Bayesian Residual Policy Optimization:
Scalable Bayesian Reinforcement Learning with Clairvoyant Experts

Gilwoo Lee, Brian Hou, Sanjiban Choudhury, Siddhartha S. Srinivasa
Paul G. Allen School of Computer Science & Engineering
University of Washington
{gilwoo,bhou,sanjibac,siddh}@cs.uw.edu

***Abstract*—Informed and robust decision making in the face of uncertainty is critical for robots that perform physical tasks alongside people. We formulate this as Bayesian Reinforcement Learning over latent Markov Decision Processes (MDPs). While Bayes-optimality is theoretically the gold standard, existing algorithms do not scale well to continuous state and action spaces. Our proposal builds on the following insight: in the absence of uncertainty, each latent MDP is easier to solve. We first obtain an ensemble of experts, one for each latent MDP, and fuse their advice to compute a baseline policy. Next, we train a Bayesian residual policy to improve upon the ensemble's recommendation and learn to reduce uncertainty. Our algorithm, Bayesian Residual Policy Optimization (BRPO), imports the scalability of policy gradient methods and task-specific expert skills. BRPO significantly improves the ensemble of experts and drastically outperforms existing adaptive RL methods.**

I. Introduction

Robots that are deployed in the real world must continue to operate in the face of model uncertainty. For example, an autonomous vehicle must safely navigate around pedestrians navigating to latent goals (Figure 1). A robot arm must reason about occluded objects when reaching into a cluttered shelf. This class of problems can be framed as Bayesian reinforcement learning (BRL) where the agent maintains a belief over latent Markov Decision Processes (MDPs). Under model uncertainty, agents do not know which latent MDP they are interacting with, preventing them from acting optimally with respect to that MDP. At best, they can be *Bayes optimal*, or optimal with respect to their current uncertainty over latent MDPs.

In this work, we focus on continuous control tasks with model uncertainty. Specifically, we aim to solve problems in which *the latent model is independently resampled at the beginning of each episode*. In the autonomous vehicle example, the pedestrians' goals are unknown and must be rediscovered whenever the agent sees a new set of pedestrians. In these settings, the agent must actively reduce uncertainty or select actions that are robust to it.

A Bayesian RL problem can be viewed as solving a large continuous belief MDP, which is computationally infeasible to solve directly [13]. These tasks, especially with continuous action spaces, are challenging even for state-of-the-art belief-space planning and robust RL algorithms. Continuous action spaces are challenging for existing POMDP algorithms, which are either limited to discrete action spaces [24] or rely on online planning and samples from the continuous action space [16]. Latent MDPs can be complex and may require vastly different policies to achieve high reward; robust RL methods [37, 46] are often unable to produce that multi-modality.

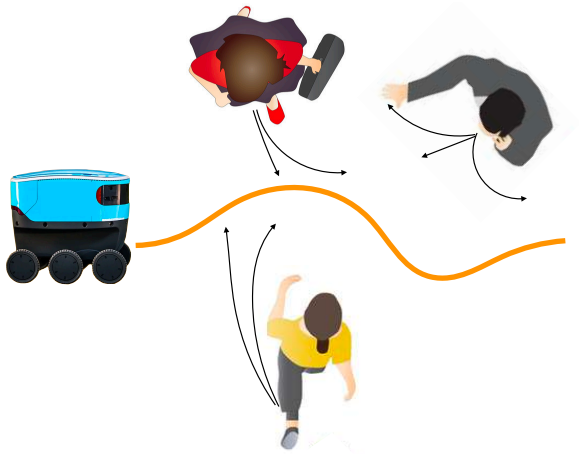


Fig. 1: An autonomous vehicle approaches an area with unpredictable pedestrians, each noisily moving toward their own latent goal. Given uncertainty on their goals, the agent must take the Bayes-optimal action to quickly drive past them without collisions.

We build upon a simple yet recurring observation [8, 22, 29]: while solving the belief MDP is hard, solving individual latent MDPs, in the absence of uncertainty, is much more tractable. If the path for each pedestrian is known, the autonomous vehicle can invoke a motion planner that avoids collision. We can think of these solutions as *clairvoyant experts*, *i.e.*, experts that think they know the latent MDP and offer advice accordingly. Combining advice from the clairvoyant experts can be effective, but such an ensemble policy can be suboptimal in the original belief MDP. Since experts are individually confident about which MDP the agent faces, the ensemble never prioritizes uncertainty reduction or robust actions, which can be critical in solving the original problem with inherent model uncertainty.

Our algorithm, Bayesian Residual Policy Optimization (BRPO), computes a *residual* policy to augment a given ensemble of clairvoyant experts (Figure 2). This is computed via policy optimization in a residual belief MDP, induced by the ensemble policy's actions on the original belief MDP. Because the ensemble is near-optimal when the entropy is low, BRPO can focus on learning how to safely collapse uncertainty in regions of higher entropy. It can also start with much higher

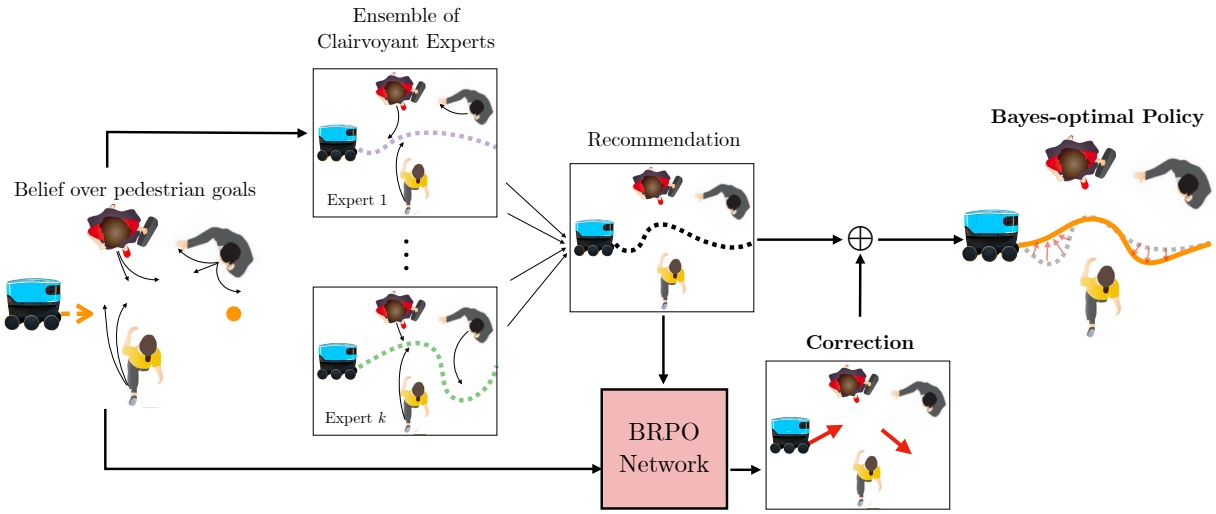


Fig. 2: An overview of Bayesian Residual Policy Optimization. (a) Pedestrian goals are latent and tracked as a belief distribution. (b) Experts propose their solutions for a scenario, which are combined into a mixture of experts. (c) Residual policy takes in the belief and ensemble of experts' proposal and returns a correction to the proposal. (d) The combined BRPO and ensemble policy is (locally) Bayes-optimal.

performance than when starting from scratch, which we prove in Section IV and empirically validate in Section V.

