.753	0.738
Professional Accounting	0.543	0.550	0.543	0.546	0.546	0.543	0.546
Professional Law	0.505	0.503	0.508	0.503	0.510	0.504	0.504
Professional Medicine	0.787	0.765	0.779	0.794	0.790	0.790	0.790
Professional Psychology	0.721	0.724	0.721	0.730	0.724	0.725	0.729
Public Relations	0.691	0.682	0.691	0.700	0.682	0.682	0.691
Security Studies	0.755	0.755	0.755	0.747	0.743	0.747	0.751
Sociology	0.841	0.846	0.851	0.846	0.851	0.846	0.841
US Foreign Policy	0.890	0.860	0.890	0.900	0.880	0.880	0.890
Virology	0.506	0.494	0.518	0.506	0.506	0.524	0.512
World Religions	0.819	0.813	0.830	0.830	0.825	0.819	0.836

C Details of Language Environments

VirtualHome and Overcooked are established testbeds for sequential decision making and language-agent reinforcement learning, used in both TWOSOME and POAD (Tan et al., 2024; Wen et al., 2024). VirtualHome grounds household activities in executable action sequences and object interactions (Puig et al., 2018), while Overcooked provides structured cooking tasks with ordered preparation and delivery requirements. Figure 9 groups our five tasks by environment. We adapt these settings to partial observability and sparse rewards, emphasizing hidden-state tracking and long-horizon credit assignment; Office Preparation adds a harder household search task. The specific observation rules, layouts, and task variants used here are detailed below.



(a) Food Preparation (b) Entertainment (c) Office Preparation

(d) Tomato Salad (e) Tomato Lettuce Salad

Figure 9: Representative scenes and layouts for the five tasks: VirtualHome tasks in the top row and Overcooked tasks in the bottom row.

C.1 VirtualHome

All VirtualHome experiments use the multi-task graph wrapper on a four-room apartment consisting of the kitchen, living room, bathroom, and bedroom. The action space consists of deterministic navigation, manipulation, and interaction macro-actions. Episodes last for at most 50 steps. The

observation is strictly room-local: at each step the agent only perceives objects in its current room together with their interaction-relevant states, such as whether a visible container is open or whether a visible interactive object is already on.

Food Preparation. The task contains a single transport goal: place the `pancake` into the `microwave`. The agent starts in the living room, the microwave is fixed in the living room, and the pancake location is randomized at reset over {kitchen, living room, bathroom, bedroom, fridge}. The episode succeeds once the pancake is inside the microwave and otherwise terminates at the 50-step horizon.

Entertainment. Entertainment is the more compositional VirtualHome benchmark. It combines one transport goal and two persistent interaction goals: place the `milk` on the `coffeetable`, switch the `tv` on, and sit on the `sofa`. The agent again starts in the living room, where the coffee table, TV, and sofa are fixed, while the milk location is randomized over the four rooms at reset. Relative to Food Preparation, the challenge here is not only locating a hidden object but also coordinating three predicates that must hold simultaneously at the end of the episode. Reward is issued only when all three subgoals are satisfied together.

Office Preparation (Additional Experiment). Office Preparation is an additional VirtualHome evaluation environment used for the harder search-and-composition experiment in the main text. It is instantiated on a separate apartment layout from the two main VirtualHome benchmarks, while preserving the same room-local observation protocol and 50-step horizon. The task requires the agent to find a hidden `cellphone`, place it on the `desk` in the living room, and then sit on the `chair`. The delivery surface and final interaction target are fixed, but the cellphone location is randomized at reset over two bedroom nightstands and two kitchen cabinets.

Relative to Food Preparation and Entertainment, this environment increases the number of plausible hidden candidates while also extending the dependency chain after discovery. The agent must not only rule out multiple containers under partial observability, but also preserve task progress across transport and post-placement interaction. This tests belief tracking across a larger search space and longer action sequences.

C.2 Overcooked

The two Overcooked tasks use POMDP adaptations with different kitchen layouts. Tomato Salad uses a smaller 7×7 fixed layout, whereas Tomato Lettuce Salad uses a larger 9×9 layout with partial ingredient randomization. In both tasks, the agent receives an egocentric radius-2 observation (a 5×5 visible square), and episodes last at most 32 steps. Rewards are sparse: the agent obtains positive reward only when the requested dish is successfully served.

Tomato Salad. Tomato Salad is the control benchmark in the main text. It uses a single-ingredient recipe and a fixed layout whose geometry is matched to the observation radius. Concretely, the agent starts at $[3, 3]$, the tomato is fixed at $[0, 6]$, the two bowls are fixed at $[5, 0]$ and $[0, 0]$, and the delivery counter is fixed at $[3, 0]$. The intended solution is to reveal the tomato, move it to a cutting board, chop it, place the prepared tomato into a bowl, and then serve.