Our key contribution is the following:

- We propose BRPO, a scalable Bayesian RL algorithm for problems with model uncertainty.
- We prove that it monotonically improves upon the expert ensemble, converging to a locally Bayes-optimal policy.
- We experimentally demonstrate that BRPO outperforms both the ensemble and existing adaptive RL algorithms.

II. Related Work

POMDP methods. Bayesian reinforcement learning formalizes RL where one has a prior distribution over possible MDPs [13, 41]. However, the Bayes-optimal policy, which is the best one can do under uncertainty, is intractable to solve for and approximation is necessary [20]. One way is to approximate the value function, as done in SARSOP [24] and PBVI [35]; however, they cannot deal with continous state actions. Another strategy is to resort to sampling, such as BAMCP [16], POMCP [42], POMCPOW [45]. However, these approaches require a significant amount of online computation.

Online approaches forgo acting Bayes-optimally right from the onset, and instead aim to eventually act optimally. The question then becomes: how do we efficiently gain information about the test time MDP to act optimally? BEB [23] and POMDP-lite [7] introduce an auxiliary reward term to encourage exploration and prove Probably-Approximately-Correct (PAC) optimality. This has inspired work on more general, non-Bayesian curiosity based heuristics for reward gathering [1, 5, 19, 33]. Online exploration is also well studied in the bandit literature, and techniques such as posterior sampling [30] bound the learner's regret. UP-OSI [48] predicts the most likely MDP and maps that to an action. Gimelfarb et al. [14] learns a gating over multiple expert value functions. However, online methods can over-explore to unsafe regimes.

Another alternative is to treat belief MDP problems as a large state space that must be compressed. Peng et al. [34] use Long Short-Term Memory (LSTM) [18] to encode a history of observations to generate an action. Methods like BPO [25] explicitly utilize the belief distribution and compress it to learn a policy. The key difference between BRPO and BPO is that BRPO uses an expert, enabling it to scale to handle complex latent tasks that may require multimodal policies.

Meta-reinforcement Learning. Meta-reinforcement learning (MRL) approaches train sample-efficient learners by exploiting structure common to a distribution of MDPs. For example, MAML [10] trains gradient-based learners while RL2 [9] trains memory-based learners. While meta-supervised learning has well established Bayesian roots [2, 3], it wasn't until recently that meta-reinforcement learning was strongly tied to Bayesian Reinforcement Learning (BRL) [28, 36]. Nevertheless, even non-Bayesian MRL approaches address problems pertinent to BRL. MAESN [17] learns structured noise for exploration. E-MAML [44] adds an explicit exploration bonus to the MAML objective. GMPS [27] exploit availability of MDP experts to partially reduce BRL to IL. Our work is more closely related to Bayesian MRL approaches. MAML-HB [15] casts MAML as hierarchical Bayes and improves posterior estimates. BMAML [47] uses non-parametric variational inference to improve posterior estimates. PLATIPUS [11] learns a parameter distribution instead of a fixed parameter. PEARL [38] learns a data-driven Bayes filter across tasks. In contrast to these approaches, we use experts at test time, learning only to optimally correct them.

Residual Learning. Residual learning has its foundations in boosting [12] where a combination of weak learners, each

learning on the failures of previous, make a strong learner. It also allows for injecting priors in RL, by boosting off of hand-designed policies or models. Prior work has leveraged known but approximate models by learning the residual between the approximate dynamics and the discovered dynamics [31, 32, 4]. There has also been work on learning residual policies over hand-defined ones for solving long horizon [43] and complex control tasks [21]. Similarly, our approach starts with a useful initialization (via experts) and learns to improve via Bayesian reinforcement learning.

III. Preliminaries: Bayesian Reinforcement Learning

We are interested in the performance of an RL agent under model uncertainty, in which *the latent model gets reset at the beginning of each episode*. As discussed in Section I, this problem can be formulated as model-based Bayesian Reinforcement Learning (BRL). Formally, the problem is defined by a tuple $\langle S, \Phi, A, T, R, P_0, \gamma \rangle$, where S is the observable state space of the underlying MDPs, Φ is the latent space, and A is the action space. T and R are the transition and reward functions parameterized by ϕ. The initial distribution over (s, ϕ) is given by $P_0 : S \times \Phi \to \mathbb{R}^+$, and γ is the discount.

Since the latent variable is not observable, Bayesian RL considers the long-term expected reward with respect to the uncertainty over ϕ rather than the true (unknown) value of ϕ. Uncertainty is represented as a *belief distribution* $b \in B$ over latent variables ϕ. The Bayes-optimal action value function is given by the Bellman equation:

$$Q(s, b, a) = R(s, b, a) + \gamma \sum_{s', b'} P(s', b' | s, b, a) \max_{a'} Q(s', b', a') \quad (1)$$

where $R(s, b, a) = \sum_{\phi \in \Phi} b(\phi) R(s, \phi, a)$ and $P(s'|s, b, a) = \sum_{\phi \in \Phi} b(\phi) P(s'|s, \phi, a)$. The posterior update $P(b'|s, b, a)$ is computed recursively, starting from initial belief b_0. $b'(\phi'|s, b, a, s') = \eta \sum_{\phi \in \Phi} b(\phi) T(s, \phi, a, s', \phi')$ where η is the normalizing constant, and the transition function is defined as $T(s, \phi, a, s', \phi') = P(s'|s, \phi, a) P(\phi'|s, \phi, a, s')$.

While some terminology is shared with online RL algorithms (e.g. Posterior Sampling Reinforcement Learning [29]), that setting assumes latent variables are fixed for multiple episodes. We refer the reader to Appendix C for further discussion.

IV. Bayesian Residual Policy Optimization (BRPO)

Bayesian Residual Policy Optimization relies on an ensemble of clairvoyant experts where each expert solves a latent MDP. This is a flexible design parameter with three guidelines. First, the ensemble must be fixed before training begins. This fixes the residual belief MDP, which is necessary for theoretical guarantees (Section IV-C). Next, the ensemble should return its recommendation quickly since it will be queried online at test time. Practically, we have observed that this factor is often more important than the strength of the initial ensemble: even weaker ensembles can provide enough of a head start for residual learning to succeed. Finally, when the belief has collapsed to a single latent MDP, the resulting recommendation must follow the corresponding expert. In general, the ensemble should become more reliable as entropy decreases.

Algorithm 1 Bayesian Residual Policy Optimization

Require: Bayes filter ψ, belief b_0, prior P_0, residual policy π_{r_0}, expert π_e, horizon T, $n_{\text{itr}}, n_{\text{sample}}$

for $i = 1, 2, \cdots, n_{\text{itr}}$ **do**
  for $n = 1, 2, \cdots, n_{\text{sample}}$ **do**
    Sample latent MDP $\mathcal{M}$: $(s_0, \phi_0) \sim P_0$
    $\tau_n \leftarrow \texttt{Simulate}(\pi_{r_{i-1}}, \pi_e, b_0, \psi, \mathcal{M}, T)$
  $\pi_{r_i} \leftarrow \texttt{BatchPolicyOpt}(\pi_{r_{i-1}}, \{\tau_n\}_{n=1}^{n_{\text{sample}}})$
return $\pi_{r_{best}}$

procedure SIMULATE($\pi_r, \pi_e, b_0, \psi, \mathcal{M}, T$)
  for $t = 1, \cdots, T$ **do**
    $a_{e_t} \sim \pi_e(s_t, b_t)$ // Expert recommendation
    $a_{r_t} \sim \pi_r(s_t, b_t, a_{e_t})$ // Residual policy
    $a_t \leftarrow a_{r_t} + a_{e_t}$
    Execute a_t on $\mathcal{M}$, receive r_{t+1}, observe s_{t+1}
    $b_{t+1} \leftarrow \psi(s_t, b_t, a_t, s_{t+1})$ // Belief update
  $\tau \leftarrow (s_0, b_0, a_{r_0}, r_1, s_1, b_1, \cdots, s_T, b_T)$
  return τ

BRPO performs batch policy optimization in the residual belief MDP, producing actions that continuously correct the ensemble recommendations. Intuitively, BRPO enjoys improved data-efficiency because the correction can be small when the ensemble is effective (e.g., when uncertainty is low or when the experts are in agreement). When uncertainty is high, the agent learns to override the ensemble, reducing uncertainty and taking actions robust to model uncertainty.

A. Ensemble of Clairvoyant Experts

For simplicity of exposition, assume the Bayesian RL problem consists of k underlying latent MDPs, $\phi_1, ..., \phi_k$. The ensemble policy maps the state and belief to a distribution over actions $\pi_e : S \times B \to P(A)$. It combines clairvoyant experts $\pi_1, \cdots, \pi_k$, one for each latent variable ϕ_i. Each expert can be computed via single-MDP RL (or optimal control, if transition and reward functions are known by the experts).

There are various strategies to produce an ensemble from a set of experts. The ensemble π_e could be the maximum a posteriori (MAP) expert: $\pi_e = \arg\max_{b(\phi)} \pi_\phi$. This particular ensemble allows BRPO to solve tasks with infinitely many latent MDPs, as long as the MAP expert can be queried online. It can also be a weighted sum of expert actions, which turns out to be the MAP action for Gaussian policies (Appendix D).

While these belief-aware ensembles are easy to attain, they are not Bayes-optimal. Since each clairvoyant expert assumes a perfect model, the ensemble does not take uncertainty reducing actions nor is it robust to model uncertainty. Instead of constructing an ensemble of experts, one could approximately solve the BRL problem with a POMDP solver and perform residual learning on this policy. While the initial policy would

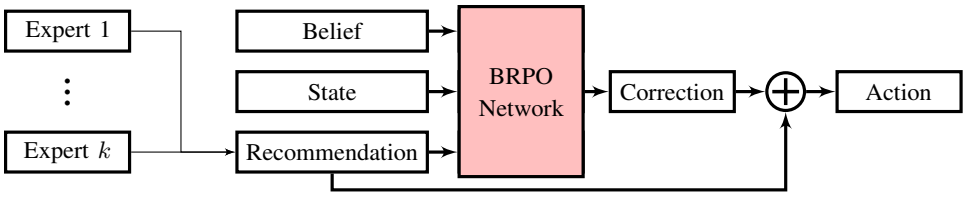


Fig. 3: Bayesian residual policy network architecture.

be much improved, state-of-the-art online POMDP solvers are expensive and would be slow at test time.

B. Bayesian Residual Policy Learning

Our algorithm is summarized in Algorithm 1. In each training iteration, BRPO collects trajectories by simulating the current policy on several MDPs sampled from the prior distribution. At every timestep of the simulation, the ensemble is queried for an action recommendation (Line 9), which is summed with the correction from the residual policy network (Figure 3) and executed (Line 10-12). The Bayes filter updates the posterior after observing the resulting state (Line 13). The collected trajectories are the input to a policy optimization algorithm, which updates the residual policy network. Note that only *residual actions* are collected in the trajectories (Line 14).

The BRPO agent effectively experiences a different MDP: in this new MDP, actions are always shifted by the ensemble recommendation. We formalize this correspondence between the residual and original belief MDPs in the next section, enabling BRPO to inherit the monotonic improvement guarantees from existing policy optimization algorithms.

C. BRPO Inherits Motononic Improvement Guarantees

In this section, we prove that BRPO guarantees monotonic improvement on the expected return of the mixture between the ensemble policy π_e and the initial residual policy π_{r_0}. The following arguments apply to all MDPs, not just belief MDPs. For clarity of exposition, we have omitted the belief from the state and defer all proofs to Appendix A.

First, we observe that π_r operates on its own residual MDP, and that the monotonic guarantee holds in this residual MDP. Then, we show that the monotonic guarantee on the residual MDP can be transferred to the original MDP by showing that the probability of a state-sequence is equal in both MDPs.

Let $\mathcal{M} = \langle S, A, T, R, P_0 \rangle$ be the original MDP. For simplicity, assume that R depends only on states.[1] Every π_e for $\mathcal{M}$ induces a residual MDP $\mathcal{M}_r = \langle S, A_r, T_r, R, P_0 \rangle$ that is equivalent to $\mathcal{M}$ except for the action space and transition function.[2] For every residual action a_r, T_r marginalizes over all expert recommendations $a_e \sim \pi_e(s)$.

$$T_r(s'|s, a_r) = \sum_{a_e} T(s'|s, a_e + a_r)\pi_e(a_e|s) \tag{2}$$

[1]If R is dependent on actions, we can define R_r analogous to (2).

[2] $A_r = A$ as long as the expert action space contains the null action.

Let $\pi_r(a_r|s, a_e)$ be a residual policy. The final policy π executed on $\mathcal{M}$ is a mixture of π_r and π_e, since actions are sampled from both policies and summed.

$$\pi(a|s) = \sum_{a_r} \pi_e(a - a_r|s)\pi_r(a_r|s, a - a_r) \tag{3}$$

Lemma 1. BRPO *monotonically improves the expected return of* π_r *in* $\mathcal{M}_r$, i.e.,

$$J(\pi_{r_{i+1}}) \geq J(\pi_{r_i})$$

with $J(\pi_r) = \mathbb{E}_{\tau \sim (\pi_r, \mathcal{M}_r)}[R(\tau)]$, *where* $\tau \sim (\pi_r, \mathcal{M}_r)$ *indicates that* τ *is a trajectory with actions sampled from* π_r *and executed on* $\mathcal{M}_r$.

Next, we show that the performance of π_r on the residual MDP $\mathcal{M}_r$ is equivalent to the BRPO agent's actual performance on the original MDP $\mathcal{M}$.

Theorem 2. *A residual policy* π_r *executed on* $\mathcal{M}_r$ *has the same expected return as the mixture policy* π *executed on* $\mathcal{M}$.

$$\mathbb{E}_{\tau \sim (\pi, \mathcal{M})}[R(\tau)] = \mathbb{E}_{\tau_r \sim (\pi_r, \mathcal{M}_r)}[R(\tau_r)]$$

We first show that the probability of observing a sequence of states is equal in both MDPs, which immediately leads to this theorem.

Let $\xi = (s_0, s_1, ..., s_{T-1})$ be a sequence of states. Let $\alpha = \{\tau\}$ be the set of all length T trajectories (state-action sequences) in $\mathcal{M}$ with ξ as the states, and $\beta = \{\tau_r\}$ be analogously defined for a set of trajectories in $\mathcal{M}_r$. Note that each state-sequence ξ may have multiple corresponding state-action trajectories $\{\tau\}$, since multiple action-sequences can have the same state-sequence.