Tomato Lettuce Salad. Tomato Lettuce Salad is the harder Overcooked benchmark and the primary POMDP stress test in this domain. The recipe requires both `lettuce` and `tomato`, while an `onion` is present as a distractor ingredient. The base layout is balanced and radius-matched: the agent starts at $[4, 4]$, the onion remains fixed at $[7, 8]$, and the bowls are fixed at $[0, 7]$, $[6, 0]$, and $[3, 8]$. At each reset, the lettuce and tomato are partially randomized over a small set of boundary positions, introducing mild layout uncertainty without changing the overall topology. The agent discovers and prepares both required ingredients before placing them in a bowl and serving, requiring longer-horizon belief tracking and discrimination between required and distractor ingredients.

C.3 Trajectory Examples and Belief-State Visualizations

Figure 10 presents two constructed execution traces from Food Preparation. Part 1 shows search and target discovery; part 2 shows retrieval, transport, and completion. The examples trace search, target discovery, retrieval, transport, and completion while visualizing belief-state updates along each path. One path includes an unnecessary room detour after the target is no longer retained explicitly. Observation excerpts use the environment text-rendering templates for the illustrated states.

Food Preparation — 1. Search and discovery

Goal: put the pancake in the microwave.
Step 0: living room; empty hands; microwave closed; pancake not visible.
Hidden layout: pancake inside kitchen fridge.
Legend: green = resolved state; amber = unknown; red = possible detour.



Illustrative scene.

BeliefRL

SEARCH ⟶ DISCOVERY

01 Walk to bedroom.

OBS. ... You are in the bedroom. ... You notice nothing useful here.

```
INSIDE(pancake, bedroom): false
```

Negative evidence is retained after leaving.

02 Walk to bathroom.

OBS. ... You are in the bathroom. ... You notice nothing useful here.

```
INSIDE(pancake, bathroom): false
```

03 Walk to kitchen.

OBS. ... You are in the kitchen. ... The fridge is closed.

```
INSIDE(pancake, kitchen): unknown
```

Decision point: an unchecked container keeps the kitchen unresolved.

04 Walk to fridge.

OBS. ... The fridge is close to you. The fridge is closed.

05 Open fridge.

OBS. ... The fridge is open. You see the pancake inside the fridge.

```
INSIDE(pancake, kitchen): true
```

Milestone: target discovered
Hands: empty | Fridge: open
Next: retrieve the observed pancake.

Ordinary method

SEARCH ⟶ DETOUR ⟶ DISCOVERY

01 Walk to bedroom.

OBS. ... You are in the bedroom. ... You notice nothing useful here.

02 Walk to bathroom.

OBS. ... You are in the bathroom. ... You notice nothing useful here.

03 Walk to kitchen.

OBS. ... You are in the kitchen. ... The fridge is closed.

```
No explicit belief-state summary supplied.
```

04 Walk back to bedroom.

OBS. ... You are in the bedroom. ... You notice nothing useful here.

05 Walk back to bathroom.

OBS. ... You are in the bathroom. ... You notice nothing useful here.

```
REPEATED SEARCH: no new target evidence.
```

06 Return to kitchen.

OBS. ... You are in the kitchen. ... The fridge is closed.

07 Walk to fridge.

OBS. ... The fridge is close to you. The fridge is closed.

08 Open fridge.

OBS. ... The fridge is open. You see the pancake inside the fridge.

Milestone: target discovered after a detour
Hands: empty | Fridge: open
Next: retrieve the same pancake.

Figure 10: Complete execution examples, part 1: search and target discovery. Numbers denote individual environment actions. Ellipses shorten schematic observation descriptions only; no actions are omitted. Colored predicates are illustrative belief-state excerpts, not logged predictions.

Food Preparation — 2. Retrieval and completion

Continuation: both agents are beside the open kitchen fridge and can see the pancake.
BeliefRL: after action 05. **Ordinary method:** after action 08.
Remaining goal: retrieve pancake → carry to living room → place in microwave.

BeliefRL

RETRIEVAL ⟶ TRANSPORT ⟶ COMPLETION

06 Grab pancake.

OBS. ... Currently, you have grabbed the pancake in hand. ...

```
INSIDE(pancake, kitchen): true
```

The held object is localized to the current room.

07 Walk to living room.

OBS. ... You are in the livingroom. Currently, you have grabbed the pancake in hand. ...

```
INSIDE(pancake, livingroom): true
```

State revision: transport changes the object's room; the earlier kitchen location is no longer current.

08 Walk to microwave.

OBS. ... The microwave is close to you. The microwave is closed.

09 Open microwave.

OBS. ... The microwave is open. ...

10 Put pancake in microwave.

OBS. ... You see the pancake inside the microwave.

```
GOAL SATISFIED -- terminal state
```

Complete path: 10 actions
Search information persists across room changes and is revised when the object moves.

Ordinary method

RETRIEVAL ⟶ TRANSPORT ⟶ COMPLETION

09 Grab pancake.

OBS. ... Currently, you have grabbed the pancake in hand. ...

Current observation directly confirms possession.

10 Walk to living room.

OBS. ... You are in the livingroom. Currently, you have grabbed the pancake in hand. ...

The transport succeeds, but the microwave target is not retained explicitly.

11 Walk to bedroom.

OBS. ... You are in the bedroom. ...

```
DETOUR: target lost; an incorrect room is selected.
```

12 Walk to living room.

OBS. ... You are in the livingroom. Currently, you have grabbed the pancake in hand. ...

13 Walk to microwave.

OBS. ... The microwave is close to you. The microwave is closed.

14 Open microwave.

OBS. ... The microwave is open. ...

15 Put pancake in microwave.

OBS. ... You see the pancake inside the microwave.

```
GOAL SATISFIED -- terminal state
```

Complete path: 15 actions
The extra room detour lengthens the successful path without adding a manipulation requirement.

Figure 10: Execution examples, part 2: retrieval, transport, and terminal success. Both branches complete the task; the right branch includes an unnecessary room detour after transport. These constructed paths are not measured policy rollouts or claims of verified optimality.

D Data Construction

D.1 Belief Schema and Reference Annotation

We use a common predicate-based annotation interface, with the predicate set instantiated from task-relevant entities, relations, and persistent interaction states. Each predicate takes one of three values: `True` when established by available evidence, `False` when ruled out, and `Unknown` when neither conclusion is justified. An unobserved predicate is not treated as false merely because it is currently invisible; updates follow the task's observation and transition semantics.

Reference beliefs are constructed recursively during training:

$$\bar{e}_t = \mathcal{G}_{\text{ref}}(\bar{e}_{t-1}, a_{t-1}, o_t; s_t), \tag{17}$$

where s_t is the simulator state available to the annotation procedure. The mapping uses observation constraints to distinguish observable evidence from unresolved information, rather than copying the full simulator state into the belief. These annotations provide BSM supervision; at deployment, beliefs are predicted by the trained BSM without access to simulator states or reference beliefs.

D.2 Offline Trajectory Collection

The training corpus combines trajectories collected by random exploration and MCTS-guided search. Random exploration provides broad state coverage, while MCTS contributes successful trajectories containing informative belief transitions.

Random trajectories are generated by uniformly sampling legal actions,

$$a_t \sim \text{Uniform}(\mathcal{A}_{\text{legal}}(o_t)), \tag{18}$$

where $\mathcal{A}_{\text{legal}}(o_t)$ denotes the set of legal actions under the current observation. This strategy exposes the BSM to both progress-making and non-progressing transitions, improving the robustness of belief updates.

Child nodes are selected according to a PUCT-style scoring function,

$$\begin{aligned}\text{Score}(n,a) = Q(n,a) + c_{\text{puct}} P(n,a) \frac{\sqrt{N(n)}}{1+N(n,a)} \\ + 0.25E(n,a) + 0.11\mathbf{1}[\Delta_{\text{subtask}}(n,a) > 0] - \Omega_{\text{sel}}(n,a),\end{aligned} \tag{19}$$

where Q and P denote the action value and prior probability, respectively, $E(n,a)$ measures action effectiveness, Δ_{subtask} represents subtask progress, and $\Omega_{\text{sel}}(n,a)$ is the selection penalty defined below. Rollout values are estimated by

$$V_{\text{search}}(z) = r(z) + \lambda_{\text{prog}} \cdot \text{progress}(z) - \min(\Omega_{\text{roll}}, 0.5), \tag{20}$$

and propagated through the search tree with discount factor γ, where z includes the simulator state and task configuration available to the offline collector.

Collection settings and search signals. Table 10 reports the collection budgets: each environment family uses $B_{\text{rand}} = 115$ random and $B_{\text{mcts}} = 160$ MCTS episodes in total, rather than per task instance. These task-specific, simulator-informed signals guide offline collection only; they are neither additional policy-training rewards nor deployment inputs.

Table 10: Offline collection settings by environment family. Simulation counts are per decision.