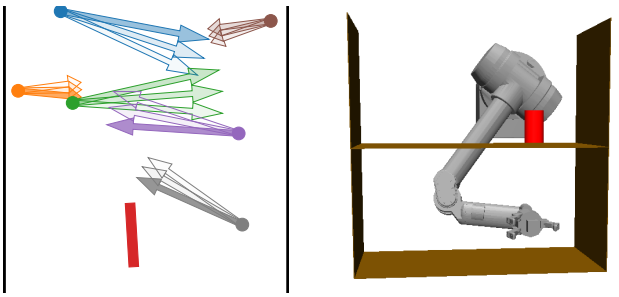
Lemma 3. *The probability of* ξ *is equal when executing* π *on* $\mathcal{M}$ *and* π_r *on* $\mathcal{M}_r$, *i.e.,*

$$\pi(\xi) = \sum_{\tau \in \alpha} \pi(\tau) = \sum_{\tau_r \in \beta} \pi_r(\tau_r) = \pi_r(\xi)$$

Since reward depends only on the states, $R(\tau) = R(\tau_r) = R(\xi)$ for all $\tau \in \alpha, \tau_r \in \beta$. Hence, Lemma 3 immediately implies Theorem 2.

$$\begin{aligned}\mathbb{E}_\tau[R(\tau)] &= \sum_\tau R(\tau)\pi(\tau) = \sum_\xi R(\xi)\pi(\xi) \\ &= \sum_\xi R(\xi)\pi_r(\xi) = \sum_{\tau_r} R(\tau_r)\pi_r(\tau_r) = \mathbb{E}_{\tau_r}[R(\tau_r)]\end{aligned}$$

Finally, we prove our main theorem that Lemma 1, the monotonic improvement guarantee on $\mathcal{M}_r$, transfers to $\mathcal{M}$.



(a) `CrowdNav` (b) `ArmShelf`

Fig. 4: Setup for `CrowdNav` and `ArmShelf`. In `CrowdNav`, the goal for the agent (red) is to go upward without colliding with pedestrians (all other colors). In `ArmShelf`, the goal is to reach for the can under noisy sensing. See Section V-A.

Theorem 4. BRPO *monotonically improves upon the mixture between ensemble policy* π_e *and initial residual policy* π_{r_0}*, eventually converging to a locally optimal policy.*

In summary, BRPO tackles RL problems with model uncertainty by building on an ensemble of clairvoyant experts (queried online) and optimizing a policy on the residual MDP induced by the ensemble (trained offline, queried online). Even suboptimal ensembles often provide a strong baseline, resulting in data-efficient learning and high returns. We empirically evaluate this hypothesis in Section V.

V. Experimental Results

We choose problems that highlight common challenges for robots with model uncertainty:

- Costly sensing is required to infer the latent MDP.
- Uncertainty reduction and robustness are critical.
- Solutions for each latent MDP are significantly different.

In all domains that we consider, BRPO improves on the ensemble's recommendation and significantly outperforms adaptive-RL baselines that do not leverage experts. Qualitative evaluation shows that robust Bayes-optimal behavior naturally emerges from training.

A. Environments

Here we give a brief description of the problem environments. Appendix B contains implementation details.

Crowd Navigation. Inspired by Cai et al. [6], an autonomous agent must quickly navigate past a crowd of people without collisions. Six people cross in front of the agent at fixed speeds, three from each side (Figure 4a). Each person noisily walks toward its latent goal on the other side, which is sampled uniformly from a discrete set of destinations. The agent observes each person's speed and position to estimate the belief distribution for each person's goal. The belief for each person is drawn as a set of vectors in Figure 4a, where length indicates speed and transparency indicates belief probability of each goal. There is a single expert which uses model predictive control: each walker is simulated toward a belief-weighted average goal position, and the expert selects cost-minimizing steering angle and acceleration.

Cartpole. In this environment, the agent's goal is to keep the cartpole upright for as long as possible. The latent parameters are cart mass and pole length (Figure 4), uniformly sampled from $[0.5, 2.0]\text{kg} \times [0.5, 2.0]\text{m}$. There is zero-mean Gaussian noise on the control. The agent's estimator is a very coarse 3×3 discretization of the 2D continuous latent space, and the resulting belief is a categorical distribution over that grid. In this environment, each expert is a Linear-Quadratic Regulator (LQR) for the center of each grid square. The ensemble recommendation used by BRPO is simply the belief-weighted sum of experts, as described in Section IV-A.

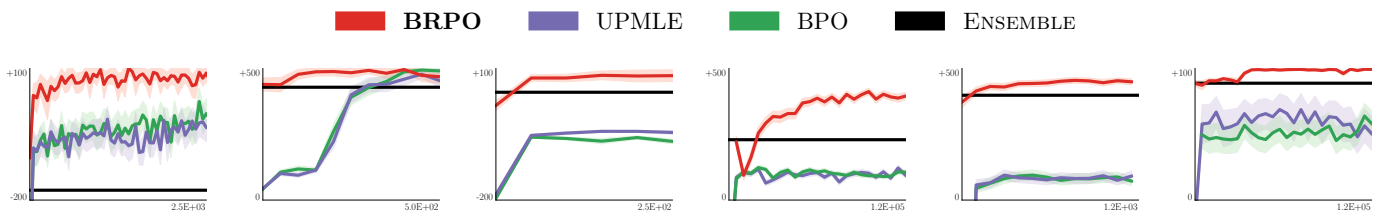
Object Localization. In the **`ArmShelf`** environment, the agent must localize an object without colliding with the environment or object. The continuous latent variable is the object's pose, which is anywhere on either shelf of the pantry (Figure 4b). The agent receives a noisy observation of the object's pose, which is very noisy when the agent does not invoke sensing. Sensing can happen as the agent moves, and is less noisy the closer the end-effector is to the object. The agent uses an Extended Kalman Filter to track the object's pose. The ensemble is the MAP expert, as described in Section IV-A. It takes the MAP object pose and proposes a collision-free movement toward the object.

Latent Goal Mazes. In the **`Maze4`** and **`Maze10`**, the agent must identify which latent goal is active. At the beginning of each episode, the latent goal is set to one of four or ten goals. The agent is rewarded highly for reaching the active goal and severely penalized for reaching an inactive goal. Sensing can happen as the agent moves; the agent receives a noisy measurement of the distance to the goal, with noise proportional to the true distance. Each expert proposes an action (computed via motion planning) that navigates to the corresponding goal. However, they are unaware of the penalty that corresponds to passing through an inactive goal. The ensemble recommends the belief-weighted sum of the experts' suggestions.

Doors. There are four possible doors to the next room of the **`Door4`** environment. At the beginning of each episode, each door is opened or closed with 0.5 probability. To check the doors, the agent can either sense or crash into them (which costs more than sensing). Sensing is permitted while moving, and returns a noisy binary vector for all four doors with exponentially decreasing accuracy proportional to the distance to each door. Crashing returns an accurate indicator of the door it crashed into. Each expert navigates directly through the closest open door, and the ensemble recommends the belief-weighted sum of experts.

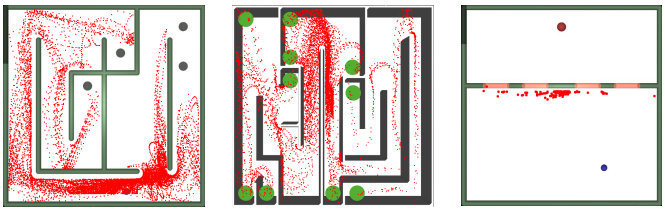
B. BRPO Improves Ensemble, Outperforms Adaptive Methods

We compare BRPO to adaptive RL algorithms that consider the belief over latent states: BPO [25] and UP-MLE, a modification to Yu et al. [48] that augments the state with the



(a) CrowdNav (b) Cartpole (c) ArmShelf (d) Maze4 (e) Maze10 (f) Door4

Fig. 5: Training curves. BRPO (red) dramatically outperforms agents that do not leverage expert knowledge (BPO in purple, UP-MLE in green), and significantly improves the ensemble of experts (black).