Setting	VirtualHome	Overcooked
Random / MCTS episodes	115 / 160	115 / 160
Maximum episode steps	38	32
Simulations per decision	48–72	80
Rollout depth	10–11	12
Forced-expansion horizon	2	2
c_{puct} / λ_{prog} / γ	1.5 / 0.8 / 0.95	1.5 / 0.8 / 0.95

Progress is the mean normalized stage over subtasks, $\text{progress}(z) = |J|^{-1} \sum_{j \in J} s_j(z)/m_j$ (zero if J is empty). VirtualHome stages track source discovery, retrieval, transport, placement, and switch/sit completion; Overcooked stages track ingredient discovery, handling, chopping, bowl assembly, and delivery. The active unfinished subtask is selected by lock level (held objects first), movable-task priority, progress ratio, and task order; Δ_{subtask} is its normalized progress change.

Action effectiveness combines aggregate progress change (1.2), active-subtask progress change (1.6), and goal-relation change (1.0), with holding, visibility, and proximity terms (VH:

$0.25, 0.15, 0.15$; Overcooked: $0.3, 0.15, 0.2$). It also includes task-specific continuation bonuses, an unchanged-state penalty, and penalties for invalid ingredient handling or unsuccessful serving. Continuation bonuses prioritize finishing a held or partly completed subtask. In Overcooked, invalid handling includes placing an ingredient into a bowl before required chopping; unsuccessful serving is penalized separately.

Selection uses $\Omega_{\text{sel}} = 0.08 I_{\text{self}} + 0.03 I_{\text{ancestor}} + 0.8 I_{\text{unchanged}}$, where the indicators denote a self-loop, a state key already on the ancestor path, and an unchanged simulator state. Separately, Ω_{roll} accumulates 0.08 per unchanged transition and 0.03 per revisit within the last four rollout states, capped at 0.5 in the leaf value; a terminal tree leaf uses its self-loop penalty. State keys represent simulator state and task configuration.

The final VirtualHome training corpus is obtained by merging random and MCTS trajectories,

$$\mathcal{D}_{\text{VH}} = \text{Canon}\left(\mathcal{D}_{\text{rand}} \cup \mathcal{D}_{\text{mcts}}\right), \tag{21}$$

where $\text{Canon}(\cdot)$ denotes schema normalization into the unified transition format used for BSM training.

D.3 Training Examples and BSM Interface

The resulting training corpus consists of transition-wise supervision

$$\mathcal{D} = \{(a_{t-1}, o_t, \bar{e}_{t-1}, \bar{e}_t, g)\}_{t=1}^{T}. \tag{22}$$

Here, g denotes the task goal associated with each trajectory. Algorithm 3 summarizes the complete data construction pipeline.

Algorithm 3 Simulator-Grounded Belief Data Construction

Input: environment $\mathcal{E}$, reference-state mapping $\mathcal{G}_{\text{ref}}$, random budget B_{rand}, MCTS budget B_{mcts}
Initialize dataset $\mathcal{D} \leftarrow \emptyset$
for each task instance $\xi \in \mathcal{E}$ sampled within the collection budgets **do**
  Obtain task goal g from ξ
  Collect random trajectories within the total episode budget B_{rand}
  Collect MCTS trajectories within the total episode budget B_{mcts}
  for each transition $(\bar{e}_{t-1}, a_{t-1}, o_t)$ **do**
    Compute $\bar{e}_t \leftarrow \mathcal{G}_{\text{ref}}(\bar{e}_{t-1}, a_{t-1}, o_t; s_t)$
    Store $(a_{t-1}, o_t, \bar{e}_{t-1}, \bar{e}_t, g)$ into $\mathcal{D}$
  end for
end for
Return: belief supervision corpus $\mathcal{D}$

The BSM is trained to predict the next belief state from the previous belief state, previous action, current observation, and task goal. The training input is constructed as

$$x_t = \Pi(e_{t-1}, a_{t-1}, o_t, g), \tag{23}$$

where $\Pi(\cdot)$ denotes the prompt template that serializes the structured inputs into a textual sequence. The supervision target is the reference belief state $\bar{e}_t$ produced by the annotation procedure.

The prompt explicitly specifies the allowed predicate schema and requires the model to update beliefs only when supported by new observations while preserving unchanged predicates. Supervision targets the symbolic belief directly, without auxiliary reasoning traces.

Scheduled Sampling. To mitigate exposure bias during recursive inference, we employ a curriculum-based scheduled sampling strategy during BSM training. Let G denote the total number of training steps, W the teacher-forced warmup length, and ϵ_{init} and ϵ_{final} the initial and final teacher-forcing probabilities. At training step g, the probability of teacher forcing is

$$\epsilon(g) = \begin{cases} \epsilon_{\text{init}}, & g \le W, \\ \max\left(\epsilon_{\text{final}}, \epsilon_{\text{init}} - \dfrac{g-W}{G-W}(\epsilon_{\text{init}} - \epsilon_{\text{final}})\right), & g > W. \end{cases} \tag{24}$$

For non-initial steps, the input belief state e_{t-1}^{in} stochastically alternates between the ground-truth previous state e_{t-1}^{GT} with probability $\epsilon(g)$ and the model's canonicalized previous prediction $\hat{e}_{t-1}$ with probability $1-\epsilon(g)$. This makes the BSM training interface closer to deployment, where the model must condition on its own previous symbolic state.

E Training Details

E.1 Model Configuration

All experiments use Meta-Llama-3.1-8B-Instruct as the backbone. The policy and BSM are trained with LoRA adapters using different configurations. The complete model configuration is summarized in Table 11. At inference time, the policy conditions on the task description, current observation, and belief state:

$$u_t = \Gamma(g, o_t, e_t), \tag{25}$$

Table 11: Model configurations used in all experiments.

Component	**Configuration**
Backbone	Meta-Llama-3.1-8B-Instruct
Policy LoRA	$r = 8$, $\alpha = 16$, dropout 0 Target modules: {q_proj, v_proj}
BSM LoRA	$r = 16$, $\alpha = 32$, dropout 0.05 Target modules: {q_proj, v_proj, k_proj, o_proj}

E.2 BSM Training Hyperparameters

The supervised BSM stage uses the final training recipe with scheduled sampling. Table 12 summarizes the settings that determine optimization and recursive training behavior.

Table 12: Supervised BSM training settings used in the final recipe.

Item	**Setting**
Optimizer	AdamW
Learning rate	5×10^{-5}
Gradient accumulation	8 steps
Sequence length	1536 tokens
Learning-rate warmup	200 optimizer steps
Teacher-forced warmup	first 25% of training steps
Scheduled sampling	$\epsilon : 1.0 \to 0.4$

E.3 RL Hyperparameters

BeliefRL uses the same token-level PPO optimization framework in both VirtualHome and Overcooked. Table 13 reports the representative task configurations; entries shared by both environment families are repeated for direct comparison.

Table 13: Representative BeliefRL RL hyperparameters for VirtualHome and Overcooked.

Item	VirtualHome	Overcooked
Learning rate	1×10^{-6}	5×10^{-7}
Critic learning rate	5×10^{-5}	1×10^{-5}
PPO epochs	5	5
Mini-batches	2	2
Entropy coefficient	1×10^{-4}	1×10^{-2}
Discount γ	0.95	0.99
Rollout horizon	32	32
Training budget	8,000 env. steps	8,000 env. steps
Belief learning rate	1×10^{-7}	1×10^{-7}
Reliability coefficient λ_q	0.01	0.05
PPO clip coefficient	0.2	0.2
Value loss coefficient	0.5	0.5
GAE λ	0.95	0.95
Gradient clipping	global norm 0.5	global norm 0.5

F Policy Optimization Details

This appendix specifies action selection, token-level advantage estimation, and implementation conventions for the policy optimization described in the main text.

F.1 Action Selection and Advantage Estimation

The actor and critic use $\mathcal{X}_t = (e_t, o_t, g)$. An action string is $a_t = (\tau_{t,1}, \ldots, \tau_{t,L_t})$, and V_t^i denotes its rollout value before token i.

Token-Level Action Factorization. The actor assigns each action string the language-model score

$$P_\theta(a_t \mid \mathcal{X}_t) = \prod_{i=1}^{L_t} P_\theta(\tau_{t,i} \mid \mathcal{X}_t, \tau_{t,<i}). \tag{26}$$

At rollout, we sample an action from the candidate-action distribution over available strings $\mathcal{A}_t$, using word-count normalization $W(a)$:

$$\pi_\theta(a \mid \mathcal{X}_t, \mathcal{A}_t) = \frac{\exp\big(W(a)^{-1} \log P_\theta(a \mid \mathcal{X}_t)\big)}{\sum_{a' \in \mathcal{A}_t} \exp(W(a')^{-1} \log P_\theta(a' \mid \mathcal{X}_t))}. \tag{27}$$

Following POAD, PPO updates use conditional LM probabilities for the selected action tokens, yielding an action-decomposed token-level surrogate. The available-action filtering and word-count normalization are used for rollout action selection. The token-level PPO objectives are given in Appendix F.2.