(a) Maze4 (b) Maze10 (c) Door4

Fig. 6: Sensing locations. In Maze4 and Maze10, sensing is dense around the starting regions (the bottom row in Maze4 and center in Maze10) and in areas where multiple latent goals are nearby. In Door4, BRPO only senses when close to the doors, where the sensor is most accurate.

most likely estimate from the Bayes filter[3]. Neither approach is able to incorporate experts. We also compare with the ensemble of experts baselines. For experiments which require explicit sensing actions (ArmShelf, Maze4, Maze10, Door4), the ensemble will not take any sensing actions (as discussed in Section IV), so we strengthen it by sensing with probability 0.5 at each timestep. More sophisticated sensing strategies can be considered but require more task-specific knowledge to design; see Appendix G for more discussion.

Figure 5 compares the training performance of all algorithms across the six environments. In Section IV-C, we proved monotonic improvement when optimizing an unconstrained objective; the clipped surrogate PPO objective that BRPO uses still yields improvement from the initial policy. Note that BRPO's initial policy does not exactly match the ensemble: the random initialization for the residual policy network adds zero-mean noise around the ensemble policy. This may result in an initial drop relative to the ensemble, as in Figure 5c and Figure 5d.

On the wide variety of problems we have considered, BRPO agents perform dramatically better than BPO and UP-MLE agents. BPO and UP-MLE were unable to match the performance of BRPO, except on the simple Cartpole environment. This seems to be due to the complexity of the latent MDPs, discussed further in Section H. In fact, for Maze4 and Maze10, we needed to modify the reward function to encourage information-gathering for BPO and UP-MLE; without such reward bonuses, they were unable to learn any meaningful behavior. Even with the bonus, these agents only partially learn to solve the task. We study the effect that such a reward bonus would have on BRPO in Appendix F. For the simpler Cartpole environment, both BPO and UP-MLE learned to perform optimally but required much more training time than BRPO.

[3]This was originally introduced in Lee et al. [25].

VI. Discussion and Future Work

In the real world, robots must deal with uncertainty, either due to complex latent dynamics or task specifics. Because uncertainty is an inherent part of these tasks, we can at best aim for optimality under uncertainty, i.e., Bayes optimality. Existing Bayesian RL algorithms or POMDP solvers do not scale well to problems with complex continuous latent MDPs or a large set of possible MDPs.

Our algorithm, Bayesian Residual Policy Optimization, builds on an ensemble of experts by operating within the resulting residual belief MDP. We prove that this strategy preserves guarantees, such as monotonic improvement, from the underlying policy optimization algorithm. The scalability of policy gradient methods, combined with task-specific expertise, enables BRPO to quickly solve a wide variety of complex problems, such as navigating through a crowd of pedestrians. BRPO improves on the original ensemble of experts and achieves much higher rewards than existing Bayesian RL algorithms by sensing more efficiently and acting more robustly.

Although out of scope for this work, a few key challenges remain. First is an efficient construction of an ensemble of experts, which becomes particularly important for continuous latent spaces with infinitely many MDPs. Infinitely many MDPs do not necessarily require infinite experts, as many may converge to similar policies. An important future direction is subdividing the latent space and computing a qualitatively diverse set of policies [26]. Another challenge is developing an efficient Bayes filter, which is an active research area. In certain occasions, the dynamics of the latent MDPs may not be accessible, which would require a learned Bayes filter. Combined with a tractable, efficient Bayes filter and an efficiently computed set of experts, we believe that BRPO will provide an even more scalable solution for BRL problems.

References

[1] Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

[2] Jonathan Baxter. Theoretical models of learning to learn. In *Learning to learn*, pages 71–94. Springer, 1998.

[3] Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

[4] Felix Berkenkamp and Angela P Schoellig. Safe and robust learning control with gaussian processes. In *2015 European Control Conference (ECC)*, pages 2496–2501. IEEE, 2015.

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

[6] Panpan Cai, Yuanfu Luo, Aseem Saxena, David Hsu, and Wee Sun Lee. Lets-drive: Driving in a crowd by learning from tree search. *arXiv preprint arXiv:1905.12197*, 2019.

[7] Min Chen, Emilio Frazzoli, David Hsu, and Wee Sun Lee. POMDP-lite for robust robot planning under uncertainty. In *IEEE International Conference on Robotics and Automation*, 2016.

[8] Sanjiban Choudhury, Mohak Bhardwaj, Sankalp Arora, Ashish Kapoor, Gireeja Ranade, Sebastian Scherer, and Debadeepta Dey. Data-driven planning via imitation learning. *The International Journal of Robotics Research*, 37(13-14):1632–1672, 2018.

[9] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[11] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *arXiv preprint arXiv:1806.02817*, 2018.

[12] Yoav Freund and Robert Schapire. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

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

[14] Michael Gimelfarb, Scott Sanner, and Chi-Guhn Lee. Reinforcement learning with multiple experts: A bayesian model combination approach. In *Advances in Neural Information Processing Systems*, pages 9528–9538, 2018.

[15] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

[16] Arthur Guez, David Silver, and Peter Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, 2012.

[17] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pages 5302–5311, 2018.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[20] David Hsu, Wee S Lee, and Nan Rong. What makes some pomdp problems easy to approximate? In *Advances in neural information processing systems*, pages 689–696, 2008.

[21] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[22] Gregory Kahn, Tianhao Zhang, Sergey Levine, and Pieter Abbeel. Plato: Policy learning using adaptive trajectory optimization. In *IEEE International Conference on Robotics and Automation*, pages 3342–3349. IEEE, 2017.

[23] Zico Kolter and Andrew Ng. Near-Bayesian exploration in polynomial time. In *International Conference on Machine Learning*, 2009.

[24] Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[25] Gilwoo Lee, Brian Hou, Aditya Mandalika, Jeongseok Lee, Sanjiban Choudhury, and Siddhartha S. Srinivasa. Bayesian policy optimization for model uncertainty. In *International Conference on Learning Representations*, 2019.

[26] Yao Liu, Zhaohan Guo, and Emma Brunskill. Pac continuous state online multitask reinforcement learning with identification. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 438–446. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[27] Russell Mendonca, Abhishek Gupta, Rosen Kralev, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Guided meta-policy search. *arXiv preprint arXiv:1904.00956*, 2019.

[28] Pedro A. Ortega, Jane X. Wang, Mark Rowland, Tim Genewein, Zeb Kurth-Nelson, Razvan Pascanu, Nicolas Heess, Joel Veness, Alexander Pritzel, Pablo Sprechmann, Siddhant M. Jayakumar, Tom McGrath, Kevin Miller, Mohammad Gheshlaghi Azar, Ian Osband, Neil C. Rabi-

nowitz, András György, Silvia Chiappa, Simon Osindero, Yee Whye Teh, Hado van Hasselt, Nando de Freitas, Matthew Botvinick, and Shane Legg. Meta-learning of sequential strategies. *arXiv preprint arXiv:1905.03030*, 2019.