Token-Level Advantage Estimation. Token-level GAE (Wen et al., 2024; Schulman et al., 2015) discounts only environment transitions. With termination flag d_t and trace parameter λ_{GAE},

$$\begin{aligned} \delta_t^i &= \begin{cases} V_t^{i+1} - V_t^i, & i < L_t, \\ r_t^{env} + \gamma(1-d_t)V_{t+1}^1 - V_t^{L_t}, & i = L_t, \end{cases} \\ A_t^i &= \begin{cases} \delta_t^i + \lambda_{GAE} A_t^{i+1}, & i < L_t, \\ \delta_t^{L_t} + \gamma\lambda_{GAE}(1-d_t)A_{t+1}^1, & i = L_t, \end{cases} \qquad R_t^i = A_t^i + V_t^i. \end{aligned} \tag{28}$$

Values come from the rollout. At its boundary, bootstrap from the next observation and set the future trace to zero; termination resets the trace.

F.2 Token-Level PPO Objectives

For completeness, we give the actor and critic objectives used in the token-level PPO updates. Following the action decomposition in POAD, let ρ_t^i be the ratio between the current and stored conditional LM probabilities of the selected token and let $\langle\cdot\rangle_{\mathcal{C}}$ denote the mean over nonpadding tokens in accumulation chunk $\mathcal{C}$. The actor objective is

$$J_{\text{actor}}^{\mathcal{C}} = \left\langle \min\left(\rho_t^i \widetilde{A}_t^i, \text{clip}(\rho_t^i, 1-\epsilon, 1+\epsilon)\widetilde{A}_t^i\right) + c_{\text{ent}} H_t^i \right\rangle_{\mathcal{C}}. \tag{29}$$

The critic uses unweighted rollout targets $R_t^i = A_t^i + V_t^i$ and clipped values

$$V_{\text{clip},t}^i = V_{\text{old},t}^i + \text{clip}(V_\phi - V_{\text{old},t}^i, -\epsilon, \epsilon). \tag{30}$$

Its value objective is

$$\mathcal{L}_V = c_V \left\langle \max\left\{\ell(R_t^i - V_\phi), \ell(R_t^i - V_{\text{clip},t}^i)\right\}\right\rangle_{\mathcal{B}}, \tag{31}$$

where ℓ is the configured squared-error or Huber loss. Reliability weights modify the actor advantages only; the critic targets and loss are unweighted.

G Theoretical Analysis and Proofs

This appendix develops two complementary analyses summarized in Section 3.3: belief-deviation-induced stagnation and advantage inversion, together with a reliability-weighted bound on the gradient discrepancy due to a specified belief-induced error component. The results characterize how belief-state adaptation and reliability-aware actor updates shape learning under partial observability.

G.1 Objects, Assumptions, and Relation to the Implementation

The policy-visible state e_t and reference state $\bar{e}_t$ induce predicate sets E_t, T_t and the agreement score q_t in Equation 8. Let

$$w(q) = \text{clip}(w_{min} + \lambda_q q, w_{min}, w_{max}), \qquad 0 < w_{min} \le w_{max} < \infty, \quad \lambda_q \ge 0. \tag{32}$$

This is a positive, bounded, nondecreasing function. All weights are detached in the actor update. We make the following assumptions for Theorem 1.

1. **Fixed update distribution.** Samples are drawn from a fixed rollout distribution and policy parameters are held fixed for the analysis. Expectations include the choice of a valid action token. Its parent decision supplies q.
2. **Integrability.** With $\zeta = \nabla_\theta \log P_\theta(\tau_i \mid X, \tau_{<i})$, the products $\zeta A^\star$ and $\zeta \varepsilon_A$ are integrable. This ensures the expectations in Equation (14) exist; score integrability alone would not suffice for arbitrary advantages.
3. **Advantage decomposition.** $\widehat{A} = A^\star + \varepsilon_A$, where $A^\star$ is a same-scale reference advantage sharing non-belief estimation effects with $\widehat{A}$, and ε_A is the additional belief-induced advantage-error component. Residual value-estimation, finite-horizon, and normalization effects common to both advantages are retained in the reference rather than included in ε_A.
4. **Conditional calibration of the belief-induced component.** For almost every q, $\mathbb{E}[\|\zeta\varepsilon_A\| \mid q] \le C(1-q)$ with finite $C \ge 0$.

Under this additional component-wise calibration condition, higher agreement receives a tighter conditional upper bound $C(1-q)$ on the score-weighted belief-induced advantage error. This condition is not implied by predicate agreement alone: matching the predicate interface need not recover the full latent state. On the event $q = 1$, the assumption requires $\zeta\varepsilon_A = 0$ almost surely. Residual value-estimation, finite-horizon, and normalization effects shared with the reference remain outside this bound and may persist at $q = 1$.

Scope of the population analysis. Equation (14) analyzes the population first-order effect of reliability weighting on the corresponding token-surrogate actor gradient. In implementation, this weighting is combined with sampled-decision normalization, token masking, chunk aggregation,

and PPO clipping (Schulman et al., 2017), while the value head uses environment-reward targets. The result compares observed and reference gradients under the same weighting, rollout distribution, and score ζ. The reference gradient is not an error-free oracle: the bound controls only the specified belief-induced component and does not bound discrepancies between the token surrogate and the true policy gradient.

G.2 Belief Deviation and One-Step Stagnation under Abstract BSM Dynamics

This analysis adapts the belief-potential and update-error framework of Zou et al. (2026b) to characterize the conditional one-step drift of belief deviation under the assumptions below. For analysis only, associate the textual belief state with a latent belief $b \in \Delta(\mathcal{S})$. Fix a latent target $s^\star$ within the analytical slice and restrict attention to

$$\Delta^+_{s^\star}(\mathcal{S}) = \{b \in \Delta(\mathcal{S}) : b(s^\star) > 0\}.$$

Define

$$\Psi(b) = -\log b(s^\star). \tag{33}$$

This potential provides an analytical lens for quantifying belief deviation. For changing physical worlds, the analysis is applied on an appropriate fixed-target slice or together with an explicit account of target changes.

Let $\mathcal{B}^\star$ be an ideal Bayesian update and $\mathcal{B}_\psi$ the abstract update associated with the BSM. We assume that the relevant outputs of both updates remain in $\Delta^+_{s^\star}(\mathcal{S})$ and that the following expectations are finite. For a fixed policy and the same action–observation distribution, define

$$I(b) = \mathbb{E}_{a,o}[\Psi(b) - \Psi(\mathcal{B}^\star(b,a,o))], \tag{34}$$

$$c(b) = \mathbb{E}_{a,o}[\Psi(\mathcal{B}_\psi(b,a,o)) - \Psi(\mathcal{B}^\star(b,a,o))], \tag{35}$$

$$P(b) = \mathbb{E}_{a,o}[\Psi(b) - \Psi(\mathcal{B}_\psi(b,a,o))] = I(b) - c(b). \tag{36}$$

Assume $I(b) \le \overline{I} < \infty$ and, for $\Psi(b) \ge U_0$, $c(b) \ge m\Psi(b) - c_0$, where $m > 0$ and $c_0, U_0 \ge 0$. Define

$$U = \max\{U_0, (\overline{I} + c_0)/m\}, \qquad \mathcal{R}_{BTR} = \{b : \Psi(b) \ge U\}. \tag{37}$$

Proposition 1 (Conditional one-step stagnation) *For every $b \in \mathcal{R}_{BTR}$ satisfying the stated assumptions, the abstract BSM update obeys*

$$P(b) \le 0, \qquad \mathbb{E}_{a,o}\left[\Psi(\mathcal{B}_\psi(b,a,o))\right] \ge \Psi(b). \tag{38}$$

Proof. By Equation 36, $P(b) \le \overline{I} - m\Psi(b) + c_0 \le 0$. By the definition of $P(b)$, this is equivalent to the second inequality.

This conditional drift characterization identifies a regime in which the abstract BSM update ceases to reduce belief deviation, highlighting the role of belief-state adaptation during agent training in maintaining reliable state estimates along policy-induced histories. The negative-TD analysis below complements this view by characterizing how persistent adverse tails can distort long-horizon advantage estimates.

G.3 Stagnating Tails and Advantage Inversion

Following the negative-tail credit-assignment analysis of Zou et al. (2026b), consider a finite environment-step trajectory of length T, before advantage standardization, with sparse or outcome-only rewards. Let V_k be its environment-step value estimate, including the terminal mask in the bootstrap term. Using GAE (Schulman et al., 2015), write

$$\delta_k = r_k + \gamma(1-d_k)V_{k+1} - V_k, \qquad \widehat{A}^{env}_t = \sum_{j=0}^{T-t-1} \alpha^j \delta_{t+j}, \quad \alpha = \gamma\lambda_{GAE} \in [0,1]. \tag{39}$$

This environment-step analysis is distinct from the token recursion in Equation 28. Fix t_S and assume $\mathbb{E}[\delta_k] \le u_k$ for $k < t_S$ and $\mathbb{E}[\delta_k] \le -r$ for $k \ge t_S$, with $r > 0$. If a random first-entry time is used, the expectations and drift bounds are conditional on that time.