[29] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, 2013.

[30] Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20 (124):1–62, 2019. URL http://jmlr.org/papers/v20/18-339.html.

[31] Chris J Ostafew, Angela P Schoellig, and Timothy D Barfoot. Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. In *IEEE International Conference on Robotics and Automation*, pages 4029–4036. IEEE, 2014.

[32] Chris J Ostafew, Angela P Schoellig, and Timothy D Barfoot. Conservative to confident: treating uncertainty robustly within learning-based control. In *IEEE International Conference on Robotics and Automation*, pages 421–427. IEEE, 2015.

[33] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[34] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation*, 2018.

[35] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, 2003.

[36] Neil C. Rabinowitz. Meta-learners' learning dynamics are unlike learners'. *arXiv preprint arXiv:1905.01320*, 2019.

[37] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. EPOpt: Learning robust neural network policies using model ensembles. In *International Conference on Learning Representations*, 2017.

[38] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *arXiv preprint arXiv:1903.08254*, 2019.

[39] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

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

[41] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *Journal on Autonomous Agents and Multiagent Systems*, 27(1):1–51, 2013.

[42] David Silver and Joel Veness. Monte-carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, 2010.

[43] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[44] Bradly C. Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

[45] Zachary Sunberg and Mykel Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces. In *International Conference on Automated Planning and Scheduling*, 2018.

[46] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[47] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 7332–7342, 2018.

[48] Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Robotics: Science and Systems*, 2017.

APPENDIX

A. Proofs of theorems and lemmas

Proof of Lemma 1. BRPO uses PPO for optimization [40]. PPO's clipped surrogate objective approximates the following objective,

$$\max_{\theta} \hat{\mathbb{E}} \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t - \beta \cdot \text{KL}(\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)) \right], \quad (4)$$

where π_θ is a policy parameterized by θ and $\pi_{\theta_{\text{old}}}$ is the policy in the previous iteration, which correspond to the current and previous residual policies $\pi_{r_i}, \pi_{r_{i-1}}$ in Algorithm 1. $\hat{A}$ is the generalized advantage estimate (GAE) and KL is the Kullback–Leibler divergence between the two policy distributions. PPO proves monotonic improvement for the policy's expected return by bounding the divergence from the previous policy in each update. This guarantee only holds if both policies are applied to the same residual MDP, i.e. the ensemble is fixed.

Proof of Lemma 3. (i) Base case, $T = 0$. It holds trivially since $\mathcal{M}$ and $\mathcal{M}_r$ share the same initial state distribution P_0. (ii) Assume it holds for $T = t$. Pick any ξ and let its last element be s. Consider an s'-extended sequence $\xi' = (\xi, s')$. Conditioned on ξ, the probability of ξ' is equal in $(\pi, \mathcal{M})$ and $(\pi_r, \mathcal{M}_r)$, which we can see by marginalizing over all state-action sequences:

$$\sum_{\tau'_r} \pi_r(\tau'_r|\xi) = \sum_{a_r} \pi_r(a_r|s) T_r(s'|s, a_r) \quad (5)$$

$$= \sum_{a_r} \pi_r(a_r|s) \sum_{a} T(s'|s, a) \pi_e(a - a_r|s) \quad (6)$$

$$= \sum_{a} \sum_{a_r} \pi_r(a_r|s) \pi_e(a - a_r|s) T(s'|s, a) \quad (7)$$

$$= \sum_{a} \pi(a|s) T(s'|s, a) \quad (8)$$

$$= \sum_{\tau'} \pi(\tau'|\xi) \quad (9)$$

The transition from (5) to (6) comes from (2) and (7) to (8) comes from (3). It follows that,

$$\pi(\xi') = \pi(\xi) \sum_{\tau'} \pi(\tau'|\xi) = \pi_r(\xi) \sum_{\tau'_r} \pi_r(\tau'_r|\xi) = \pi_r(\xi'),$$

which proves the lemma. Note that this proof directly leads to the proof of Theorem 2.

Proof of Theorem 4. From Lemma 1, we have that π_r monotonically improves on the residual MDP $\mathcal{M}_r$. From Theorem 2, monotonic improvement of π_r on $\mathcal{M}_r$ implies monotonic improvement of the mixture policy π on the actual MDP $\mathcal{M}$. If the initial residual policy's actions are small, the expected return of the mixture policy π on $\mathcal{M}$ is close to that of the ensemble π_e.

B. Experimental Environments

Crowd Navigation. At the beginning of the episode, initial pedestrian positions are sampled uniformly along the left and right sides of the environment. Speeds are sampled uniformly between 0.1 and 1.0 m/s. The agent observes each person's speed and position to estimate the goal distribution.

The agent starts at the bottom of the environment, with initial speed sampled uniformly from 0 to 0.4 m/s. The agent controls acceleration and steering angle, bounded between ± 0.12 m/s^2 and ± 0.1 rad. Pedestrians are modeled as a 1m diameter circle. The agent is modeled as a rectangular vehicle of 0.5 m width and 2 m length. A collision results in a terminal cost of $100 \cdot (2v)^2 + 0.5$. Successfully reaching the top of the environment produces terminal reward of 250, while navigating to the left or right side results in terminal cost of 1000. A per timestep penalty of 0.1 encourages the agent to complete the episode quickly.

Cartpole. The cartpole initializes with small initial velocity around the upright position. The environment terminates when the pole is more than 1.2 rad away from the vertical upright position or the cart is 4.0 m away from the center. The agent is rewarded by 1 for every step the cartpole survives. The environment has finite horizon of 500 steps.

Object Localization. The agent can control the end-effector in the (x, y, z) directions. The goal is to move the hand to the object without colliding with the environment or object. The agent observes the top and bottom shelf poses, end-effector pose, arm configuration, and the noise scale. The noise scale is the standard deviation of the Gaussian noise on the agent's observation of the object's pose. Without sensing, the noise is very large: $w \sim \mathcal{N}(0, 5.0^2)$ where the width of the shelf is only 0.35 m, When sensing is invoked, the noise is reduced to $w \sim \mathcal{N}(0, d^2)$ where d is the distance between the object and the end-effector.

Latent Goal Mazes. The agent observes its current position, velocity, and distance to all latent goals. If sensing is invoked, it also observes the noisy distance to the goal. In addition, the agent observes the categorical belief distribution over the latent goals. In `Maze4`, reaching the active goal provides a terminal reward of 500, while reaching an incorrect goal gives a penalty of 500. The task ends when the agent receives either the terminal reward or penalty, or after 500 timesteps. In `Maze10`, the agent receives a penalty of 50 and continues to explore after reaching an incorrect goal.

Doors. To check the doors, the agent can either sense (-1) or crash into them (-10). At every step, the agent observes its position, velocity, distance to goal, and whether it crashed or passed through a door. In addition, the agent observes the categorical distribution over the $2^4 = 16$ possible door configurations (from the Bayes filter) and the ensemble's recommendation. The agent receives a terminal reward of 100 if it reaches the goal within 300 timesteps.

C. Bayesian Reinforcement Learning and Posterior Sampling

Posterior Sampling Reinforcement Learning (PSRL) [29] is an online RL algorithm that maintains a posterior over latent MDP parameters ϕ. However, the problem setting it considers and how it uses this posterior are quite different than what we consider in this paper.

In this work, we are focused on scenarios where the agent can only interact with the test MDP for a *single episode*; latent parameters are resampled for each episode. The PSRL regret analysis assumes MDPs with finite horizons and *repeated episodes* with the same test MDP, i.e. the latent parameters are fixed for all episodes.