Proposition 2 (Prefix advantage inversion) *Every $t < t_S$ satisfies*

$$\mathbb{E}[\widehat{A}_t^{env}] \le \underbrace{\sum_{j=0}^{t_S-t-1} \alpha^j u_{t+j}}_{S_{pre}(t)} - r \underbrace{\sum_{j=t_S-t}^{T-t-1} \alpha^j}_{S_{tail}(t)}, \qquad t < t_S. \tag{40}$$

If $rS_{tail}(t) > S_{pre}(t)$, the expected estimate is negative, even if the prefix action is informative under an independently specified criterion.

Proof. Split the finite GAE sum at $j = t_S - t$. Its coefficients are nonnegative. Applying the respective prefix and tail bounds term by term gives

$$\mathbb{E}[\widehat{A}_t^{env}] = \sum_{j=0}^{t_S-t-1} \alpha^j \mathbb{E}[\delta_{t+j}] + \sum_{j=t_S-t}^{T-t-1} \alpha^j \mathbb{E}[\delta_{t+j}] \le S_{pre}(t) - rS_{tail}(t). \tag{41}$$

The strict condition shows that a persistent negative tail can reverse the expected sign of the prefix advantage estimate.

Two-region weighting interpretation. If two idealized groups have constant scores $q_{tail} \le q_{pre}$, their raw update weights satisfy $w(q_{tail})/w(q_{pre}) \le 1$. BeliefRL applies this positive weight to each decision's complete standardized advantage, so reliability weighting complements rather than replaces the training-time BSM adaptation mechanism: the latter improves visited-state estimates, while the former emphasizes policy updates supported by more reliable estimates. The conditional bound below addresses the specified belief-induced gradient-error component.

G.4 Proof of the Reliability-Weighted Belief-Induced Gradient Bound

Proof of Theorem 1. Subtracting the two gradients and using the advantage decomposition yields

$$g_w^{obs} - g_w^{ref} = \frac{\mathbb{E}[w(q)\zeta\varepsilon_A]}{\mathbb{E}[w(q)]}. \tag{42}$$

The denominator is positive. The triangle inequality, conditional expectation, and calibration give

$$\|g_w^{obs} - g_w^{ref}\| \le \frac{\mathbb{E}[w(q)\|\zeta\varepsilon_A\|]}{\mathbb{E}[w(q)]} \tag{43}$$

$$= \frac{\mathbb{E}[w(q)\mathbb{E}[\|\zeta\varepsilon_A\| \mid q]]}{\mathbb{E}[w(q)]} \le C\frac{\mathbb{E}[w(q)(1-q)]}{\mathbb{E}[w(q)]}. \tag{44}$$

For independent identically distributed copies Q, Q' of q,

$$2\,\mathrm{Cov}(w(Q), 1-Q) = \mathbb{E}[(w(Q) - w(Q'))((1-Q) - (1-Q'))] \le 0, \tag{45}$$

since w is nondecreasing while $1 - Q$ is nonincreasing. Hence $\mathbb{E}[w(q)(1-q)] \le \mathbb{E}[w(q)]\mathbb{E}[1-q]$, proving Equation (15). Thus, monotone reliability weights yield a no-larger calibrated upper bound on the specified belief-induced population gradient discrepancy than constant weights. This compares calibrated upper bounds, not necessarily the actual errors under the two weighting schemes.

Corollary 1 (Affine-weight bound reduction) *If clipping is inactive on the support of q, so that $w(q) = w_{min} + \lambda_q q$, then*

$$\frac{\mathbb{E}[w(q)(1-q)]}{\mathbb{E}[w(q)]} = \mathbb{E}[1-q] - \frac{\lambda_q \,\mathrm{Var}(q)}{w_{min} + \lambda_q \mathbb{E}[q]}. \tag{46}$$

Consequently, the reduction in the stated upper bound is $C\lambda_q \,\mathrm{Var}(q)/(w_{min} + \lambda_q \mathbb{E}[q])$.

Proof. Expand the numerator as $w_{min}\mathbb{E}[1-q] + \lambda_q \mathbb{E}[q(1-q)]$ and use $\mathbb{E}[q(1-q)] = \mathbb{E}[q]\mathbb{E}[1-q] - \mathrm{Var}(q)$. Division by $\mathbb{E}[w] = w_{min} + \lambda_q \mathbb{E}[q]$ gives Equation 46.

Together, these analyses characterize the roles of the two BeliefRL mechanisms: cross-entropy supervision during agent training adapts the state estimator on visited histories, while detached reliability weights yield a no-larger calibrated upper bound on the specified belief-induced population gradient discrepancy under the stated calibration assumption.