Before each episode, PSRL samples an MDP from its posterior over MDPs, computes the optimal policy for the sampled MDP, and executes it on the fixed test MDP. Its posterior is updated after each episode, concentrating the distribution around the true latent parameters. During this exploration period, it can perform arbitrarily poorly. Furthermore, sampling a latent MDP from the posterior determinizes the parameters; as a result, there is no uncertainty in the sampled MDP, and the resulting optimal policies that are executed will never take sensing actions.

The Gap between Bayes Optimality and Posterior Sampling. We present a toy problem to highlight the distinction between them.

Consider a deterministic tree-like MDP (Figure 7). Reward is received only at the terminal leaf states: one leaf contains a pot of gold ($R = 100$) and all others contain a dangerous tiger ($R = -10$). All non-leaf states have two actions, go left (L) and go right (R). The start state additionally has a sense action (S), which is costly ($R = -0.1$) but reveals the exact location of the pot of gold. Both algorithms are initialized with a uniform prior over the $N = 2^d$ possible MDPs (one for each possible location of the pot of gold).

To contrast the performance of the Bayes-optimal policy and posterior sampling, we consider the multi-episode setting where the agent repeatedly interacts with the same MDP. The MDP is sampled once from the uniform prior, and agents interact with it for T episodes. This is the setting typically considered by posterior sampling (PSRL) [29].

Before each episode, PSRL samples an MDP from its posterior over MDPs, computes the optimal policy, and executes it. After each episode, it updates the posterior and repeats. Sampling from the posterior determinizes the underlying latent parameter. As a result, PSRL will never produce sensing actions to reduce uncertainty about that parameter because the sampled MDP has no uncertainty. More concretely, the optimal policy for each tree MDP is to navigate directly to the gold *without sensing*; PSRL will never take the sense action. Thus, PSRL makes an average of $\frac{N-1}{2}$ mistakes before sampling the correct pot of gold location and the cumulative reward over T episodes is

$$\underbrace{-10\left(\tfrac{N-1}{2}\right)}_{\text{mistakes}} + \underbrace{100\left(T - \tfrac{N-1}{2}\right)}_{\text{pot of gold}} \tag{10}$$

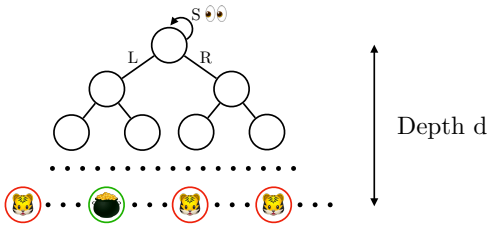


Fig. 7: A tree-like MDP that highlights the distinction between BRL and PSRL.

In the first episode, the Bayes-optimal first action is to sense. All subsequent actions in this first episode navigate toward the pot of gold, for an episode reward of $-0.1+100$. In the subsequent $T-1$ episodes, the Bayes-optimal policy navigates directly toward the goal without needing to sense, for a cumulative reward of $100T-0.1$. The performance gap between the Bayes-optimal policy and posterior sampling grows exponentially with depth of the tree d.

Practically, a naïve policy gradient algorithm (like BPO) would struggle to learn the Bayes-optimal policy: it would need to learn to both sense and navigate the tree to the sensed goal. BRPO can take advantage of the set of experts, in which each navigate to their designated leaf. During training, the BRPO agent only needs to learn to balance sensing with navigation.

D. Maximum A Posterior as ensemble of experts

One choice for the ensemble policy π_e is to select the maximum a posteriori (MAP) action, $a_{\text{MAP}} = \arg\max_a \sum_{i=1}^{k} b(\phi_i)\pi_i(a|s)$. However, computing the MAP estimate may require optimizing a non-convex function, e.g., when the distribution is multimodal. We can instead maximize the lower bound using Jensen's inequality.

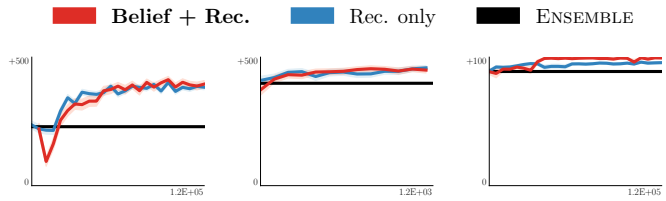
$$\log \sum_{i=1}^{k} b(\phi_i)\pi_i(a|s) \geq \sum_{i=1}^{k} b(\phi_i) \log \pi_i(a|s) \tag{11}$$

This is much easier to solve, especially if $\log \pi_i(a|s)$ is convex. If each $\pi_i(a|s)$ is a Gaussian with mean μ_i and covariance Σ_i, e.g. from TRPO [39], the resultant action is the belief-weighted sum of mean actions:

$$\begin{aligned} a^* &= \arg\max_a \sum_{i=1}^{k} b(\phi_i) \log \pi_i(a|s) \\ &= \left[\sum_{i=1}^{k} b(\phi_i)\Sigma_i^{-1}\right]^{-1} \sum_{i=1}^{k} b(\phi_i)\Sigma_i^{-1}\mu_i \end{aligned}$$

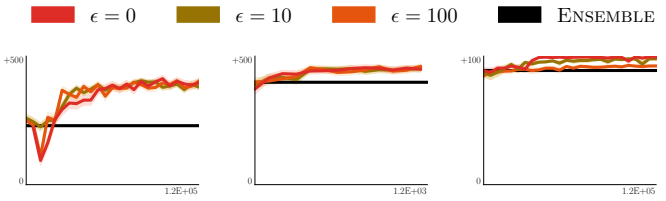
E. Ablation Study: Residual Policy Inputs

The BRPO policy takes the belief distribution, state, and ensemble recommendation as inputs (Figure 3). We considered two versions of BRPO with different inputs - only recommendation (which implicitly encodes belief), and one with both recommendation and belief.



(a) Maze4 (b) Maze10 (c) Door4

Fig. 8: Ablation study on input features. Including both belief and recommendation as policy inputs results in faster learning in Door4.



(a) Maze4 (b) Maze10 (c) Door4

Fig. 9: Ablation study on information-gathering reward (Equation 12). BRPO is robust to information-gathering reward.

The results show that providing both belief and recommendation as inputs to the policy are important (Figure 8). Although BRPO with only the recommendation performs comparably to BRPO with both inputs on Maze4 and Maze10, the one with both inputs produce faster learning on Door4.

F. Ablation Study: Information-Gathering Reward Bonuses

Because BRPO maximizes the Bayesian Bellman equation (Equation 1), exploration is incorporated into its long-term objective. As a result, auxiliary rewards to encourage exploration are unncessary. However, existing work that does not explicitly consider the belief has suggested various auxiliary reward terms to encourage exploration, such as surprisal rewards [1] or intrinsic rewards [33]. To investigate whether such rewards benefit the BRPO agent, we augment the reward function with the following auxiliary bonus from [7]:

$$\tilde{r}(s,b,a) = r(s,b,a) + \epsilon \cdot \mathbb{E}_{b'}\left[\|b - b'\|_1\right] \qquad (12)$$

where $\|b - b'\|_1 = \sum_{i=1}^{k} |b(\phi_i) - b'(\phi_i)|$ rewards change in belief.

Figure 9 summarizes the performance of BRPO when training with $\epsilon = 0, 10, 100$. Too much emphasis on information-gathering causes the agent to over-explore and therefore underperform. In Door4 with $\epsilon = 100$, we qualitatively observe that the agent crashes into the doors more often. Crashing significantly changes the belief for that door; the huge reward bonus outweighs the penalty of crashing from the environment.

We find that BPO and UP-MLE are unable to learn without an exploration bonus on Maze4, Maze10, and Door4. We used $\epsilon = 1$ for Maze4 and Door4, and $\epsilon = 100$ for Maze10. Upon qualitative analysis, we found that the bonus helps BPO and UP-MLE learn to sense initially, but the algorithms are unable to make further progress. We believe that this is because solving the latent mazes is challenging.

In addition to this study, we have performed two additional ablations on input features to the residual policy (Appendix E) and hand-tuned ensembles that are better at sensing (Appendix G). Including both the belief and ensemble recommendation as inputs to the residual policy produces faster learning. BRPO takes advantage of the stronger ensemble and continues to improve on that better baseline.

G. Ablation Study: Better Sensing Ensemble

The ensemble we used for training BRPO in Figure 5 randomly senses with probability 0.5. A more effective sensing ensemble baseline policy could be designed manually, and used as the initial policy for the BRPO agent to improve on. Note that in general, designing such a policy can be challenging: it requires either task-specific knowledge, or solving an approximate Bayesian RL problem. We bypass these requirements by using BRPO.

On the Maze10 environment, we have found via offline tuning that a more effective ensemble baseline agent senses only for the first 150 of 750 timesteps. Table I shows that BRPO results in higher average return and success rate. The performance gap comes from the suboptimality of the ensemble recommendation, as experts are unaware of the penalty for reaching incorrect goals.

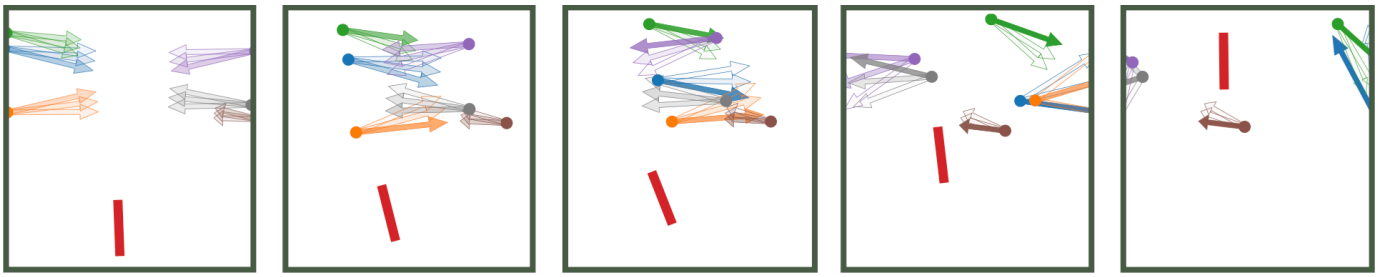
	BRPO	RandomSensing	BetterSensing
Avg. Return	**465.7** $\pm$ 4.7	409.5 $\pm$ 10.8	416.3 $\pm$ 9.4
Success Rate	**100%**	-	96.3%

TABLE I: Comparison of BRPO and ensembles on Maze10.

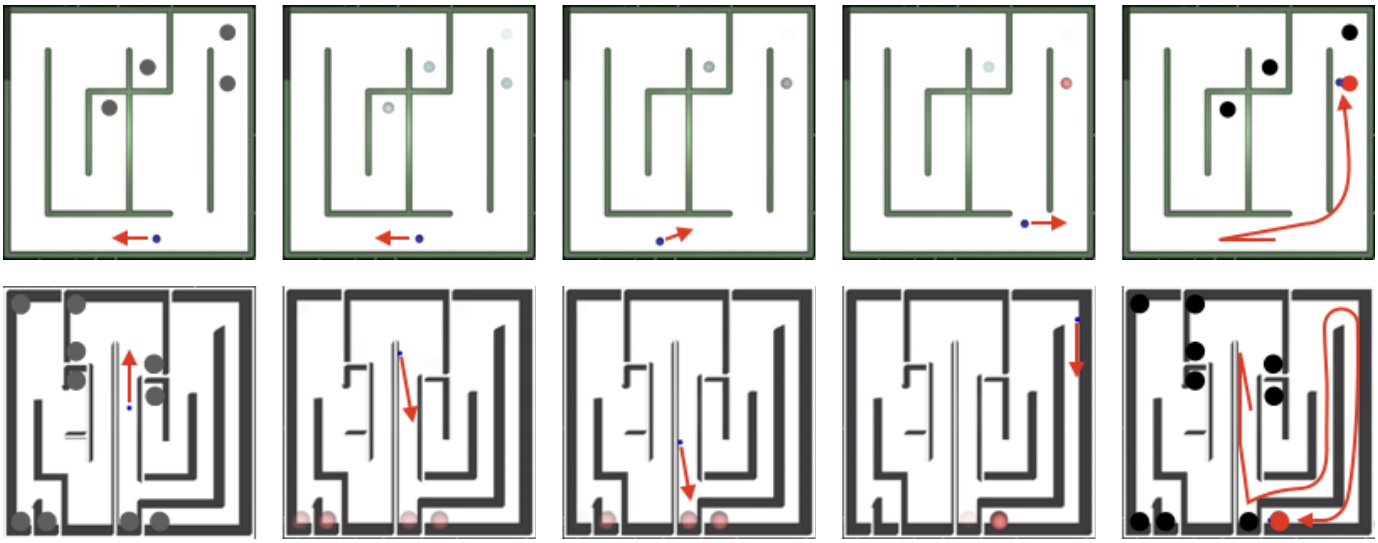
H. Qualitative Behavior Analysis

Figure 10 shows some representative trajectories taken by BRPO agents. Across multiple environments (CrowdNav, Maze4, Maze10), we see that BRPO agent adapts to the evolving posterior. As the posterior over latent goals updates, the agent shifts directions. While this rerouting partly emerges from the ensemble policies as the posterior sharpens, BRPO's residual policy reduces uncertainty (Maze4, Maze10) and pushes the agent to navigate faster, resulting in higher performance than the ensembles.

For Maze4, Maze10 and Door4, we have visualized where the agent invokes explicit sensing (Figure 6). For Maze4 and Maze10, the BRPO agent learns to sense when goals must be distinguished, e.g. whenever the road diverges. For Door4, it senses when that is most cost-effective: near the doors, where accuracy is highest. This results in a rather interesting policy (Figure 10c). The agent dashes to the wall, senses only once or twice, and drives through the closest open door. The BRPO agent avoids crashing in almost all scenarios.



(a) **`CrowdNav`**. Arrows are the directions to discrete latent goals. Each arrow's transparency indicate the posterior probability of the corresponding goal, and its length indicate the speed. The agent changes its direction as it forsees collision in its original plan.



(b) Latent goal mazes with four (**`Maze4`**) and ten (**`Maze10`**) possible goals. The agent senses as it navigates, changing its direction as goals are deemed less likely (more transparent). We have marked the true goal with red in the last frame for clarity.

(c) **`Door4`**. The agent senses only when it is near the wall with doors, where sensing is most accurate. The transparency of the red bars indicates the posterior probability that the door is blocked. With sensing, the agent notices that the third door is likely to be open.

Fig. 10: BRPO policy keyframes. Best viewed in color.